V. PROGRAMMES INFORMATIQUES DESTINES A EFFECTUER LE TRAITEMENT DES DONNEES

```
program nwLevels;
{This program analyses all the behaviors of the subject and writes one common outputfile with the time, and some informations
about the event.}
uses nwstat,wallfunctions,nperslib,genlib,CCD,Common,AnaLib;


const prefix                   = 'Level';
      precedingforoutput       = 'gen';
      aftertime                =  Round(3.0 * 100);
      outfilename              = 'Levelall';
      relevantcodes            = [{power}1,2,3,4,5,6,7,8,10,11,
                                  12,13,14,16,17,18,19,20,
                                  23,24,25,26,27,28,
                                  31,32,33,34,35,36,37,38,40,
                                  41,42,43,44,46,47,48,51,52,53,
                                  85,86,87,88,91,92,93,94,
                                  100,101,102,103,104,105,106,107,108,
                                  109,110,111,112,113,114,115,116,
                                  117,118,119,120,121,122,123,124,
                                  125,126,127,
                                  {prisonandpower}9,15,21,29,30,39,45,49,50,
                                  89,90,95,96,128,129,130,131,
                                  {speed}54,55,56,57,58,59,60,61,62,63,64,
                                  {shield}67,68,69,70,71,72,
                                  {hourglass}73,74,75,
                                  {risk}65,66,
                                  {repair}79,139,
                                  {telephone}76,77,
                                  {wall}80,81,82,83,84,
                                  {home}97,{bonus}{98,}99];

var   CollectiveTable          : TCollectiveCodeData;
      alldone,prepared,
      prepare2nd               : boolean;{true= quand l'analyse a ,t, faite sur tous les fichiers}
      lastpos                  : integer;{nombre a partir duquel il faut g,n,rer des num,ros de code}
      dirinfilename            : string;
      mbfound                  : boolean;
      danger,fatality,
      visibleenemies           : real;
      closestEnemy             : Char;
      subjectname              : integer;
      nothing,x,i              : integer;
      collective2nd            : text;


procedure writealloutput(var colltable: Tcollectivecodedata;var outfile: text;
                         fname: integer; mbcode: integer; var check: boolean);
    begin
        incCollectiveCode(colltable,mbcode);
        if mbcode < 10 then
          write(outfile,fname:4,'   ',mbcode,' ')
        else if mbcode < 100 then
          write(outfile,fname:4,'  ',mbcode,' ')
        else if mbcode >= 100 then
          write(outfile,fname:4,' ',mbcode,' ');
        check:= true;
    end;

procedure AnalyzeWalltools(var colloutFile : Text;var collectivetable:TCollectiveCodeData;
                           fn : String; name: integer);

  var   eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;
        Walltime,entertime,victorytime,
        checkfromthere,bonustime          : Ttime;
        x,i                               : integer;
        wallresult,cause                  : char;
        power,fast,entersafety,victory,
        bonus,wait                        : boolean;


  begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);

    i:= 0; {total of walls}


    (*** find wall application events ***)
    while eventSearch.MoveToNextEvent(apply_wall_snd,anylevel) do begin
        walltime:= eventsearch.eventinfos.time;
        if not statesearch.movetostateattime(walltime)then;

        if not statesearch.movetoposition(statesearch.position - 1) then;
        if (statesearch.stateinfos.mindistJ = -1) and
         (statesearch.stateinfos.mindistH = -1) and
          (statesearch.stateinfos.mindistF = -1) and
           (statesearch.stateinfos.mindistE  > -1) then
            closestEnemy:= 'E'
        else if (statesearch.stateinfos.mindistJ > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistJ <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'J'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistH > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistH <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'H'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistF > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistF <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'F'
            else closestEnemy:= 'E';
        end
        else closestEnemy:= 'Z';

        danger:= statesearch.stateinfos.danger;
        fatality:= statesearch.stateinfos.fatality;
        visibleenemies:= statesearch.stateinfos.nbrofvisibleenemies;

        mbfound:= false;
        power:= false;
        fast:= false;
        victory:= false;
        entersafety:= false;
        entertime:= 0;
        victorytime:= 0;
        checkfromthere:= 0;
        bonus:= false;
        bonustime:= 0;
        wait:= false;
        inc(i);

        wallresult:= walltest(statesearch,eventsearch);
        if statesearch.stateinfos.onoffstates[ispowered] then
          power:= true;
        if statesearch.stateinfos.onoffstates[isfast] then
          fast:= true;

        if power then begin
            if not statesearch.movetostateattime(walltime) then;
            repeat
                if not (statesearch.stateinfos.onoffstates[ispowered]) then begin
                    if not statesearch.movetoposition(statesearch.position + 1) then;
                    checkfromthere:= statesearch.stateinfos.time;
                end;
                if not statesearch.movetoposition(statesearch.position - 1) then;
            until (checkfromthere > 0);

            entertime:= checkenterfromstarttostop(statesearch, eventsearch,checkfromthere,walltime);
            if (entertime > 0) then
              entersafety := true;
            victory:= OHEventExists(fn,esvictories,checkfromthere,walltime,victorytime);
            cause:= causalsearch(eventsearch,checkfromthere);
            if cause = 'B' then
              if waitforjanus(statesearch,checkfromthere) then
                wait:= true
              else;

        end{power}

        else if fast then begin
            if not statesearch.movetostateattime(walltime) then;
            repeat
                if not (statesearch.stateinfos.onoffstates[isfast]) then begin
                    if not statesearch.movetoposition(statesearch.position + 1) then;
                    checkfromthere:= statesearch.stateinfos.time;
                end;
                if not statesearch.movetoposition(statesearch.position - 1) then;
            until checkfromthere > 0;

            entertime:= checkenterfromstarttostop(statesearch,eventsearch, checkfromthere,walltime);

            if entertime > 0 then
              entersafety := true;
            bonus:= OHEventExists(fn,[bonus_get_snd],checkfromthere,walltime,bonustime);
        end;

        if not statesearch.movetostateattime(walltime)then;
        if not eventsearch.movetofirsteventatoraftertime(walltime)then;


        if wallresult = 'F' then
          writealloutput(collectivetable,colloutfile,name,82,mbfound)

        else if wallresult = 'G' then
          writealloutput(collectivetable,colloutfile,name,81,mbfound)

        else if wallresult = 'A' then begin
            if power then begin

                if ( (cause = 'A') and (victory or entersafety) ) or
                  ( (cause = 'B') and
                   ( (not wait) or victory or entersafety) ) then
                    writealloutput(collectivetable,colloutfile,name,83,mbfound);
            end
            else if fast then begin

                if ( entersafety or bonus ) then
                  writealloutput(collectivetable,colloutfile,name,83,mbfound);
            end
            else
              writealloutput(collectivetable,colloutfile,name,83,mbfound);
                             {no power no fast}
        end{wall = A}

        else if wallresult = 'B' then begin

            if power then begin

                if ( (cause = 'A') and (victory or entersafety) ) or
                  ( (cause = 'B') and
                   ( (not wait) or victory or entersafety) ) then
                     writealloutput(collectivetable,colloutfile,name,84,mbfound);
            end
            else if fast then begin

                if ( entersafety or bonus ) then
                  writealloutput(collectivetable,colloutfile,name,84,mbfound);
            end
            else
              writealloutput(collectivetable,colloutfile,name,84,mbfound);{no power no fast}
        end{wall = B}

        else if wallresult = 'C' then begin
            if power then begin
                if ( (cause = 'A') and (victory or entersafety) ) or
                  ( (cause = 'B') and
                   ( (not wait) or victory or entersafety) ) then
                     writealloutput(collectivetable,colloutfile,name,83,mbfound);
            end
            else if fast then begin
                if ( entersafety or bonus ) then
                  writealloutput(collectivetable,colloutfile,name,83,mbfound);
            end
            else writealloutput(collectivetable,colloutfile,name,83,mbfound);{no power no fast}
        end{wall = C}

        else if wallresult = 'D' then begin
            if power then begin

                if ( (cause = 'A') and (victory or entersafety) ) or
                  ( (cause = 'B') and
                   ( (not wait) or victory or entersafety) ) then
                     writealloutput(collectivetable,colloutfile,name,83,mbfound);
            end
            else if fast then begin
                if ( entersafety or bonus ) then
                  writealloutput(collectivetable,colloutfile,name,83,mbfound);
            end
            else
              writealloutput(collectivetable,colloutfile,name,83,mbfound);{no power no fast}
        end{wall = D}

        else if wallresult = 'E' then begin
            if power then begin
                if ( (cause = 'A') and (victory or entersafety) ) or
                  ( (cause = 'B') and
                   ( (not wait) or victory or entersafety) ) then
                     writealloutput(collectivetable,colloutfile,name,80,mbfound);
            end
            else if fast then begin
                if ( entersafety or bonus ) then
                  writealloutput(collectivetable,colloutfile,name,80,mbfound);
            end
            else
              writealloutput(collectivetable,colloutfile,name,80,mbfound);{no power no fast}
        end{wall = E}

        else if wallresult = 'H' then begin
            if power then begin
                if ( (cause = 'A') and (victory or entersafety) ) or
                  ( (cause = 'B') and
                   ( (not wait) or victory or entersafety) ) then
                     writealloutput(collectivetable,colloutfile,name,84,mbfound);
            end
            else if fast then begin
                if ( entersafety or bonus ) then
                  writealloutput(collectivetable,colloutfile,name,84,mbfound);
            end
            else
              writealloutput(collectivetable,colloutfile,name,84,mbfound);{no power no fast}
        end;{wall = H}

        if mbfound then begin
            write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
            write(colloutfile,closestEnemy,'  ');
            write(colloutfile,visibleenemies:3:0);

            write(colloutfile,(danger * (danger + fatality) ):6:2 );
            write(colloutfile,'  ',walltime/100:10:2);
            write(colloutfile, '   ', time2string(walltime/100));
            writeln(colloutfile);
        end;

    end;{while new wall applied}

    eventSearch.Done;
    stateSearch.Done;
```

```
  end;{walltools analysis: whole procedure}


procedure Analyzetelephone(var colloutfile: Text;var collectivetable:TCollectiveCodeData;
                                fn : String; name: integer);

  var   eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;
        levelend,
        telephonetime                     : TTIme;
        subjectdone,safeapply,prison,
        maladapted                        : Boolean;
        x,i,b,position                    : integer;

  begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);

    i:= 0; {total of telephones}


    (*** find telephone application events ***)
    while eventSearch.MoveToNextEvent(apply_bell_snd,anylevel) do begin
        if not stateSearch.MoveToStateAtTime(eventSearch.eventInfos.time) then begin
          WriteLn('Unable to find state');
          Break
        end;{v,rification que dans le fichier mes l',tat isshielded existe au meme moment}

        (*initialization for each new shieldmode found, before analysis*)
        inc(i);
        safeapply:= false;
        maladapted:= false;
        prison:= false;
        levelend:= 0; (*necessary for the findendofmode function *)
        telephonetime:= eventSearch.eventinfos.time;

        if not statesearch.movetostateattime(telephonetime)then;
        if not statesearch.movetoposition(statesearch.position - 1) then;
        if (statesearch.stateinfos.mindistJ = -1) and
         (statesearch.stateinfos.mindistH = -1) and
          (statesearch.stateinfos.mindistF = -1) and
           (statesearch.stateinfos.mindistE  > -1) then
            closestEnemy:= 'E'
        else if (statesearch.stateinfos.mindistJ > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistJ <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'J'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistH > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistH <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'H'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistF > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistF <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'F'
            else closestEnemy:= 'E';
        end
        else closestEnemy:= 'Z';

        danger:= statesearch.stateinfos.danger;
        fatality:= statesearch.stateinfos.fatality;
        visibleenemies:= statesearch.stateinfos.nbrofvisibleenemies;

        mbfound:= false;


        if not statesearch.movetostateattime(telephonetime)then;
        if not statesearch.movetoposition(statesearch.position - 1)then;
        if isSafePlace(statesearch) and notallenemiesinprison(statesearch) then
            safeapply:= true;

        if (statesearch.stateinfos.onoffstates[isPlayerInPrison] = true) then
          prison:= true;

        {check if maladapted}
        if not statesearch.movetostateattime(telephonetime) then;
        if not statesearch.movetoposition(statesearch.position - 1) then;
        if (statesearch.stateinfos.integrity = 1) and
         ((statesearch.stateinfos.nbrofvisibleenemies = 0) or
           ((notallenemiesinprison(statesearch) = false) and
            (statesearch.stateinfos.onoffstates[isplayerinprison] = false) ) )  then
             maladapted := true;


        (*compute results of the analysis *)
        if (safeapply) or (prison) then
          writealloutput(collectivetable,colloutfile,name,77,mbfound)


        else if (not safeapply) and (not prison) then
          writealloutput(collectivetable,colloutfile,name,76,mbfound);

        if mbfound then begin
            write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
            write(colloutfile,closestEnemy,'  ');
            write(colloutfile,visibleenemies:3:0);
            write(colloutfile,(danger * (danger + fatality) ):6:2 );
            write(colloutfile,'  ',telephonetime/100:10:2);
            write(colloutfile, '   ', time2string(telephonetime/100));
            writeln(colloutfile);
        end;

    end;{while new telephone applied}

    eventSearch.Done;
    stateSearch.Done;
  end;{telephone analysis: whole procedure}


procedure AnalyzeSpeedmode(var colloutfile: Text;var collectivetable:TCollectiveCodeData;
                              fn : String;  name: integer);

  const aftertime                        =  Round(3.0 * 100);

  var   totalspeeds                       : T_ccd_value_type;
        eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;
        starttime,levelend,stoptime_temp,
        stoptime,entertime,collecttime,
        quittime,walltime,risktime,
        bonustime,endofbonus,killtime,
        hurttime,pursuitstoptime,
        toolcollecttime                   : TTIme;
        power,bonustaken,
        safestart,unsafestart,risk,
        collected, Path_A, Path_B,
        quitsafety,entersafety,toolcollect,
        bonuscollect,safecollect,
        bonusmissed,escape,
        subjectdone,bonusvisible,
        lastbonusdistance,playerhurt,
        playerkilled                      : Boolean;
        x,i,b,position,fatalrisk          : integer;
        wallresult                        : char;

  begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);
    subjectdone:= false;
    i:= 0; {total of fast tools}


    (*** find fast application events ***)
    while eventSearch.MoveToNextEvent(apply_fast_snd,anylevel) do begin
        if not stateSearch.MoveToStateAtTime(eventSearch.eventInfos.time) then begin
          WriteLn('Unable to find state');
          Break
        end;{v,rification que dans le fichier mes l',tat isfast existe au meme moment}
        starttime:= eventSearch.eventinfos.time;

        if not statesearch.movetoposition(statesearch.position - 1) then;
        if (statesearch.stateinfos.mindistJ = -1) and
         (statesearch.stateinfos.mindistH = -1) and
          (statesearch.stateinfos.mindistF = -1) and
           (statesearch.stateinfos.mindistE  > -1) then
            closestEnemy:= 'E'
        else if (statesearch.stateinfos.mindistJ > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistJ <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'J'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistH > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistH <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'H'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistF > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistF <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'F'
            else closestEnemy:= 'E';
        end
        else closestEnemy:= 'Z';

        danger:= statesearch.stateinfos.danger;
        fatality:= statesearch.stateinfos.fatality;
        visibleenemies:= statesearch.stateinfos.nbrofvisibleenemies;

        mbfound:= false;
        (*initialization for each new speedmode found, before analysis*)
        inc(i);
        levelend:= 0; (*necessary for the findendofmode function *)
        stoptime_temp:= 0;
        safestart:= false;
        quitsafety:= false;
        entersafety:= false;
        wallresult:= 'z';
        power:= false;
        bonustaken:= false;
        bonusvisible:= false;
        bonusmissed:= false;
        lastbonusdistance:= false;
        endofbonus:= 0;
```

```
pursuitstoptime:= 0;
toolcollecttime:= 0;
hurttime:= 0;
killtime:= 0;
playerhurt:= false;
playerkilled:= false;
toolcollect:= false;
safecollect:= false;
escape:= false;
risktime:= 0;
entertime:= 0;
quittime:= 0;
bonustime:= 0;

stoptime_temp:= findendofmode(statesearch,fn,starttime,isFast,subjectdone,levelend);

(** if levelend > 0 the analysis has to stop there, stoptime
must thus take the value levelend **)
if levelend > 0 then
  stoptime:= levelend
else stoptime:= stoptime_temp;


(* check visibility of bonus before speedmode*)
if (statesearch.stateinfos.onoffstates[isBonusVisible]) then
    bonusvisible:= true;
if not statesearch.movetostateattime(starttime)then;


(*check if player becomes powered at the same time*)
if (checkfromstarttostop(statesearch,starttime,stoptime,isPowered) >0) then begin
    power:= true;
    if not statesearch.movetostateattime(starttime)then;
end;

(* check if fastmode started in a safeplace *)
if not statesearch.movetostateattime(starttime)then;
if isSafeplace(statesearch) and notallenemiesinprison(statesearch) then begin
    safestart:= true;


    (*check if player exits the safeplace*)
    quittime:= (checkexitfromstarttostop(statesearch,starttime,stoptime));
    if quittime > 0 then
        quitsafety:= true;
end;

(*check if player enters and stays in safe place*)
if safestart = false then begin
    entertime:= checkenterfromstarttostop(statesearch,eventsearch,starttime,stoptime);
    if (entertime > 0) and ((levelend = 0)or(entertime < levelend)) then
        entersafety:= true;
end;

(* check if player places a wall and if yes, what for *)
if oheventexists(fn,[apply_wall_snd],starttime,stoptime,walltime) then begin
    if not eventsearch.movetofirsteventattime(walltime)then;
    if not statesearch.movetostateattime(walltime)then;
    wallresult:= walltest(statesearch,eventsearch);
end;

(* check if player uses a risk tool *)
if oheventexists(fn,[apply_teleport_snd],starttime,stoptime,risktime) then
    risk:= true;

(* check if player is hurt *)
if oheventexists(fn,esplayerhurt,starttime,stoptime + aftertime,hurttime) then
    playerhurt:= true;


(* check if player dies  just at the end of fastmode or during *)
if oheventexists(fn,[die_snd],starttime,stoptime + aftertime,killtime) then
    playerkilled:= true;

(* check if player collects object *)
if oheventexists(fn,(escollecttools + [get_super_snd]),starttime,stoptime,toolcollecttime) then begin
    Toolcollect:= true;
    if not statesearch.movetostateattime(toolcollecttime) then;
    if( isSafeplace(statesearch) and notallenemiesinprison(statesearch)) then
      safecollect:= true;
end;


if bonusvisible then begin
    if not statesearch.movetostateattime(starttime) then;
    repeat
       if not (statesearch.stateinfos.onoffstates[isbonusvisible]) then begin
         endofbonus:= statesearch.stateinfos.time;
         break;
       end;
       if not statesearch.movetoposition(statesearch.position + 1) then
         endofbonus:= statesearch.stateinfos.time;
    until endofbonus > 0;


    (* check if bonus collected *)
    if eventexists(fn,[bonus_get_snd],starttime,endofbonus) then
        bonustaken:= true;

    pursuitstoptime:= endofbonus;
```

```
     if (playerhurt and (hurttime < pursuitstoptime)) then begin
         pursuitstoptime:= hurttime;
         if not statesearch.movetostateattime(hurttime)then;
         if not statesearch.movetoposition(statesearch.position - 1) then;
         if (statesearch.stateinfos.bonusdistance < 4 ) then begin
            if not statesearch.movetostateattime(endofbonus)then;
            if not statesearch.movetoposition(statesearch.position - 1) then;
            if not (statesearch.stateinfos.bonusdistance > 10) then
              lastbonusdistance:= true;
         end;
     end{playerhurt}
     else if (playerkilled and (killtime < pursuitstoptime)) then begin
         pursuitstoptime:= killtime;
         if not statesearch.movetostateattime(killtime)then;
         if not statesearch.movetoposition(statesearch.position - 1) then;
         if (statesearch.stateinfos.bonusdistance < 4 ) then begin
            if not statesearch.movetostateattime(endofbonus)then;
            if not statesearch.movetoposition(statesearch.position - 1) then;
            if not (statesearch.stateinfos.bonusdistance > 10) then
              lastbonusdistance:= true;
         end;
     end{playerkilled}
     else begin
         if not statesearch.movetostateattime(endofbonus)then;
         if not statesearch.movetoposition(statesearch.position - 1) then;
         if (statesearch.stateinfos.bonusdistance < 4)  and
            (statesearch.stateinfos.bonusdistance > -1)then
           lastbonusdistance:= true;
     end;{not hurt, not killed}
     if pursuitbonus(statesearch,'bonus',starttime,pursuitstoptime)
       and lastbonusdistance then
         bonusmissed:= true;
end;{bonus visible}


(*compute results of the analysis *)

if not power then begin
    if safestart then begin
        if quitsafety then begin
            if (bonusvisible and bonustaken) then begin
                if (wallresult <> 'z') then begin  {walltime = smaller otherwise the event would not have been found}
                    if wallresult in blockemptyplace then
                      writealloutput(collectivetable,colloutfile,name,63,mbfound){block empty place}
                    else if wallresult in blockenemieselsewhere then
                      writealloutput(collectivetable,colloutfile,name,62,mbfound){block enemies elsewhere}
                    else if wallresult in blockprisoners then
                      writealloutput(collectivetable,colloutfile,name,64,mbfound){block prisoners}
                    else
                      writealloutput(collectivetable,colloutfile,name,55,mbfound){wall but no block}
                end{wall}
                else if (wallresult = 'z') then
                  writealloutput(collectivetable,colloutfile,name,55,mbfound){no wall}
            end{bonustaken}
            else begin
                if bonusmissed then begin
                    if (wallresult <> 'z') then begin
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,63,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,62,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,64,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,56,mbfound){wall but no block}
                    end{wall}
                    else if (wallresult = 'z') then
                      writealloutput(collectivetable,colloutfile,name,56,mbfound){no wall}
                end{bonusmissed}
                else begin
                    if (wallresult <> 'z') then begin
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,63,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,62,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,64,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,58,mbfound){wall but no block}
                    end{wall}
                    else if (wallresult = 'z') then
                      writealloutput(collectivetable,colloutfile,name,58,mbfound);{no wall}
                end;{no bonusmissed}
            end;{bonus not taken}
        end{quit safety}
        else if not quitsafety then
          writealloutput(collectivetable,colloutfile,name,54,mbfound){stay safe}
    end{safestart}
    else if not safestart then begin
        if (entersafety and (not safecollect) ) then begin
            if bonusvisible and bonustaken then begin
                if (wallresult <> 'z') then begin
                    if (entertime < walltime) and
                     ((killtime = 0) or (killtime > entertime))then
                       writealloutput(collectivetable,colloutfile,name,57,mbfound){entersafety before}
                    else if (walltime <= entertime) then begin
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,61,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,59,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,60,mbfound){block prisoners}
                        else if (entertime < bonustime) and
                         ((killtime = 0) or (killtime > entertime)) then
                           writealloutput(collectivetable,colloutfile,name,57,mbfound){wall but not blocking}
                        else
                          writealloutput(collectivetable,colloutfile,name,55,mbfound)
                          {bonus before enter or bonus after enter but enter after death}
                    end{wall first}
                end{wall}
                else if ( wallresult = 'z') and
                 ((killtime = 0) or (killtime > entertime))then
                   writealloutput(collectivetable,colloutfile,name,57,mbfound){no wall}
                else
                  writealloutput(collectivetable,colloutfile,name,55,mbfound)
                  {loss before enter no wall but bonus...}
            end{bonustaken}
            else begin
                if (wallresult <> 'z') then begin
                    if (entertime < walltime) and
                     ((killtime = 0) or (killtime > entertime))then
                       writealloutput(collectivetable,colloutfile,name,57,mbfound){entersafety before}
                    else begin {wallbefore or after kill + enter}
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,61,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,59,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,60,mbfound){block prisoners}
                        else if wallresult = 'F' then
                          writealloutput(collectivetable,colloutfile,name,54,mbfound){wall but not blocking}
                        else if ((killtime = 0) or (killtime > entertime)) then
                          writealloutput(collectivetable,colloutfile,name,57,mbfound)
                        else
                          writealloutput(collectivetable,colloutfile,name,54,mbfound);
                          {no bonus wall + continue enter but after loss only}
                    end{wall first}
                end{wall}
                else if (wallresult = 'z') and
                 ((not playerkilled) or (killtime > entertime)) then
                   writealloutput(collectivetable,colloutfile,name,57,mbfound){no wall}
                else
                  writealloutput(collectivetable,colloutfile,name,54,mbfound);
                  {no bonus no wall enter but after loss only}
            end{no bonustaken}
        end{entersafety}
        else if not ( entersafety  and (not safecollect) )then begin
            if (bonusvisible and bonustaken) then begin
                if (wallresult <> 'z') then begin
                    if wallresult in blockemptyplace then
                      writealloutput(collectivetable,colloutfile,name,61,mbfound){block empty place}
                    else if wallresult in blockenemieselsewhere then
                      writealloutput(collectivetable,colloutfile,name,59,mbfound){block enemies elsewhere}
                    else if wallresult in blockprisoners then
                      writealloutput(collectivetable,colloutfile,name,60,mbfound){block prisoners}
                    else
                      writealloutput(collectivetable,colloutfile,name,55,mbfound){wall but not blocking}
                end{wall}
                else if (wallresult = 'z') then
                  writealloutput(collectivetable,colloutfile,name,55,mbfound){no wall}
            end{bonus taken}
            else begin
                if bonusmissed then begin
                    if (wallresult <> 'z') then begin
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,61,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,59,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,60,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,56,mbfound){wall but not blocking}
                    end{wall}
                    else if (wallresult = 'z') then
                      writealloutput(collectivetable,colloutfile,name,56,mbfound){no wall}
                end{bonusmissed}
                else begin
                   if (wallresult <> 'z') then begin
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,61,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,59,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,60,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,54,mbfound){wall but not blocking}
                    end{wall}
                    else if (wallresult = 'z') then
                      writealloutput(collectivetable,colloutfile,name,54,mbfound);{no wall}
                end;{no bonusmissed}
            end;{no bonustaken}
        end;{stay unsafe}
    end;{unsafestart}
end;{subject not powered}


if mbfound then begin
     write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
     write(colloutfile,closestEnemy,'  ');
```

```
            write(colloutfile,visibleenemies:3:0);

            write(colloutfile,(danger * (danger + fatality) ):6:2 );
            write(colloutfile,'  ',starttime/100:10:2);
            write(colloutfile, '   ', time2string(starttime/100));
            writeln(colloutfile);
        end;


       if not statesearch.movetostateattime(stoptime_temp) then;
       (* those 2 lines must stay here, no change. *)
       if subjectdone then break;
    end;{while new fast applied}

    eventSearch.Done;
    stateSearch.Done;

  end;{speedmode analysis: whole procedure}

Procedure AnalyzeShieldMode(var colloutfile: Text;var collectivetable:TCollectiveCodeData;
                                fn : String; name: integer);

  var   totalshields                      : T_ccd_value_type;
        eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;
        startTime,levelend,
        stopTime,entertime,collecttime    : TTIme;
        outofhome,entersafety,quitsafety,
        safestart,unsafestart,maladapted,
        collected, safecollect,subjectdone
                                          : Boolean;
        shieldstarttime                   : real;
        x,i,b,position                    : integer;

begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);

    i:= 0; {total of shields}


    (*** find shield application events ***)
    while eventSearch.MoveToNextEvent(apply_shield_snd,anylevel) do begin
        if not stateSearch.MoveToStateAtTime(eventSearch.eventInfos.time) then begin
          WriteLn('Unable to find state');
          Break
        end;{v,rification que dans le fichier mes l',tat isshielded existe au meme moment}

        (*initialization for each new shieldmode found, before analysis*)
        inc(i);
        levelend:= 0; (*necessary for the findendofmode function *)
        safestart:= false;
        unsafestart:= false;
        collected:= false;
        quitsafety:= false;
        entersafety:= false;
        safecollect:= false;
        maladapted:= false;
        entertime:= 0;
        starttime:= eventSearch.eventinfos.time;

        if not statesearch.movetostateattime(starttime)then;
        if not statesearch.movetoposition(statesearch.position - 1) then;
        if (statesearch.stateinfos.mindistJ = -1) and
         (statesearch.stateinfos.mindistH = -1) and
          (statesearch.stateinfos.mindistF = -1) and
           (statesearch.stateinfos.mindistE  > -1) then
            closestEnemy:= 'E'
        else if (statesearch.stateinfos.mindistJ > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistJ <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'J'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistH > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistH <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'H'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistF > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistF <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'F'
            else closestEnemy:= 'E';
        end
        else closestEnemy:= 'Z';

        danger:= statesearch.stateinfos.danger;
        fatality:= statesearch.stateinfos.fatality;
        visibleenemies:= statesearch.stateinfos.nbrofvisibleenemies;

        mbfound:= false;
        stoptime:= findendofmode(statesearch,fn,starttime,isShielded,subjectdone,levelend);
        collected:= OHEventexists(fn,escollectToolsplus,stoptime,stoptime + reactionTime,Collecttime);


        (* check if shieldmode started in a safeplace *)
        if not statesearch.movetostateattime(starttime)then;
        if isSafeplace(statesearch) and notallenemiesinprison(statesearch) then begin
```

```
            safestart:= true;


            (*check if player exits the safeplace*)
            if (checkexitfromstarttostop(statesearch,starttime,stoptime) > 0)then
              quitsafety:= true;


        end
        else if (safestart = false) then begin


        (*check if player enters and stays in safe place*)
            entertime:= checkenterfromstarttostop(statesearch,eventsearch,starttime,stoptime);
            if (entertime > 0) and ((levelend = 0) or (entertime < levelend)) and
             notallenemiesinprison(statesearch) then
                entersafety:= true;
        end;


        if (collected and entersafety) then begin
            if not (checkexitfromstarttostop(statesearch,entertime,collecttime) > 0) then
              safecollect:= true;
        end;


        { maladapt, ?}
        if not statesearch.movetostateattime(starttime) then;
        if not statesearch.movetoposition(statesearch.position - 1) then;
        if ((statesearch.stateinfos.nbrofvisibleenemies = 0) or
            ((notallenemiesinprison(statesearch) = false) and
             (statesearch.stateinfos.onoffstates[isplayerinprison] = false) ) )  then
              maladapted:= true;



        (*compute results of the analysis *)
        if safestart then begin
            if quitsafety then begin
                if collected then
                  writealloutput(collectivetable,colloutfile,name,72,mbfound){collected}
                else
                  writealloutput(collectivetable,colloutfile,name,71,mbfound);{no collect}
            end{quitsafety}
            else if not quitsafety then
              writealloutput(collectivetable,colloutfile,name,67,mbfound);{staysafe}
        end{safestart}

        else if not safestart then begin
            if entersafety then begin
                if (collected and (not safecollect)) or
                  (not collected) then
                   writealloutput(collectivetable,colloutfile,name,69,mbfound)
                   {collected but no more in safe place, or no collect}
                else if safecollect then
                  writealloutput(collectivetable,colloutfile,name,70,mbfound){enter + safecollect}
            end{entersafety}
            else if not entersafety then begin
                if collected then
                  writealloutput(collectivetable,colloutfile,name,68,mbfound){collected}
                else
                  writealloutput(collectivetable,colloutfile,name,67,mbfound);{no collect}
            end;{stay unsafe}
        end;{unsafestart}



        if mbfound then begin
            write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
            write(colloutfile,closestEnemy,'  ');
            write(colloutfile,visibleenemies:3:0);

            write(colloutfile,(danger * (danger + fatality) ):6:2 );
            write(colloutfile,'  ',starttime/100:10:2);
            write(colloutfile, '   ', time2string(starttime/100));
            writeln(colloutfile);
        end;

    end;{while new shield applied}

    eventSearch.Done;
    stateSearch.Done;


  end;{shieldmode analysis: whole procedure}


procedure AnalyzeRisk(var colloutfile: Text;var collectivetable:TCollectiveCodeData;
                         fn : String; name: integer);

  var   eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;
        levelend,
        risktime                          : TTIme;
        subjectdone,safeapply,
        maladapted, prison                : Boolean;
        x,i,b,position                    : integer;
        totaldanger,totalfatality,
        meandanger,meanfatality           : real;
  begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);

    i:= 0; {total of risks}
    totalfatality:= 0;
    totaldanger:= 0;

    (*** find risk application events ***)
    while eventSearch.MoveToNextEvent(apply_teleport_snd,anylevel) do begin
        if not stateSearch.MoveToStateAtTime(eventSearch.eventInfos.time) then begin
          WriteLn('Unable to find state');
          Break
        end;

        (*initialization for each new risk found, before analysis*)
        inc(i);
        safeapply:= false;
        maladapted:= false;
        prison:= false;
        levelend:= 0; (*necessary for the findendofmode function *)
        risktime:= eventSearch.eventinfos.time;

        if not statesearch.movetostateattime(risktime) then;
        if not statesearch.movetoposition(statesearch.position - 1)then;

        if (statesearch.stateinfos.mindistJ = -1) and
         (statesearch.stateinfos.mindistH = -1) and
          (statesearch.stateinfos.mindistF = -1) and
           (statesearch.stateinfos.mindistE  > -1) then
            closestEnemy:= 'E'
        else if (statesearch.stateinfos.mindistJ > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistJ <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'J'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistH > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistH <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'H'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistF > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistF <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'F'
            else closestEnemy:= 'E';
        end
        else closestEnemy:= 'Z';

        danger:= statesearch.stateinfos.danger;
        fatality:= statesearch.stateinfos.fatality;
        visibleenemies:= statesearch.stateinfos.nbrofvisibleenemies;

        mbfound:= false;

        if not statesearch.movetostateattime(risktime) then;
        if not statesearch.movetoposition(statesearch.position - 1)then;
        (* check safeplace when applying the risk *)
        if isSafePlace(statesearch) and notallenemiesinprison(statesearch) then
           safeapply:= true;


        if (statesearch.stateinfos.onoffstates[isPlayerInPrison] = true) then
          prison:= true;

        totaldanger:= totaldanger + (statesearch.stateinfos.danger);
        totalfatality:= totalfatality + (statesearch.stateinfos.fatality);

        (*compute results of the analysis *)
        if (safeapply) or (prison) then
          writealloutput(collectivetable,colloutfile,name,66,mbfound)

        else if (not safeapply) and (not prison) then
          writealloutput(collectivetable,colloutfile,name,65,mbfound);

        if mbfound then begin
            write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
            write(colloutfile,closestEnemy,'  ');
            write(colloutfile,visibleenemies:3:0);

            write(colloutfile,(danger * (danger + fatality) ):6:2 );
            write(colloutfile,'  ',risktime/100:10:2);
            write(colloutfile, '   ', time2string(risktime/100));
            writeln(colloutfile);
        end;

    end;{while new risk applied}

    eventSearch.Done;
    stateSearch.Done;

  end;{riskanalysis: whole procedure}


procedure AnalyzeRepair(var colloutfile: Text;var collectivetable:TCollectiveCodeData;
                            fn : String; name: integer);

  var   totalrisks                        : T_ccd_value_type;
```

```
      eventSearch                       : TEventSearch;
      stateSearch                       : TStateSearch;
      levelend,
      repairtime                        : TTIme;
      subjectdone,safeapply,
      maladapted                        : Boolean;
      x,i,b,position                    : integer;

begin
  eventSearch.Create(fn);
  stateSearch.Create(fn);

  i:= 0; {total of risks}

  (*** find risk application events ***)
  while eventSearch.MoveToNextEvent(apply_repair_snd,anylevel) do begin
      if not stateSearch.MoveToStateAtTime(eventSearch.eventInfos.time) then begin
        WriteLn('Unable to find state');
        Break
      end;

      (*initialization for each new risk found, before analysis*)
      inc(i);
      safeapply:= false;
      maladapted:= false;
      levelend:= 0; (*necessary for the findendofmode function *)
      repairtime:= eventSearch.eventinfos.time;
      if not statesearch.movetostateattime(repairtime) then;

      if not statesearch.movetoposition(statesearch.position - 1) then;
      if (statesearch.stateinfos.mindistJ = -1) and
       (statesearch.stateinfos.mindistH = -1) and
        (statesearch.stateinfos.mindistF = -1) and
         (statesearch.stateinfos.mindistE  > -1) then
          closestEnemy:= 'E'
      else if (statesearch.stateinfos.mindistJ > -1) then begin
          if ((statesearch.stateinfos.mindistE  = -1) or
           (statesearch.stateinfos.mindistJ <= statesearch.stateinfos.mindistE)) then
            closestEnemy:= 'J'
          else closestEnemy:= 'E';
      end
      else if (statesearch.stateinfos.mindistH > -1) then begin
          if ((statesearch.stateinfos.mindistE  = -1) or
           (statesearch.stateinfos.mindistH <= statesearch.stateinfos.mindistE)) then
            closestEnemy:= 'H'
          else closestEnemy:= 'E';
      end
      else if (statesearch.stateinfos.mindistF > -1) then begin
          if ((statesearch.stateinfos.mindistE  = -1) or
           (statesearch.stateinfos.mindistF <= statesearch.stateinfos.mindistE)) then
            closestEnemy:= 'F'
          else closestEnemy:= 'E';
      end
      else closestEnemy:= 'Z';

      danger:= statesearch.stateinfos.danger;
      fatality:= statesearch.stateinfos.fatality;
      visibleenemies:= statesearch.stateinfos.nbrofvisibleenemies;

      mbfound:= false;

      (* check safeplace when applying the risk *)
      if not statesearch.movetostateattime(repairtime) then;
      if isSafePlace(statesearch) and notallenemiesinprison(statesearch) then
        safeapply:= true;

      { check maladapted}
      if not statesearch.movetostateattime(repairtime) then;
      if not statesearch.movetoposition(statesearch.position - 1) then;
      if (statesearch.stateinfos.integrity = 1) then
        maladapted:= true;


      (*compute results of the analysis *)
      if (safeapply) then
        Writealloutput(collectivetable,colloutfile,name,139,mbfound)


      else if (not safeapply)  then
        writealloutput(collectivetable,colloutfile,name,79,mbfound);


      if mbfound then begin
          write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
          write(colloutfile,closestEnemy,'  ');
          write(colloutfile,visibleenemies:3:0);

          write(colloutfile,(danger * (danger + fatality) ):6:2 );
          write(colloutfile,'  ',repairtime/100:10:2);
          write(colloutfile, '   ', time2string(repairtime/100));
          writeln(colloutfile);
      end;


  end;{while new risk applied}

  eventSearch.Done;
  stateSearch.Done;

end;{repairanalysis: whole procedure}
```

```
procedure Analyzeprisonandpower(var colloutfile: Text;var collectivetable:TCollectiveCodeData;
                                fn : String; name: integer);

  var   totalprison                      : T_ccd_value_type;
        eventSearch                      : TEventSearch;
        stateSearch                      : TStateSearch;
        startTime,levelend,
        stopTime,stoptime_temp,powerstart,
        walltime,risktime,victorytime     : TTime;{longint}

        fast,subjectdone,risk,victory,
        arestillvisibleenemies           : Boolean;
        x,i,b,fatalrisk                  : integer;
        cause,wallresult                 : char;

  begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);
    subjectdone:= false;
    cause:= ' ';
    stoptime:= 0; {doit etre utilis, pour les analyses pendant le mode}
    stoptime_temp:= 0;{doit prendre la valeur de la fin du mode meme si il existe un nveau niveau}
    levelend:= 0;{doit prendre une valeur seul. si il y a chgement de nive.}
    i:= 0; {number of powermodes found}

    (*** find powermodes states ***)
    while moveToNextState(statesearch,isPlayerinPrison,subjectdone)  do begin
        (** initialize values for each start of powerstate **)
        levelend:= 0;
        stoptime:=0 ;
        fast:= false;
        walltime:= 0;
        wallresult:= 'z';
        risk:= false;
        victory:= false;
        victorytime:= 0;
        risktime:= 0;
        powerstart:= 0;


        (** find start, stop and levelend values **)
        starttime:= statesearch.stateinfos.time;
        stoptime_temp:= findendofmode(statesearch,fn,starttime,isPlayerinPrison,subjectdone,levelend);

        (** if levelend > 0 the analysis has to stop there, stoptime
        must thus take the value levelend **)
        if levelend > 0 then
          stoptime:= levelend
        else stoptime:= stoptime_temp;

        (* check if player becomes powered when in the prison. if not then = end of analysis.*)
        (* the analysis will be done from the moment th player becomes
           powered and stop when he exits the prison*)
        powerstart:= checkfromstarttostop(statesearch,starttime,stoptime,isPowered);
        if powerstart > 0 then begin
            starttime:= powerstart;
            i:= i + 1;
            (* check how player became powered *)
            cause:= causalsearch(eventsearch,starttime);
            if not statesearch.movetostateattime(starttime) then;

            if not statesearch.movetoposition(statesearch.position - 1) then;
            if (statesearch.stateinfos.mindistJ = -1) and
             (statesearch.stateinfos.mindistH = -1) and
              (statesearch.stateinfos.mindistF = -1) and
               (statesearch.stateinfos.mindistE  > -1) then
                closestEnemy:= 'E'
            else if (statesearch.stateinfos.mindistJ > -1) then begin
                if ((statesearch.stateinfos.mindistE  = -1) or
                 (statesearch.stateinfos.mindistJ <= statesearch.stateinfos.mindistE)) then
                  closestEnemy:= 'J'
                else closestEnemy:= 'E';
            end
            else if (statesearch.stateinfos.mindistH > -1) then begin
                if ((statesearch.stateinfos.mindistE  = -1) or
                 (statesearch.stateinfos.mindistH <= statesearch.stateinfos.mindistE)) then
                  closestEnemy:= 'H'
                else closestEnemy:= 'E';
            end
            else if (statesearch.stateinfos.mindistF > -1) then begin
                if ((statesearch.stateinfos.mindistE  = -1) or
                 (statesearch.stateinfos.mindistF <= statesearch.stateinfos.mindistE)) then
                  closestEnemy:= 'F'
                else closestEnemy:= 'E';
            end
            else closestEnemy:= 'Z';

            danger:= statesearch.stateinfos.danger;
            fatality:= statesearch.stateinfos.fatality;
            visibleenemies:= statesearch.stateinfos.nbrofvisibleenemies;

            mbfound:= false;

            (* check if enemies are still on the maze. *)
            if not statesearch.movetostateattime(starttime) then;
            if (statesearch.stateinfos.nbrofvisibleenemies <= 0) then
              arestillvisibleenemies:= false
            else arestillvisibleenemies := true;

            if arestillvisibleenemies then begin
                (*check if player uses a roller in the same time*)
                if (checkfromstarttostop(statesearch,starttime,stoptime,isFast) >0) then
                  fast:= true;


                (* check if player places a wall and if yes, what for *)
                if oheventexists(fn,[apply_wall_snd],starttime,stoptime,walltime) then begin
                    if not eventsearch.movetofirsteventattime(walltime) then;
                    if not statesearch.movetostateattime(walltime) then;
                    wallresult:= walltest(statesearch,eventsearch);
                end;

                (* check if player uses a risk tool *)
                if oheventexists(fn,[apply_teleport_snd],starttime,stoptime,risktime) then
                    risk:= true;

                (* check if player defeats enemies*)
                if oheventexists(fn,esvictories,starttime,stoptime,victorytime) then
                    victory:= true;


                (*compute results of the analysis *)
                if (cause = 'A') then begin
                    if fast then begin
                        if (wallresult <> 'z') then begin
                            if victory then
                              writealloutput(collectivetable,colloutfile,name,30,mbfound){victory}
                            else if wallresult in blockemptyplace then
                              writealloutput(collectivetable,colloutfile,name,129,mbfound){block place: x}
                            else if wallresult in blockprisoners then
                              writealloutput(collectivetable,colloutfile,name,29,mbfound){block prisoners: z}
                        end{wall}
                        else if (wallresult = 'z') then
                          writealloutput(collectivetable,colloutfile,name,15,mbfound);{victory}
                        {no wall}
                    end{fast}
                    else if not fast then begin
                       if (wallresult <> 'z') then begin
                            if victory then
                              writealloutput(collectivetable,colloutfile,name,20,mbfound){victory}
                            else if wallresult in blockemptyplace then
                              writealloutput(collectivetable,colloutfile,name,128,mbfound){block place: x}
                            else if wallresult in blockprisoners then
                              writealloutput(collectivetable,colloutfile,name,21,mbfound){block prisoners: z}
                        end{wall}
                        else if (wallresult = 'z') then begin
                            if victory then
                              writealloutput(collectivetable,colloutfile,name,9,mbfound){victory}
                        end{no wall}
                    end{not fast}
                end{cause = A}


                else if (cause = 'B') then begin
                    if fast then begin
                        if (wallresult <> 'z') then begin
                            if (wallresult in blockprisoners) and victory then
                              writealloutput(collectivetable,colloutfile,name,50,mbfound){victory}
                            else if (wallresult in blockemptyplace) and (not victory) then
                              writealloutput(collectivetable,colloutfile,name,130,mbfound){block place: x}
                        end{wall}
                        else if (wallresult = 'z') then begin
                            if victory then
                              writealloutput(collectivetable,colloutfile,name,45,mbfound){victory}
                        end{no wall}
                    end{fast}
                    else if not fast then begin
                        if (wallresult <> 'z') then begin
                            if (wallresult in blockprisoners) and victory then
                              writealloutput(collectivetable,colloutfile,name,49,mbfound){victory}
                            else if (wallresult in blockemptyplace) and (not victory) then
                              writealloutput(collectivetable,colloutfile,name,131,mbfound){block place: x}
                        end{wall}
                        else if (wallresult = 'z') and victory then
                              writealloutput(collectivetable,colloutfile,name,39,mbfound);{victory}
                        {no wall}
                    end{not fast}
                end {cause = B}


                else if (cause = 'C') then begin
                    if fast then begin
                        if (wallresult <> 'z') and victory then
                          writealloutput(collectivetable,colloutfile,name,96,mbfound){block prisoners }
                        else if victory then
                          writealloutput(collectivetable,colloutfile,name,90,mbfound){victory + no wall}
                    end{fast}
                    else if not fast then begin
                        if (wallresult <> 'z') and victory then
                          writealloutput(collectivetable,colloutfile,name,95,mbfound){block prisoners}
                        else if victory then
                          writealloutput(collectivetable,colloutfile,name,89,mbfound);{no wall + victory}
                    end;{not fast}
                end;{cause C }


            end;{there arestillvisibleenemies on the maze}
```

```
        if mbfound then begin
            write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
            write(colloutfile,closestEnemy,'  ');
            write(colloutfile,visibleenemies:3:0);

            write(colloutfile,(danger * (danger + fatality) ):6:2 );
            write(colloutfile,'  ',starttime/100:10:2);
            write(colloutfile, '   ', time2string(starttime/100));
            writeln(colloutfile);
        end;

        {  writeln(colloutfile);{this line must stay here}
        end;{if player is powered}
        if not statesearch.movetostateattime(stoptime_temp)then;(* those 2 lines must stay here, no change. *)
        if subjectdone then break;
    end;{while new powermode is found applied in prison}

    eventSearch.Done;
    stateSearch.Done;

  end;{prison+power analysis: whole procedure}


procedure Analyzehome(var colloutfile: Text;var collectivetable:TCollectiveCodeData;
                         fn : String; name: integer);
  const backtogame                        = [go_snd,newlife_snd,die_snd];

  var   totalhomes                        : T_ccd_value_type;
        eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;
        startTime,levelend,stoptimea,
        delta,
        stopTime,entertime,collecttime,
        levelstartdurations,stoptime_temp,
        backhomedurations,quittime        : TTIme;



        collected,notonlyprisoners,
        subjectdone,stay,excluded         : Boolean;
        x,i,b,directenemies               : integer;
        startposition                     : Longint;
        cause                             : TEvent;
begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);

    i:= 0; {total of homes}
    levelstartdurations:= 0;
    backhomedurations:= 0;

    (*** find home application events ***)
    while eventSearch.MoveToNextEvent(home_in_snd,anylevel) do begin
        if not stateSearch.MoveToStateAtTime(eventSearch.eventInfos.time) then begin
          WriteLn('Unable to find state');
          Break
        end;{v,rification que dans le fichier mes l',tat ishome existe au meme moment}

        (*initialization for each new homemode found, before analysis*)
        inc(i);
        levelend:= 0; (*necessary for the findendofmode function *)
        collected:= false;
        stay:= false;
        excluded:= false;
        notonlyprisoners:= false;
        delta:= 0;
        stoptime:= 0;
        stoptime_temp:= 0;
        starttime:= eventSearch.eventinfos.time;
        startposition:= eventsearch.position;



        if not statesearch.movetostateattime(starttime) then;
        if not statesearch.movetoposition(statesearch.position - 1) then;
        if (statesearch.stateinfos.mindistJ = -1) and
         (statesearch.stateinfos.mindistH = -1) and
          (statesearch.stateinfos.mindistF = -1) and
           (statesearch.stateinfos.mindistE  > -1) then
            closestEnemy:= 'E'
        else if (statesearch.stateinfos.mindistJ > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistJ <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'J'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistH > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistH <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'H'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistF > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistF <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'F'
            else closestEnemy:= 'E';
        end
        else closestEnemy:= 'Z';
        danger:= statesearch.stateinfos.danger;
        fatality:= statesearch.stateinfos.fatality;
        visibleenemies:= statesearch.stateinfos.nbrofvisibleenemies;

        mbfound:= false;


        (* check precedents *)
        cause:= checkbefore(eventsearch,startposition,backtogame);
        if cause = go_snd then
          excluded:= true

        else if (cause = die_snd ) or (cause = newlife_snd) then
          excluded:= true;

        if not statesearch.movetostateattime(starttime) then;
        if notallenemiesinprison(statesearch) then
          notonlyprisoners:= true;

        if ((eventsearch.eventinfos.level = 1)
         or (notonlyprisoners = false) ) then begin
            if eventsearch.eventinfos.event = home_out_snd then
               stoptime:= eventsearch.eventinfos.time
            else repeat
                if not eventsearch.movetoposition(eventsearch.position + 1) then;

                if eventsearch.eventinfos.event = home_out_snd then
                   stoptime:= eventsearch.eventinfos.time;
            until stoptime > 0;
            delta:= stoptime - starttime;
        end{level = 1}
        else begin
            if not statesearch.movetostateattime(starttime) then;
            repeat
                if (statesearch.stateinfos.nbrofdirectenemies > 0) then
                  stoptime_temp:= statesearch.stateinfos.time;
                if not statesearch.movetoposition(statesearch.position + 1) then
                  stoptime_temp:= statesearch.stateinfos.time
            until stoptime_temp > 0;
            if not eventsearch.movetofirsteventatoraftertime(starttime) then;
            repeat
                if eventsearch.eventinfos.event = food_snd then
                  stoptime:= eventsearch.eventinfos.time;
                if not eventsearch.movetoposition(eventsearch.position + 1) then
                  stoptime:= eventsearch.eventinfos.time;
            until stoptime > 0;

            if stoptime_temp < stoptime then
              stoptime:= stoptime_temp;
            delta:= stoptime - starttime;
        end;


        (* check duration of safe stay *)
        if (delta > minhometime) then
            stay:= true;

        if cause = go_snd then
          levelstartdurations:= levelstartdurations + delta;

        if stay then
           (* check if collect object *)
           collected:= Eventexists(fn,escollectToolsplus,starttime,stoptime);

        (*compute results of the analysis *)

        if stay and ( not excluded ) and (not collected ) and notonlyprisoners then
          begin
              writealloutput(collectivetable,colloutfile,name,97,mbfound);
              backhomedurations:= backhomedurations + delta;
          end;

        if mbfound then begin
            write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
            write(colloutfile,closestEnemy,'  ');
            write(colloutfile,visibleenemies:3:0);

            write(colloutfile,(danger * (danger + fatality) ):6:2 );
            write(colloutfile,'  ',starttime/100:10:2);
            write(colloutfile, '   ', time2string(starttime/100));
            writeln(colloutfile);
        end;

    end;{while new home applied}

    eventSearch.Done;
    stateSearch.Done;

  end;{homemode analysis: whole procedure}


procedure Analyzehourglass(var colloutfile: Text;var collectivetable:TCollectiveCodeData;
                               fn : String; name: integer);

  const specificreactionTime              = Round(0.08 * 100); { 0.5 seconds }

  var   totalHglas                        : T_ccd_value_type;
        eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;
```

```
      levelend,
      pausestart, pausestop             : TTIme;
      subjectdone,select,applytool,
      maladapted                        : Boolean;
      x,i,b,position                    : integer;

begin
  eventSearch.Create(fn);
  stateSearch.Create(fn);

  i:= 0; {total of hourglasses}


  (*** find hourglass application events ***)
  while eventSearch.MoveToNextEvent(apply_pause_snd,anylevel) do begin
      if not stateSearch.MoveToStateAtTime(eventSearch.eventInfos.time) then begin
        WriteLn('Unable to find state');
        Break
      end;
      pausestart:= eventSearch.eventinfos.time;


      if not statesearch.movetoposition(statesearch.position - 1) then;
      if (statesearch.stateinfos.mindistJ = -1) and
       (statesearch.stateinfos.mindistH = -1) and
        (statesearch.stateinfos.mindistF = -1) and
         (statesearch.stateinfos.mindistE  > -1) then
          closestEnemy:= 'E'
      else if (statesearch.stateinfos.mindistJ > -1) then begin
          if ((statesearch.stateinfos.mindistE  = -1) or
           (statesearch.stateinfos.mindistJ <= statesearch.stateinfos.mindistE)) then
            closestEnemy:= 'J'
          else closestEnemy:= 'E';
      end
      else if (statesearch.stateinfos.mindistH > -1) then begin
          if ((statesearch.stateinfos.mindistE  = -1) or
           (statesearch.stateinfos.mindistH <= statesearch.stateinfos.mindistE)) then
            closestEnemy:= 'H'
          else closestEnemy:= 'E';
      end
      else if (statesearch.stateinfos.mindistF > -1) then begin
          if ((statesearch.stateinfos.mindistE  = -1) or
           (statesearch.stateinfos.mindistF <= statesearch.stateinfos.mindistE)) then
            closestEnemy:= 'F'
          else closestEnemy:= 'E';
      end
      else closestEnemy:= 'Z';

      danger:= statesearch.stateinfos.danger;
      fatality:= statesearch.stateinfos.fatality;
      visibleenemies:= statesearch.stateinfos.nbrofvisibleenemies;

      mbfound:= false;

      (*initialization for each new risk found, before analysis*)
      inc(i);
      applytool:= false;
      select:= false;
      maladapted:= false;
      levelend:= 0; (*necessary for the findendofmode function *)
      pausestop:= 0;

      if not eventsearch.movetonextevent(end_pause_snd,anylevel) then;
      pausestop:= eventsearch.eventinfos.time;

      if eventexists(fn,[toolselect_snd],pausestart,pausestop) then
        select:= true;

      if eventexists(fn,esApplyTools,pausestop,pausestop +specificreactiontime) then
        applytool:= true;

      (*compute results of the analysis *)
      if select then begin
          if applytool then
            writealloutput(collectivetable,colloutfile,name,75,mbfound){apply}
          else
            writealloutput(collectivetable,colloutfile,name,74,mbfound);{no apply}
      end{select}
      else
        writealloutput(collectivetable,colloutfile,name,73,mbfound);{no select}

      if mbfound then begin
          write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
          write(colloutfile,closestEnemy,'  ');
          write(colloutfile,visibleenemies:3:0);

          write(colloutfile,(danger * (danger + fatality) ):6:2 );
          write(colloutfile,'  ',pausestart/100:10:2);
          write(colloutfile, '   ', time2string(pausestart/100));
          writeln(colloutfile);
      end;
  end;{while new pause applied}

  eventSearch.Done;
  stateSearch.Done;

end;{hourglassanalysis: whole procedure}
```

```
procedure Analyzebonus(var colloutfile: Text;var collectivetable:TCollectiveCodeData;
                           fn : String; name: integer);

  var   totalbonus                        : T_ccd_value_type;
        eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;
        startTime,levelend,
        stopTime,stoptime_temp,bonustime,
        speedtime, powertime,entertime,
        pursuitstoptime,hurttime,
        killtime                          : TTIme;
        bonusmissed,entersafety,
        collectbonus, speed,power,
        playerhurt,playerkilled,
        subjectdone,lastbonusdistance     : Boolean;
        x,i,b,position                    : integer;

begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);
    subjectdone:= false;
    i:= 0; {total of visiblebonuses}


    (*** find bonus appearing events ***)
    while eventSearch.MoveToNextEvent(bonus_app_snd,anylevel) do begin
        starttime:= eventsearch.eventinfos.time;
        if not stateSearch.MoveToStateAtTime(starttime) then begin
          WriteLn('Unable to find state');
          Break
        end;{v,rification que dans le fichier mes l',tat existe au meme moment}

        (*initialization for each new bonus found, before analysis*)
        inc(i);
        levelend:= 0; (*necessary for the findendofmode function *)
        stoptime_temp:= 0;
        collectbonus:= false;
        speed:= false;
        power:= false;
        powertime:= 0;
        speedtime:= 0;
        hurttime:= 0;
        entertime:= 0;
        killtime:= 0;
        playerhurt:= false;
        playerkilled:= false;
        entersafety:= false;
        lastbonusdistance:= false;
        bonusmissed:= false;
        bonustime:= 0;
        mbfound:= false;

        if not statesearch.movetoposition(statesearch.position - 1) then;
        if (statesearch.stateinfos.mindistJ = -1) and
         (statesearch.stateinfos.mindistH = -1) and
          (statesearch.stateinfos.mindistF = -1) and
           (statesearch.stateinfos.mindistE  > -1) then
            closestEnemy:= 'E'
        else if (statesearch.stateinfos.mindistJ > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistJ <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'J'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistH > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistH <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'H'
            else closestEnemy:= 'E';
        end
        else if (statesearch.stateinfos.mindistF > -1) then begin
            if ((statesearch.stateinfos.mindistE  = -1) or
             (statesearch.stateinfos.mindistF <= statesearch.stateinfos.mindistE)) then
              closestEnemy:= 'F'
            else closestEnemy:= 'E';
        end
        else closestEnemy:= 'Z';

        danger:= statesearch.stateinfos.danger;
        fatality:= statesearch.stateinfos.fatality;
        visibleenemies:= statesearch.stateinfos.nbrofvisibleenemies;

        stoptime_temp:= findendofmode(statesearch,fn,starttime,isBonusVisible,subjectdone,levelend);

        (** if levelend > 0 the analysis has to stop there, stoptime
        must thus take the value levelend **)
        if levelend > 0 then
          stoptime:= levelend
        else stoptime:= stoptime_temp;

        if ohEventexists(fn,[bonus_get_snd],starttime,stoptime + reactiontime,bonustime) then
          collectbonus:= true;

        (* check if player is hurt *)
        if oheventexists(fn,esplayerhurt,starttime,stoptime,hurttime) then
            playerhurt:= true;

        (* check if player dies *)
        if oheventexists(fn,[die_snd],starttime,stoptime,killtime) then
          playerkilled:= true;
         (*check if player becomes powered at the same time*)
        powertime:= checkfromstarttostop(statesearch,starttime,bonustime,ispowered);
        if (powertime > starttime ) then begin
            power:= true;
            if not statesearch.movetostateattime(starttime)then;
        end;

         (*check if player becomes fast at the same time*)
        speedtime:= checkfromstarttostop(statesearch,starttime,bonustime,isfast);
        if (speedtime > starttime ) then begin
            speed:= true;
            if not statesearch.movetostateattime(starttime)then;
        end;

        {check if player enters safe place and stays > minhometime}
        if not statesearch.movetostateattime(starttime) then;
        begin
            while ((statesearch.stateinfos.time < stoptime) and (not entersafety)) do begin
                if not statesearch.movetoposition(statesearch.position + 1) then;

                if (isSafeplacespecial(statesearch) and
                 notallenemiesinprison(statesearch) and
                  (computestateduration(statesearch,isSafeplacespecial) >= minhometime) ) then
                    entersafety:= true;
            end;
            if not statesearch.movetostateattime(starttime) then;
        end;


        if not collectbonus then begin
            pursuitstoptime:= stoptime;
            if (playerhurt and (hurttime < pursuitstoptime)) then begin
                pursuitstoptime:= hurttime;
                if not statesearch.movetostateattime(hurttime)then;
                if not statesearch.movetoposition(statesearch.position - 1) then;
                if ((statesearch.stateinfos.bonusdistance < 5) and
                 (statesearch.stateinfos.bonusdistance > -1) ) then
                  lastbonusdistance:= true;
            end{playerhurt}
            else if playerkilled  then begin
                pursuitstoptime:= killtime;
                if not statesearch.movetostateattime(killtime)then;
                if not statesearch.movetoposition(statesearch.position - 1) then;
                if ((statesearch.stateinfos.bonusdistance < 5) and
                 (statesearch.stateinfos.bonusdistance > -1)  ) then begin
                    if not statesearch.movetostateattime(stoptime)then;
                    if not statesearch.movetoposition(statesearch.position - 1) then;
                    if not (statesearch.stateinfos.bonusdistance > 10) then
                     lastbonusdistance:= true;
                end
                else lastbonusdistance:= false;
            end{playerkilled}
            else begin
                if not statesearch.movetostateattime(pursuitstoptime)then;
                if not statesearch.movetoposition(statesearch.position - 1) then;
                if((statesearch.stateinfos.bonusdistance < 5) and
                 (statesearch.stateinfos.bonusdistance > -1) ) then
                  lastbonusdistance:= true;
            end;{not hurt, not killed}
            if (pursuitbonus(statesearch,'bonus',starttime,pursuitstoptime)
             and lastbonusdistance) then
               bonusmissed:= true;
        end;{no collect bonus}


         (*compute results of the analysis *)
       { if (collectbonus = true) then
          writealloutput(collectivetable,colloutfile,name,98,mbfound)
       }

        if( (not collectbonus) and (bonusmissed) and (not entersafety)) then
          writealloutput(collectivetable,colloutfile,name,99,mbfound);

        if mbfound then begin
            write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
            write(colloutfile,closestEnemy,'  ');
            write(colloutfile,visibleenemies:3:0);

            write(colloutfile,(danger * (danger + fatality) ):6:2 );
            write(colloutfile,'  ',starttime/100:10:2);
            write(colloutfile, '   ', time2string(starttime/100));
            writeln(colloutfile);
        end;

        if not statesearch.movetostateattime(stoptime_temp) then;
         { writeln(' problem at nwpower: line 659.46');}(* those 2 lines must stay here, no change. *)
        if subjectdone then break;
    end;{while new bonus visible applied}

    eventSearch.Done;
    stateSearch.Done;

  end;{bonus analysis: whole procedure}


procedure AnalyzepowerMode(var colloutfile: Text; var collectivetable:TCollectiveCodeData;
                           fn : String; name: integer);

  var   totalpowermodes                   : T_ccd_value_type;
        eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;
        startTime,levelend,
        stopTime,stoptime_temp,killtime,
        hurttime,bonustime,pursuitstoptime,
        entertime,collecttime,quittime,
        walltime,risktime,victorytime,
        victoryEtime,victoryFtime,first,
        endofbonus,pursuitbonusstoptime   : TTime;{longint}
        sequenceEnd,zoneChanged,
        safestart,collect, prison,fast,
        subjectdone,quitsafety,pursuit,
        entersafety,wait,risk,victory,
        playerhurt,playerkilled,
        toolcollect,bonustaken ,
        bonusvisible,missbonus,
        safecollect,faststart,
        pursuit_F,pursuit_E,victoryE,victoryF,
        lastdistanceE,lastdistanceF,
        lastbonusdistance,maladapted,
        Eneverthere,Fneverthere           : Boolean;
        x,i,b,position,fatalrisk          : integer;
        cause,wallresult                  : char;
        powerstartposition,
        powerstopposition                 : longint;


        procedure computecausea;

        begin
            (* safestart *)
                if safestart then begin
                    if quitsafety then begin
                        if fast then begin
                            if victory then begin
                                if (wallresult = 'z') or ((wallresult <> 'z') and (victorytime < walltime)) then
                                  writealloutput(collectivetable,colloutfile,name,12,mbfound){victory before}
                                else if ((wallresult <> 'z') and (victorytime >= walltime)) then begin
                                    if wallresult in blockemptyplace then
                                      writealloutput(collectivetable,colloutfile,name,116,mbfound){block empty place}
                                    else if wallresult in blockenemieselsewhere then
                                      writealloutput(collectivetable,colloutfile,name,115,mbfound){block enemies elsewhere}
                                    else if wallresult in blockprisoners then
                                      writealloutput(collectivetable,colloutfile,name,117,mbfound){block prisoners}
                                    else
                                      writealloutput(collectivetable,colloutfile,name,12,mbfound);
                                      {wall exists but not blocking}
                                end;{victory after}
                            end{victory}

                            else if not victory then begin
                                if pursuit then begin
                                    if (wallresult <> 'z') then begin
                                        if wallresult in blockemptyplace then
                                          writealloutput(collectivetable,colloutfile,name,32,mbfound){block empty place}
                                        else if wallresult in blockenemieselsewhere then
                                          writealloutput(collectivetable,colloutfile,name,31,mbfound){block enemies
elsewhere}

                                        else if wallresult in blockprisoners then
                                          writealloutput(collectivetable,colloutfile,name,33,mbfound){block prisoners}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,13,mbfound);{other wall}
                                    end{wall}
                                    else if (wallresult = 'z') then
                                      writealloutput(collectivetable,colloutfile,name,13,mbfound);{no wall}
                                end{pursuit}
                                else if not pursuit then begin
                                    if (wallresult <> 'z') then begin
                                        if wallresult in blockemptyplace then
                                          writealloutput(collectivetable,colloutfile,name,32,mbfound){block empty place}
                                        else if wallresult in blockenemieselsewhere then
                                          writealloutput(collectivetable,colloutfile,name,31,mbfound){block enemies
elsewhere}

                                        else if wallresult in blockprisoners then
                                          writealloutput(collectivetable,colloutfile,name,33,mbfound){block prisoners}
                                        else if risk then
                                          writealloutput(collectivetable,colloutfile,name,17,mbfound){risk}
                                        else if collect then
                                          writealloutput(collectivetable,colloutfile,name,14,mbfound){collect object or
bonus}

                                        else if missbonus and (not bonustaken)then
                                          writealloutput(collectivetable,colloutfile,name,103,mbfound){missbonus}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,7,mbfound);
                                          {simply power to get out of home with speed}
                                    end{wall}
                                    else if (wallresult = 'z') then begin
                                        if risk then
                                          writealloutput(collectivetable,colloutfile,name,17,mbfound){risk}
                                        else if collect then
                                          writealloutput(collectivetable,colloutfile,name,14,mbfound){collect object or
bonus}

                                        else if missbonus and (not bonustaken)then
                                          writealloutput(collectivetable,colloutfile,name,103,mbfound){missbonus}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,7,mbfound);
                                          {simply power to get out of home with speed}
                                    end;{no wall}
                                end;{no pursuit}
                            end;{no victory}
                        end{speed}
                        else if not fast then begin
                            if victory then begin
                                if (wallresult = 'z') or ((wallresult <> 'z') and (victorytime < walltime)) then
                                  writealloutput(collectivetable,colloutfile,name,5,mbfound){victory before}
                                else if ((wallresult <> 'z') and (victorytime >= walltime)) then begin
                                    if wallresult in blockemptyplace then
                                      writealloutput(collectivetable,colloutfile,name,110,mbfound){block empty place}
                                    else if wallresult in blockenemieselsewhere then
                                      writealloutput(collectivetable,colloutfile,name,109,mbfound){block enemies elsewhere}
                                    else if wallresult in blockprisoners then
                                      writealloutput(collectivetable,colloutfile,name,111,mbfound){block prisoners}
                                    else
                                      writealloutput(collectivetable,colloutfile,name,5,mbfound);{wall exists but not
blocking}
                                end;{victory after}
                            end{victory}
                            else if not victory then begin
                                if pursuit then begin
                                    if (wallresult <> 'z') then begin
                                        if wallresult in blockemptyplace then
                                          writealloutput(collectivetable,colloutfile,name,24,mbfound){block empty place}
                                        else if wallresult in blockenemieselsewhere then
                                          writealloutput(collectivetable,colloutfile,name,23,mbfound){block enemies
elsewhere}

                                        else if wallresult in blockprisoners then
                                          writealloutput(collectivetable,colloutfile,name,25,mbfound){block prisoners}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,6,mbfound);{other wall}
                                    end{wall}
                                    else if (wallresult = 'z') then
                                      writealloutput(collectivetable,colloutfile,name,6,mbfound);{no wall}
                                end{pursuit}

                                else if not pursuit then begin
                                    if (wallresult <> 'z') then begin
                                        if wallresult in blockemptyplace then
                                          writealloutput(collectivetable,colloutfile,name,24,mbfound){block empty place}
                                        else if wallresult in blockenemieselsewhere then
                                          writealloutput(collectivetable,colloutfile,name,23,mbfound){block enemies
elsewhere}

                                        else if wallresult in blockprisoners then
                                          writealloutput(collectivetable,colloutfile,name,25,mbfound){block prisoners}
                                        else if risk then
                                          writealloutput(collectivetable,colloutfile,name,17,mbfound){risk}
                                        else if collect then
                                          writealloutput(collectivetable,colloutfile,name,8,mbfound){collect object or bonus}
                                        else if missbonus and (not bonustaken) then
                                          writealloutput(collectivetable,colloutfile,name,102,mbfound){missbonus}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,7,mbfound);
                                          {simply power to get out of home with speed}
                                    end{wall}
                                    else if (wallresult = 'z') then begin
                                        if risk then
                                          writealloutput(collectivetable,colloutfile,name,17,mbfound){risk}
                                        else if collect then
                                          writealloutput(collectivetable,colloutfile,name,8,mbfound){collect object or bonus}
                                        else if missbonus and (not bonustaken) then
                                          writealloutput(collectivetable,colloutfile,name,102,mbfound){missbonus}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,7,mbfound);
                                          {simply power to get out of home with speed}
                                    end;{no wall}
                                end;{no pursuit}
                            end;{no victory}
                        end;{no speed}

                    end{safestart and quitsafety}
                    else if not quitsafety then
                      writealloutput(collectivetable,colloutfile,name,3,mbfound);{safestart and not quitsafety}
                end{safestart}

                (* unsafestart *)
                else if not safestart then begin
                    if entersafety then begin
                        if victory then begin
                            if (wallresult <> 'z') then begin
                                if (victorytime < walltime) and (victorytime < entertime) then
                                  writealloutput(collectivetable,colloutfile,name,1,mbfound){victory first}
                                else if (entertime < walltime) and (entertime < victorytime) then begin
                                    if safecollect then
                                      writealloutput(collectivetable,colloutfile,name,3,mbfound){safecollect}
                                    else if ((killtime = 0) or (killtime > entertime)) then
                                      writealloutput(collectivetable,colloutfile,name,4,mbfound)
                                       {no collect and no death or death after}
                                    else
                                      writealloutput(collectivetable,colloutfile,name,3,mbfound)
                                       {no collect but death ==> enter because death}
                                end{enterhome first}
                                else if (walltime < victorytime) and (walltime <= entertime) then begin
                                    if wallresult in blockemptyplace then
                                      writealloutput(collectivetable,colloutfile,name,19,mbfound){block empty place}
                                    else if wallresult in blockenemieselsewhere then
                                      writealloutput(collectivetable,colloutfile,name,18,mbfound){block enemies elsewhere}
                                    else if wallresult in blockprisoners then
                                      writealloutput(collectivetable,colloutfile,name,20,mbfound){block prisoners}
                                    else begin
                                        if safecollect then
                                          writealloutput(collectivetable,colloutfile,name,3,mbfound){safecollect}
                                        else if ((killtime = 0) or (killtime > entertime)) then
                                          writealloutput(collectivetable,colloutfile,name,4,mbfound)
                                           {no collect and no death or death after}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,3,mbfound)
                                           {no collect but death ==> enter because death}
                                    end{wall but no block}
                                end{wall first}

                            end{wall}
                            else if (wallresult = 'z') then begin
                                if (victorytime < entertime) then
                                  writealloutput(collectivetable,colloutfile,name,1,mbfound){victory before enterhome}
                                else if safecollect then
                                  writealloutput(collectivetable,colloutfile,name,3,mbfound){safecollect}
                                else if ((killtime = 0) or (killtime > entertime)) then
                                  writealloutput(collectivetable,colloutfile,name,4,mbfound)
                                   {no collect and no death or death after}
                                else
                                  writealloutput(collectivetable,colloutfile,name,3,mbfound)
                                   {no collect but death ==> enter because death}
                            end{}
                        end{victory}
                        else if not victory then begin
                            if (wallresult <> 'z') then begin
                                if (walltime <= entertime) then begin
                                    if wallresult in blockemptyplace then
                                      writealloutput(collectivetable,colloutfile,name,19,mbfound){block empty place}
                                    else if wallresult in blockenemieselsewhere then
                                      writealloutput(collectivetable,colloutfile,name,18,mbfound){block enemies elsewhere}
                                    else if wallresult in blockprisoners then
                                      writealloutput(collectivetable,colloutfile,name,20,mbfound){block prisoners}
                                    else if wallresult = 'F' then
                                      writealloutput(collectivetable,colloutfile,name,3,mbfound)
                                    else if safecollect then
                                      writealloutput(collectivetable,colloutfile,name,3,mbfound){safecollect}
                                    else if ((killtime = 0) or (killtime > entertime)) then
                                      writealloutput(collectivetable,colloutfile,name,4,mbfound)
                                       {no collect and no death or death after}
                                    else
                                      writealloutput(collectivetable,colloutfile,name,3,mbfound)
                                       {no collect but death ==> enter because death}
                                end{wall first}
                                else if (entertime < walltime) then begin
                                    if safecollect then
                                      writealloutput(collectivetable,colloutfile,name,3,mbfound){safecollect}
                                    else if ((killtime = 0) or (killtime > entertime)) then
                                      writealloutput(collectivetable,colloutfile,name,4,mbfound)
                                       {no collect and no death or death after}
                                    else
                                      writealloutput(collectivetable,colloutfile,name,3,mbfound)
                                       {no collect but death ==> enter because death}
                                end{enterhome first}
                            end{wall}
                            else if (wallresult = 'z') then begin
                                if safecollect then
                                  writealloutput(collectivetable,colloutfile,name,3,mbfound){safecollect}
                                else if ((killtime = 0) or (killtime > entertime)) then
                                  writealloutput(collectivetable,colloutfile,name,4,mbfound)
                                   {no collect and no death or death after}
                                else if (not safecollect ) then
                                  writealloutput(collectivetable,colloutfile,name,3,mbfound)
                                   {no collect but death ==> enter because death}
                            end;{no wall}
                        end{no victory}
                    end{entersafety}
                    else if not entersafety then begin
                        if fast then begin
                            if victory then begin
                                if (wallresult <> 'z') then begin
                                    if (walltime < victorytime) then begin
                                        if wallresult in blockemptyplace then
                                          writealloutput(collectivetable,colloutfile,name,113,mbfound){block empty place}
                                        else if wallresult in blockenemieselsewhere then
                                          writealloutput(collectivetable,colloutfile,name,112,mbfound){block enemies
elsewhere}
                                        else if wallresult in blockprisoners then
                                          writealloutput(collectivetable,colloutfile,name,114,mbfound){block prisoners}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,10,mbfound)
                                    end{wall first}
                                    else if (victorytime < walltime) then
                                      writealloutput(collectivetable,colloutfile,name,10,mbfound){victory first}
                                end{wall}
                                else if (wallresult = 'z') then
                                  writealloutput(collectivetable,colloutfile,name,10,mbfound){no wall}
                            end{victory}
                            else if not victory then begin
                                if pursuit then begin
                                    if (wallresult <> 'z') then begin
                                         if wallresult in blockemptyplace then
                                           writealloutput(collectivetable,colloutfile,name,27,mbfound){block empty place}
                                         else if wallresult in blockenemieselsewhere then
                                           writealloutput(collectivetable,colloutfile,name,26,mbfound){block enemies
elsewhere}
                                         else if wallresult in blockprisoners then
                                           writealloutput(collectivetable,colloutfile,name,28,mbfound){block prisoners}
                                         else
                                           writealloutput(collectivetable,colloutfile,name,11,mbfound){wall not blocking}
                                    end{wall}
                                    else if (wallresult = 'z') then
                                      writealloutput(collectivetable,colloutfile,name,11,mbfound){no wall}
                                end{pursuit}
                                else if not pursuit then begin
                                    if (wallresult <> 'z') then begin
                                        if wallresult in blockemptyplace then
                                          writealloutput(collectivetable,colloutfile,name,27,mbfound){block empty place}
                                        else if wallresult in blockenemieselsewhere then
                                          writealloutput(collectivetable,colloutfile,name,26,mbfound){block enemies
elsewhere}

                                        else if wallresult in blockprisoners then
                                          writealloutput(collectivetable,colloutfile,name,28,mbfound){block prisoners}
                                        else if risk then
                                          writealloutput(collectivetable,colloutfile,name,16,mbfound){risk}
                                        else if bonustaken then
                                          writealloutput(collectivetable,colloutfile,name,101,mbfound){bonustaken}
                                        else if missbonus and (not bonustaken)then
                                          writealloutput(collectivetable,colloutfile,name,103,mbfound){missbonus}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,3,mbfound){nothing more}
                                    end{wall}
                                    else if (wallresult = 'z') then begin
                                        if risk then
                                          writealloutput(collectivetable,colloutfile,name,16,mbfound){risk}
                                        else if bonustaken then
                                          writealloutput(collectivetable,colloutfile,name,101,mbfound){bonustaken}
                                        else if missbonus and (not bonustaken) then
                                          writealloutput(collectivetable,colloutfile,name,103,mbfound){missbonus}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,3,mbfound);{nothing more}
                                    end;{no wall}
                                end;{no pursuit}
                            end;{no victory}
                        end{speed}
                        else if not fast then begin
                            if victory then begin
                                if (wallresult <> 'z') then begin
                                    if (walltime < victorytime) then begin
                                        if wallresult in blockemptyplace then
                                          writealloutput(collectivetable,colloutfile,name,107,mbfound){block empty place}
                                        else if wallresult in blockenemieselsewhere then
                                          writealloutput(collectivetable,colloutfile,name,106,mbfound){block enemies
elsewhere}

                                        else if wallresult in blockprisoners then
                                          writealloutput(collectivetable,colloutfile,name,108,mbfound){block prisoners}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,1,mbfound)
                                    end{wall first}
                                    else if (victorytime < walltime) then
                                      writealloutput(collectivetable,colloutfile,name,1,mbfound){victory first}
                                end{wall}
                                else if (wallresult = 'z') then
                                  writealloutput(collectivetable,colloutfile,name,1,mbfound){no wall}
                            end{victory}
                            else if not victory then begin
                                if pursuit then begin
                                    if (wallresult <> 'z') then begin
                                        if wallresult in blockemptyplace then
                                          writealloutput(collectivetable,colloutfile,name,19,mbfound){block empty place}
                                         else if wallresult in blockenemieselsewhere then
                                           writealloutput(collectivetable,colloutfile,name,18,mbfound){block enemies
elsewhere}

                                         else if wallresult in blockprisoners then
                                           writealloutput(collectivetable,colloutfile,name,20,mbfound){block prisoners}
                                         else
                                           writealloutput(collectivetable,colloutfile,name,2,mbfound){wall not blocking}
                                    end{wall}
                                    else if (wallresult = 'z') then
                                      writealloutput(collectivetable,colloutfile,name,2,mbfound){no wall}
                                end{pursuit}
                                else if not pursuit then begin
                                    if (wallresult <> 'z') then begin
                                        if wallresult in blockemptyplace then
                                          writealloutput(collectivetable,colloutfile,name,19,mbfound){block empty place}
                                        else if wallresult in blockenemieselsewhere then
                                          writealloutput(collectivetable,colloutfile,name,18,mbfound){block enemies
elsewhere}

                                        else if wallresult in blockprisoners then
                                          writealloutput(collectivetable,colloutfile,name,20,mbfound){block prisoners}
                                        else if risk then
                                          writealloutput(collectivetable,colloutfile,name,16,mbfound){risk}
                                        else if bonustaken then
                                          writealloutput(collectivetable,colloutfile,name,100,mbfound){bonustaken}
                                        else if missbonus and (not bonustaken) then
                                          writealloutput(collectivetable,colloutfile,name,102,mbfound){missbonus}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,3,mbfound){nothing more}
                                    end{wall}
                                    else if (wallresult = 'z') then begin
                                        if risk then
                                          writealloutput(collectivetable,colloutfile,name,16,mbfound){risk}
                                        else if bonustaken then
                                          writealloutput(collectivetable,colloutfile,name,100,mbfound){bonustaken}
                                        else if missbonus and (not bonustaken) then
                                          writealloutput(collectivetable,colloutfile,name,102,mbfound){missbonus}
                                        else
                                          writealloutput(collectivetable,colloutfile,name,3,mbfound);{nothing more}
                                    end;{no wall}
                                end;{no pursuit}
                            end;{no victory}
```

```
                      end;{no speed}
                  end;{not entersafety}
              end;{unsafestart }
             { if maladapted then begin
                 IncCollectivecode(collectiveTable,135);
                 write(colloutfile,' useless ==> + 135');
              end;{maladapted use of fist}

      end;{computecauseA}

procedure computecauseb;
begin
    if not wait then begin
        if fast then begin
            if victory then
              writealloutput(collectivetable,colloutfile,name,40,mbfound){victory}
            else if pursuit then
              writealloutput(collectivetable,colloutfile,name,41,mbfound){pursuit}
        end{fast}
        else if (not fast) and victory then
          writealloutput(collectivetable,colloutfile,name,34,mbfound){not fast + victory}
    end{did not wait}
    else if wait then begin
        if fast then begin
            if victory then begin
                if (wallresult <> 'z') then begin
                    if (victorytime < walltime) then
                      writealloutput(collectivetable,colloutfile,name,42,mbfound){victory before}
                    else if (walltime < victorytime) then begin
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,123,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,121,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,122,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,42,mbfound){wall not blocking}
                    end{wall before}
                end{wall}
                else if (wallresult = 'z') then
                  writealloutput(collectivetable,colloutfile,name,42,mbfound){no wall}
            end{victory}
            else if not victory then begin
                if pursuit then begin
                    if (wallresult <> 'z') then begin
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,53,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,51,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,52,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,53,mbfound){wall not blocking}
                    end{wall}
                    else if (wallresult = 'z') then
                      writealloutput(collectivetable,colloutfile,name,43,mbfound){no wall}
                end{pursuit}
                else if not pursuit then begin
                    if (wallresult <> 'z') then begin
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,53,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,51,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,52,mbfound){block prisoners}
                        else if collect then
                          writealloutput(collectivetable,colloutfile,name,44,mbfound){collect}
                        else if not collect then
                          writealloutput(collectivetable,colloutfile,name,37,mbfound){no collect}
                    end{wall}
                    else if (wallresult = 'z') then begin
                        if collect then
                          writealloutput(collectivetable,colloutfile,name,44,mbfound){collect}
                        else if not collect then
                          writealloutput(collectivetable,colloutfile,name,37,mbfound){no collect}
                    end{no wall}
                end{no pursuit}
            end{not victory}
        end{fast}
        else if not fast then begin
            if victory then begin
                if (wallresult <> 'z') then begin
                    if (victorytime < walltime) then
                      writealloutput(collectivetable,colloutfile,name,35,mbfound){victory before}
                    else if (walltime < victorytime) then begin
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,119,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,120,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,118,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,35,mbfound){wall not blocking}
                    end{wall before}
                end{wall}
                else if (wallresult = 'z') then
                  writealloutput(collectivetable,colloutfile,name,35,mbfound){no wall}
            end{victory}
            else if not victory then begin
                if pursuit then begin
```

```
                    if (wallresult <> 'z') then begin
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,47,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,48,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,46,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,36,mbfound){wall not blocking}
                    end{wall}
                    else if (wallresult = 'z') then
                      writealloutput(collectivetable,colloutfile,name,36,mbfound){no wall}
                end{pursuit}
                else if not pursuit then begin
                    if (wallresult <> 'z') then begin
                        if wallresult in blockemptyplace then
                          writealloutput(collectivetable,colloutfile,name,47,mbfound){block empty place}
                        else if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,48,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,46,mbfound){block prisoners}
                        else if collect and (not safecollect) then
                          writealloutput(collectivetable,colloutfile,name,38,mbfound){collect}
                        else if not collect then
                          writealloutput(collectivetable,colloutfile,name,37,mbfound){no collect}
                    end{wall}
                    else if (wallresult = 'z') then begin
                        if collect and (not safecollect) then
                          writealloutput(collectivetable,colloutfile,name,38,mbfound){collect}
                        else if not collect then
                          writealloutput(collectivetable,colloutfile,name,37,mbfound){no collect}
                    end{no wall}
                end{no pursuit}
            end{not victory}
        end{not fast}
    end;{wait for janus}
 {   if maladapted then begin
         IncCollectivecode(collectiveTable,135);
         write(colloutfile,' useless ==> + 135');
    end;{maladapted use of fist}

end; {computecauseb}

procedure computecauseC;
    begin
        if fast then begin
            if victory then begin
                if (wallresult <> 'z') then begin
                    if (victorytime < walltime) then
                      writealloutput(collectivetable,colloutfile,name,86,mbfound){victory first}
                    else if (walltime < victorytime) then begin
                        if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,126,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,127,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,86,mbfound){wall not blocking}
                    end{wall first}
                end{wall}
                else if (wallresult = 'z') then
                  writealloutput(collectivetable,colloutfile,name,86,mbfound)
            end{victory}
            else if not victory then begin
                if pursuit then begin
                    if (wallresult <> 'z') then begin
                        if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,93,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,94,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,88,mbfound){wall not blocking}
                    end{wall}
                    else if (wallresult = 'z') then
                      writealloutput(collectivetable,colloutfile,name,88,mbfound){no wall}
                end{no pursuit}
                else if not pursuit then begin
                    if (wallresult <> 'z') then begin
                        if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,93,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,94,mbfound);{block prisoners}
                    end;{wall}
                end;{no pursuit}
            end;{no victory}
        end{fast}
        else if not fast then begin
            if victory then begin
                if (wallresult <> 'z') then begin
                    if (victorytime < walltime) then
                      writealloutput(collectivetable,colloutfile,name,85,mbfound){victory first}
                    else if (walltime < victorytime) then begin
                        if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,124,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,125,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,85,mbfound){wall not blocking}
                    end{wall first}
                end{wall}
                else if (wallresult = 'z') then
```

```
                  writealloutput(collectivetable,colloutfile,name,85,mbfound)
            end{victory}
            else if not victory then begin
                if pursuit then begin
                    if (wallresult <> 'z') then begin
                        if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,91,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,92,mbfound){block prisoners}
                        else
                          writealloutput(collectivetable,colloutfile,name,87,mbfound){wall not blocking}
                    end{wall}
                    else if (wallresult = 'z') then
                      writealloutput(collectivetable,colloutfile,name,87,mbfound){no wall}
                end{no pursuit}
                else if not pursuit then begin
                    if (wallresult <> 'z') then begin
                        if wallresult in blockenemieselsewhere then
                          writealloutput(collectivetable,colloutfile,name,91,mbfound){block enemies elsewhere}
                        else if wallresult in blockprisoners then
                          writealloutput(collectivetable,colloutfile,name,92,mbfound);{block prisoners}
                    end;{wall}
                end;{no pursuit}
            end;{no victory}
        end;{not fast}
    end;{cause = C}



begin
  eventSearch.Create(fn);
  stateSearch.Create(fn);
  subjectdone:= false;
  cause:= ' ';
  stoptime:= 0; {doit etre utilis, pour les analyses pendant le mode}
  stoptime_temp:= 0;{doit prendre la valeur de la fin du mode meme si il existe un nveau niveau}
  levelend:= 0;{doit prendre une valeur seul. si il y a chgement de nive.}
  i:= 0; {number of powermodes found}

  (*** find powermodes states ***)
  while moveToNextState(statesearch,isPowered,subjectdone)  do begin
      (** initialize values for each start of powerstate **)


      mbfound:= false;
      i:= i + 1;
      levelend:= 0;
      stoptime:=0 ;
      prison:= false;
      pursuit_F:= false;
      pursuit_E:= false;
      wait:= false;
      fast:= false;
      safestart:= false;
      quitsafety:= false;
      entersafety:= false;
      entertime:= 0;
      walltime:= 0;
      pursuitbonusstoptime:= 0;
      wallresult:= 'z';
      risk:= false;
      victory:= false;
      victoryE:= false;
      victoryF:= false;
      playerhurt:= false;
      playerkilled:= false;
      pursuit:= false;
      faststart:= false;
      safecollect:= false;
      collect:= false;
      maladapted:= false;
      victorytime:= 0;
      victoryEtime:= 0;
      victoryFtime:= 0;
      risktime:= 0;
      quittime:= 0;
      killtime:= 0;
      hurttime:= 0;
      bonustime:= 0;
      first:= 0;
      bonustaken:= false; {if = true then bonus was visible at starttime + eaten by player}
      bonusvisible:= false;
      missbonus:= false;
      endofbonus:= 0;
      collecttime:= 0;
      fatalrisk:= 0;
      toolcollect:= false;
      lastdistanceE:= false;
      lastdistanceF:= false;
      lastbonusdistance:= false;
      Eneverthere:= true;
      Fneverthere:= true;
      pursuitstoptime:= 0;
      (** find start, stop and levelend values **)
      starttime:= statesearch.stateinfos.time;
      powerstartposition:= statesearch.position;

      if not statesearch.movetoposition(statesearch.position - 1) then;
      if (statesearch.stateinfos.mindistJ = -1) and
       (statesearch.stateinfos.mindistH = -1) and
        (statesearch.stateinfos.mindistF = -1) and
         (statesearch.stateinfos.mindistE  > -1) then
          closestEnemy:= 'E'
      else if (statesearch.stateinfos.mindistJ > -1) then begin
          if ((statesearch.stateinfos.mindistE  = -1) or
           (statesearch.stateinfos.mindistJ <= statesearch.stateinfos.mindistE)) then
            closestEnemy:= 'J'
          else closestEnemy:= 'E';
      end
      else if (statesearch.stateinfos.mindistH > -1) then begin
          if ((statesearch.stateinfos.mindistE  = -1) or
           (statesearch.stateinfos.mindistH <= statesearch.stateinfos.mindistE)) then
            closestEnemy:= 'H'
          else closestEnemy:= 'E';
      end
      else if (statesearch.stateinfos.mindistF > -1) then begin
          if ((statesearch.stateinfos.mindistE  = -1) or
           (statesearch.stateinfos.mindistF <= statesearch.stateinfos.mindistE)) then
            closestEnemy:= 'F'
          else closestEnemy:= 'E';
      end
      else closestEnemy:= 'Z';

      danger:= statesearch.stateinfos.danger;
      fatality:= statesearch.stateinfos.fatality;
      visibleenemies:= statesearch.stateinfos.nbrofvisibleenemies;

      stoptime_temp:= findendofmode(statesearch,fn,starttime,isPowered,subjectdone,levelend);

      (** if levelend > 0 the analysis has to stop there, stoptime
      must thus take the value levelend **)
      if levelend > 0 then
        stoptime:= levelend
      else stoptime:= stoptime_temp;

      if not statesearch.movetostateattime(stoptime)then;
      powerstopposition:= statesearch.position;

      (* check how player became powered *)
      cause:= causalsearch(eventsearch,starttime);


      (* check if player is in the prison: if yes the analysis has to stop here*)

      if (statesearch.stateinfos.onoffstates[IsPlayerInPrison]) then
        prison:= true;


      if (prison = false) then begin

          (* check if powermode started in a safeplace *)
          if not statesearch.movetostateattime(starttime) then
            writeln('problem line 123:37 of nwpower ');
          if ( isSafeplace(statesearch) and notallenemiesinprison(statesearch)) then begin
              safestart:= true;

              (*check if player exits the safeplace*)
              quittime:= (checkexitfromstarttostop(statesearch,starttime,stoptime));
              if quittime > 0 then
                quitsafety:= true;
          end

          (*check if player enters and stays in safe place*)
          else begin
              entertime:= checkenterfromstarttostop(statesearch,eventsearch,starttime,stoptime);
              if ( (entertime > 0) and
               ((entertime < levelend) or (levelend = 0)) and
                notallenemiesinprison(statesearch) ) then
                  entersafety:= true;
          end;

          (* check if player waited for Janus *)
          if (cause = 'B') and (waitforjanus(statesearch,starttime) = true) then
            wait:= true;

          (* check if player places a wall and if yes, what for *)
          if oheventexists(fn,[apply_wall_snd],starttime,stoptime,walltime) then begin
              if not eventsearch.movetofirsteventattime(walltime) then;
              if not statesearch.movetostateattime(walltime) then;
              wallresult:= walltest(statesearch,eventsearch);
          end;

          (* check if player uses a risk tool *)
          if oheventexists(fn,[apply_teleport_snd],starttime,stoptime,risktime) then
            risk:= true;

          (* check if player defeats enemies*)
          if oheventexists(fn,[victory_snd],starttime,stoptime,victoryEtime) then
              victoryE:= true;

          if oheventexists(fn,[bravo_snd],starttime,stoptime,victoryFtime) then
              victoryF:= true;

          if ( victoryE = true) or (victoryF = true) then begin
              victory:= true;
              if (victoryF and victoryE) then begin
                  if victoryEtime < victoryFtime then
                    victorytime:= victoryEtime
                  else
                    victorytime:= victoryFtime;
              end
              else if victoryF and (not victoryE) then
                victorytime:= victoryFtime
              else if victoryE and (not victoryF) then
                victorytime:= victoryEtime;
          end;


          (* check if player is hurt *)
          if oheventexists(fn,esplayerhurt,starttime,stoptime + aftertime,hurttime) then
              playerhurt:= true;

          (* check if player dies  just at the end of powermode or during powermode *)
          if oheventexists(fn,[die_snd],starttime,stoptime + aftertime,killtime) then
              playerkilled:= true;

          (*check if player uses a roller in the same time*)
          if (checkfromstarttostop(statesearch,starttime,stoptime,isFast) >0) then
            fast:= true;
          if not statesearch.movetostateattime(starttime) then;
          if statesearch.stateinfos.onoffstates[isfast] then
            faststart:= true;


          (* check if player collects object *)
          if oheventexists(fn,(escollecttools + [get_super_snd]+ [get_magic_snd]),starttime,stoptime,collecttime) then
         begin
              Toolcollect:= true;
              if not statesearch.movetostateattime(collecttime) then
                writeln(' problem at nwpower: line 190.58');
              if( isSafeplace(statesearch) and notallenemiesinprison(statesearch)) then
                safecollect:= true;
          end;

         (* check visibility of bonus before powermode*)
         if not statesearch.movetostateattime(starttime) then
           writeln(' problem at nwpower: line 197.14');
         if (statesearch.stateinfos.onoffstates[isBonusVisible]) then
           bonusvisible:= true;

         if bonusvisible then begin
             if not statesearch.movetostateattime(starttime) then;
             repeat
                 if not (statesearch.stateinfos.onoffstates[isbonusvisible]) then begin
                   endofbonus:= statesearch.stateinfos.time;
                   break;
                 end;
                 if not statesearch.movetoposition(statesearch.position + 1) then
                   endofbonus:= statesearch.stateinfos.time;
             until endofbonus > 0;

             if (oheventexists(fn,[bonus_get_snd],starttime,endofbonus,bonustime) )then
               bonustaken:= true;


             pursuitbonusstoptime:= endofbonus;
             if (playerhurt and (hurttime < pursuitbonusstoptime)) then begin
                 pursuitbonusstoptime:= hurttime;
                 if not statesearch.movetostateattime(hurttime)then;
                 if not statesearch.movetoposition(statesearch.position - 1) then;
                 if ((statesearch.stateinfos.bonusdistance < 4) and
                  (statesearch.stateinfos.bonusdistance > -1) ) then begin
                     if not statesearch.movetostateattime(endofbonus)then;
                     if not statesearch.movetoposition(statesearch.position - 1) then;
                     if not (statesearch.stateinfos.bonusdistance > 10) then
                       lastbonusdistance:= true;
                 end;
             end{playerhurt}
             else if (playerkilled and (killtime < pursuitbonusstoptime)) then begin
                 pursuitbonusstoptime:= killtime;
                 if not statesearch.movetostateattime(killtime)then;
                 if not statesearch.movetoposition(statesearch.position - 1) then;
                 if ((statesearch.stateinfos.bonusdistance < 4) and
                  (statesearch.stateinfos.bonusdistance > -1)  ) then begin
                     if not statesearch.movetostateattime(endofbonus)then;
                     if not statesearch.movetoposition(statesearch.position - 1) then;
                     if not (statesearch.stateinfos.bonusdistance > 10) then
                       lastbonusdistance:= true;
                 end;
             end{playerkilled}
             else begin
                 if not statesearch.movetostateattime(endofbonus)then;
                 if not statesearch.movetoposition(statesearch.position -1) then;
                 if((statesearch.stateinfos.bonusdistance < 4) and
                  (statesearch.stateinfos.bonusdistance > -1) ) then
                   lastbonusdistance:= true;
             end;{not hurt, not killed}
             if (pursuitbonus(statesearch,'bonus',starttime,pursuitbonusstoptime)
              and lastbonusdistance) then
               missbonus:= true;
         end;{bonus visible}



         if bonustaken or toolcollect then
           collect:= true;


         (* check if player tried to pursuit enemies*)
         if victoryF then
```

```
  pursuitstoptime:= victoryFtime
else if victoryE then
  pursuitstoptime:= victoryEtime
else begin
   if hurttime > 0 then begin
      pursuitstoptime:= hurttime;
      if not statesearch.movetostateattime(hurttime)then;
      if not statesearch.movetoposition(statesearch.position - 1) then;

      if (statesearch.stateinfos.mindistE < 7 ) and (statesearch.stateinfos.mindistE > -1)
        then lastdistanceE:= true;

      if (statesearch.stateinfos.mindistF < 7 ) and (statesearch.stateinfos.mindistF > -1)
        then lastdistanceF:= true;
    end

    else if killtime > 0 then begin
      pursuitstoptime:= killtime;
      if not statesearch.movetostateattime(killtime)then;
      if not statesearch.movetoposition(statesearch.position - 1) then;
      if (statesearch.stateinfos.mindistE < 7 ) and (statesearch.stateinfos.mindistE > -1)
        then lastdistanceE:= true;
      if (statesearch.stateinfos.mindistF < 7 ) and (statesearch.stateinfos.mindistF > -1)
        then lastdistanceF:= true;
    end

    else begin
      pursuitstoptime:= stoptime;
      if not statesearch.movetostateattime(stoptime)then;
      if (statesearch.stateinfos.mindistE < 7 ) and (statesearch.stateinfos.mindistE > -1)
        then lastdistanceE:= true;
      if (statesearch.stateinfos.mindistF < 7 ) and (statesearch.stateinfos.mindistF > -1)
        then lastdistanceF:= true;

    end;
end;


if not statesearch.movetostateattime(starttime) then ;
while not (statesearch.stateinfos.time > stoptime) do begin
    if (statesearch.stateinfos.mindistE > -1) then
      Eneverthere:= false;
    if (statesearch.stateinfos.mindistF > -1) then
      Fneverthere:= false;
    if not statesearch.movetoposition(statesearch.position + 1) then break;
end;
if not statesearch.movetostateattime(starttime) then;

if Eneverthere = false then begin
    if pursuitE(statesearch,'E',starttime,pursuitstoptime)
     and ((lastdistanceE = true) or (victoryE = true)) then
      pursuit_E:= true
    else if pursuitE(statesearch,'E',starttime,pursuitstoptime) and missbonus then
        missbonus:= false;{poursuite mais pas toutes les conditions, suffit pour que pas missbonus!!}
end;

if Fneverthere = false then begin
    if pursuitF(statesearch,'F',starttime,pursuitstoptime)
     and ( (lastdistanceF = true) or (victoryF = true) ) then
      pursuit_F:= true
    else if pursuitF(statesearch,'F',starttime,pursuitstoptime) and missbonus then
        missbonus:= false;{poursuite mais pas toutes les conditions, suffit pour que pas missbonus!!}
end;



if ((pursuit_E = true) and (pursuit_F = true)) then
  pursuit:= true
else if ( (pursuit_E = true) and (pursuit_F = false)) then
  pursuit:= true
else if ((pursuit_E = false) and (pursuit_F = true)) then
  pursuit:= true
else if ((pursuit_E = false) and (pursuit_F = false)) then
  pursuit:= false;


{ check maladapted}
if not statesearch.movetostateattime(starttime) then;
if not statesearch.movetoposition(statesearch.position-1) then;
if (cause = 'A') and ((statesearch.stateinfos.nbrofvisibleenemies = 0) or
 ((notallenemiesinprison(statesearch) = false) and
  (statesearch.stateinfos.onoffstates[isplayerinprison] = false) ) ) then
   maladapted:= true
else if (cause = 'B') and wait and
 ((statesearch.stateinfos.nbrofvisibleenemies = 0) or
  ((notallenemiesinprison(statesearch) = false) and
   (statesearch.stateinfos.onoffstates[isplayerinprison] = false) ) )  then
     maladapted:= true;


(*compute results of the analysis *)
if (cause = 'A') then
  computecauseA

else if (cause = 'B') then
  computecauseB

else if (cause = 'C') then
  computecauseC;
```

```
        end;{if player is not in prison}

        if mbfound then begin
            write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
            write(colloutfile,closestEnemy,'  ');
            write(colloutfile,visibleenemies:3:0);

             write(colloutfile,(danger * (danger + fatality) ):6:2 );
            write(colloutfile,'  ',starttime/100:10:2);
            write(colloutfile, '   ', time2string(starttime/100));

            writeln(colloutfile);

        end;



        if not statesearch.movetostateattime(stoptime_temp) then;
         { writeln(' problem at nwpower: line 659.46');}(* those 2 lines must stay here, no change. *)
        if subjectdone then break;
    end;{while new powermode is found applied}

    eventSearch.Done;
    stateSearch.Done;

  end;{powermode analysis: whole procedure}


begin
    lastpos:= 0;
    prepared:= false;
    prepare2nd:= false;
    assign(collective_outfile,outputdir + outfilename + textext);
    rewrite(collective_outfile);
    writeln(collective_outfile,' File: ',outfilename + textext);
    writeln(collective_outfile);
    Rewrite(collective_outFile);


    assign(collective2nd,outputdir + 'genlevels' + textext);
    Rewrite(collective2nd);
    initializecolldata(collectivetable,relevantcodes);
    prepareoutput(collective2nd,collectiveTable,prepare2nd,precedingforoutput); (*** prepare collective outputfile ***)

    repeat
        infilename:= findinfile(lastpos,experimentfiles,alldone);
        val(infilename,subjectname,nothing);

        dirinfilename:= inputdir + infilename;

        if alldone then break;
        setName(CollectiveTable,infilename);
        (* initialisation des valeurs *)
        for i:= 1 to ccd_num_of_codes do begin
            if i in relevantcodes then
              setCollectiveCode( Collectivetable,i,0);
        end;
        Analyzepowermode(collective_outfile,collectiveTable,dirinfilename,subjectname);
        analyzebonus(collective_outfile,collectiveTable,dirinfilename,subjectname);
        analyzewalltools(collective_outfile,collectiveTable,dirinfilename,subjectname);
        analyzetelephone(collective_outfile,collectiveTable,dirinfilename,subjectname);
        analyzespeedmode(collective_outfile,collectiveTable,dirinfilename,subjectname);
        analyzeshieldmode(collective_outfile,collectiveTable,dirinfilename,subjectname);
        analyzerisk(collective_outfile,collectiveTable,dirinfilename,subjectname);
        analyzerepair(collective_outfile, collectiveTable,dirinfilename,subjectname);
        analyzeprisonandpower(collective_outfile, collectiveTable,dirinfilename,subjectname);
        analyzehome(collective_outfile, collectiveTable,dirinfilename,subjectname);
        analyzehourglass(collective_outfile, collectiveTable,dirinfilename,subjectname);

        { *** write results in collective outfile}
        write(collective2nd,' ',collectiveTable.name,'  ');
        for x:= 1 to ccd_num_of_codes do begin
            if CollectiveTable.data[x].mask then
              write(collective2nd, getCollectiveCode(CollectiveTable,x):7);
        end;
        writeln(collective2nd);

    until alldone;
    if alldone = true then begin
      writeln ('no more files ');{done means all the files have been analyzed}
      close(collective_outfile);
      close(collective2nd);
    end;
end.
```

```
program outcomes;
{This program analyses the behaviors of the subject when he is in powermode.}
uses nperslib,nwstat,wallfunctions,nwdanlib,genlib,CCD,Common,AnaLib;

const prefix                 = 'outco';
      precedingforoutput     = 'out';
      aftertime              =  Round(3.0 * 100);
      outfilename            = 'outcomall';
      relevantcodes          = [98,{99,}105,140,145,146];

var   Collectivetable        : Tcollectivecodedata;
      alldone,prepared,
      prepare2nd             : boolean;{true= quand l'analyse a ,t, faite sur tous les fichiers}
      lastpos                : integer;{nombre a partir duquel il faut g,n,rer des num,ros de code}
      dirinfilename          : string;
      mbfound                : boolean;
      danger,fatality,
      visibleenemies         : real;
      closestEnemy           : Char;
      subjectname            : integer;
      nothing,x,i            : integer;
      collective2nd          : text;

procedure writeoutputforoutcomes(var colltable: Tcollectivecodedata;var outfile: text;
                             fname: integer; mbcode: integer);
    begin
        incCollectiveCode(colltable,mbcode);
        if mbcode < 10 then
          write(outfile,fname:4,'   ',mbcode,' ')
        else if mbcode < 100 then
          write(outfile,fname:4,'  ',mbcode,' ')
        else if mbcode >= 100 then
          write(outfile,fname:4,' ',mbcode,' ');
    end;

procedure Analyzehurt1(var colloutFile : Text;var collectivetable: tcollectivecodedata;fn : String;
                               name: integer);

  var   eventSearch                      : TEventSearch;
        stateSearch                      : TStateSearch;
        hurttime                         : Ttime;
        x,i,survivaltools,usefultools,
        totaltools                       : integer;


  begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);

    i:= 0; {total of walls}


    (*** find wall application events ***)
    while eventSearch.MoveToNextEvent(cry_snd,anylevel) do begin
        hurttime:= eventsearch.eventinfos.time;
        survivaltools:= 0;
        usefultools:= 0;

        if not statesearch.movetostateattime(hurttime)then;

        (*count tools in toolbox*)
        checktools(eventsearch,statesearch,hurttime,survivaltools,usefultools,totaltools);

        writeoutputforoutcomes(collectivetable,colloutfile,name,140);

        write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
        write(colloutfile,survivaltools:4,usefultools:4);
        write(colloutfile,'  ',hurttime/100:10:2);
        write(colloutfile, '   ', time2string(hurttime/100));
        writeln(colloutfile);
    end;{while new wall applied}

    eventSearch.Done;
    stateSearch.Done;


  end;{hurt1 analysis: whole procedure}


procedure Analyzehurt2(var colloutFile : Text;var collectivetable: Tcollectivecodedata;
                           fn : String; name: integer);

  var   eventSearch                      : TEventSearch;
        stateSearch                      : TStateSearch;
        hurttime                         : Ttime;
        x,i,survivaltools,usefultools,
        totaltools                       : integer;


  begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);

    i:= 0; {total of walls}

    (*** find wall application events ***)
    while eventSearch.MoveToNextEvent(whine_snd,anylevel) do begin
        hurttime:= eventsearch.eventinfos.time;
        survivaltools:= 0;
        usefultools:= 0;
        if not statesearch.movetostateattime(hurttime)then;

        (*count tools in toolbox*)
        checktools(eventsearch,statesearch,hurttime,survivaltools,usefultools,totaltools);

        writeoutputforoutcomes(collectivetable,colloutfile,name,140);
        write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
        write(colloutfile,survivaltools:4,usefultools:4);
        write(colloutfile,'  ',hurttime/100:10:2);
        write(colloutfile, '   ', time2string(hurttime/100));
        writeln(colloutfile);
    end;{while new wall applied}

    eventSearch.Done;
    stateSearch.Done;

  end;{hurt2 analysis: whole procedure}

procedure Analyzelosses(var colloutFile : Text;var collectivetable: tcollectivecodedata;
                           fn : String; name: integer);

  var   eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;
        hurttime                          : Ttime;
        x,i,survivaltools,usefultools,
        totaltools                        : integer;

  begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);

    i:= 0; {total of walls}

    (*** find wall application events ***)
    while eventSearch.MoveToNextEvent(die_snd,anylevel) do begin
        hurttime:= eventsearch.eventinfos.time;
        survivaltools:= 0;
        usefultools:= 0;

        if not statesearch.movetostateattime(hurttime)then;
        (*count tools in toolbox*)
        checktools(eventsearch,statesearch,hurttime,survivaltools,usefultools,totaltools);

        writeoutputforoutcomes(collectivetable,colloutfile,name,105);
        write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
        write(colloutfile,survivaltools:4,usefultools:4);
        write(colloutfile,'  ',hurttime/100:10:2);
        write(colloutfile, '   ', time2string(hurttime/100));
        writeln(colloutfile);

    end;{while new wall applied}

    eventSearch.Done;
    stateSearch.Done;


  end;{hurt1 analysis: whole procedure}

procedure AnalyzeConfigurations(var colloutFile : Text;var collectivetable: Tcollectivecodedata;
                                   fname : String;name: integer);

  var   eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;

        startTime,levelend,lastpowerstart,
        lastshieldstart,lastprisonstart,
        lastspeedstart,stopTime,
        stoptime_temp,excludingtime,
        hurttime,killtime,tooltime                 : TTime;
        subjectdone,playerhurt,
        playerkilled,power,prison,
        teleport,excluded,toolused        : boolean;
        survivaltools,usefultools,
        totaltools,x,i,
        survivalused,usefulused,
        alltoolsused,totaluseful,
        totalsurvival,TBsurvival,
        TBuseful,nbrsurvival1            : integer;

begin
    eventSearch.Create(fname);
    stateSearch.Create(fname);
    subjectdone:= false;
    i:= 0; {total of configurations}

    (*** find new configuration ***)
    while MoveToNextconfiguration(statesearch,subjectdone,IsConfigurationB) do begin

        starttime:= stateSearch.stateinfos.time;

        (* initialization for each new shieldmode found, before analysis *)
        inc(i);
        stoptime:= 0; {doit etre utilis, pour les analyses pendant le mode}
        stoptime_temp:= 0;{doit prendre la valeur de la fin du mode meme si il existe un nveau niveau}
        levelend:= 0; (* necessary for the findendofmode function *)
        playerhurt:= false;
        playerkilled:= false;
        teleport:= false;
        survivaltools:= 0;
        usefultools:= 0;
        totaltools:= 0;
        excluded:= false;
        excludingtime:= 0;
        hurttime:= 0;
        killtime:= 0;
        toolused:= false;
        tooltime:= 0;
        (* for configurations a-k use the nomoredanger;
           for configurations l,m,n,o use the findendofconfig.*)
        {stoptime_temp:= findendofConfig(statesearch,fname,starttime,isConfigurationL,subjectdone,levelend);}
        stoptime_temp:= nomoredanger(statesearch,fname,starttime,isConfigurationB,subjectdone,levelend);

        (** if levelend > 0 the analysis has to stop there, stoptime
        must thus take the value levelend **)
        if levelend > 0 then
          stoptime:= levelend
        else stoptime:= stoptime_temp;


        (*count tools in toolbox*)
        checktools(eventsearch,statesearch,starttime,survivaltools,usefultools,totaltools);

        if not eventsearch.movetofirsteventatoraftertime(starttime) then;
        while eventsearch.eventinfos.time < stoptime do begin
            if eventsearch.eventinfos.event in esapplytools then begin
                toolused:= true;
                tooltime:= eventsearch.eventinfos.time;
            end
            else if (eventsearch.eventinfos.event = home_in_snd) or
             ( eventsearch.eventinfos.event = teleport_snd) then begin
               excluded:= true;
               excludingtime:= eventsearch.eventinfos.time;
            end;
            if not eventsearch.movetoposition(eventsearch.position  + 1) then;
        end;

        (* check if player is hurt *)
        if oheventexists(fname,esplayerhurt,starttime,stoptime,hurttime) then
            playerhurt:= true;

        (* check if player dies *)
        if oheventexists(fname,[die_snd],starttime,stoptime,killtime) then
            playerkilled:= true;
        if ( (toolused = false) and (playerhurt = false) and
         (playerkilled = false) and (excluded = true) )then
          i:= i - 1
        else begin
            writeoutputforoutcomes(collectivetable,colloutfile,name,145);
            write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
            write(colloutfile,survivaltools:4,usefultools:4);
            write(colloutfile,'  ',starttime/100:10:2);
            write(colloutfile, '   ', time2string(starttime/100));
            writeln(colloutfile);
        end;

        if not statesearch.movetostateattime(stoptime_temp) then;
        (* those 2 lines must stay here, no change. *)
        if subjectdone then break;
    end;{while new configuration is found}


    eventSearch.Done;
    stateSearch.Done;


  end;{Configuration analysis: whole procedure}

procedure testbonus(var colloutfile: Text;var collectivetable: Tcollectivecodedata;
                       fn : String; name: integer);

  var   eventSearch                       : TEventSearch;
        stateSearch                       : TStateSearch;
        startTime,levelend,
        stopTime,stoptime_temp,bonustime,
        speedtime, powertime,entertime,
        pursuitstoptime,hurttime,
        killtime                          : TTIme;
        bonusmissed,entersafety,
        collectbonus, speed,power,
        playerhurt,playerkilled,
        subjectdone,lastbonusdistance     : Boolean;
        i                                 : integer;

begin
    eventSearch.Create(fn);
    stateSearch.Create(fn);
    subjectdone:= false;
    i:= 0; {total of visiblebonuses}


    (*** find bonus appearing events ***)
    while eventSearch.MoveToNextEvent(bonus_app_snd,anylevel) do begin
        starttime:= eventsearch.eventinfos.time;
        if not stateSearch.MoveToStateAtTime(starttime) then begin
          WriteLn('Unable to find state');
          Break
        end;{v,rification que dans le fichier mes l',tat existe au meme moment}

        (*initialization for each new bonus found, before analysis*)
        IncCollectivecode(collectivetable,146);

        inc(i);
        levelend:= 0; (*necessary for the findendofmode function *)
        stoptime_temp:= 0;
        collectbonus:= false;
        speed:= false;
        power:= false;
        powertime:= 0;
        speedtime:= 0;
        hurttime:= 0;
        entertime:= 0;
        killtime:= 0;
        playerhurt:= false;
        playerkilled:= false;
        entersafety:= false;
        lastbonusdistance:= false;
        bonusmissed:= false;
        bonustime:= 0;

        stoptime_temp:= findendofmode(statesearch,fn,starttime,isBonusVisible,subjectdone,levelend);

        (** if levelend > 0 the analysis has to stop there, stoptime
        must thus take the value levelend **)
        if levelend > 0 then
          stoptime:= levelend
        else stoptime:= stoptime_temp;

        if ohEventexists(fn,[bonus_get_snd],starttime,stoptime + reactiontime,bonustime) then
          collectbonus:= true;


        (* check if player is hurt *)
        if oheventexists(fn,esplayerhurt,starttime,stoptime,hurttime) then
            playerhurt:= true;


        (* check if player dies *)
        if oheventexists(fn,[die_snd],starttime,stoptime,killtime) then
          playerkilled:= true;



        {check if player enters safe place and stays > minhometime}
        if not statesearch.movetostateattime(starttime) then;
        begin
            while ((statesearch.stateinfos.time < stoptime) and (not entersafety)) do begin
                if not statesearch.movetoposition(statesearch.position + 1) then;

                if (isSafeplacespecial(statesearch) and
                 notallenemiesinprison(statesearch) and
                  (computestateduration(statesearch,isSafeplacespecial) >= minhometime) ) then
                    entersafety:= true;
            end;
            if not statesearch.movetostateattime(starttime) then;
        end;


     {   if not collectbonus then begin
            pursuitstoptime:= stoptime;
            if (playerhurt and (hurttime < pursuitstoptime)) then begin
                pursuitstoptime:= hurttime;
                if not statesearch.movetostateattime(hurttime)then;
                if not statesearch.movetoposition(statesearch.position - 1) then;
                if ((statesearch.stateinfos.bonusdistance < 5) and
                 (statesearch.stateinfos.bonusdistance > -1) ) then
                  lastbonusdistance:= true;
            end{playerhurt}
       {     else if playerkilled  then begin
                pursuitstoptime:= killtime;
                if not statesearch.movetostateattime(killtime)then;
                if not statesearch.movetoposition(statesearch.position - 1) then;
                if ((statesearch.stateinfos.bonusdistance < 5) and
                 (statesearch.stateinfos.bonusdistance > -1)  ) then begin
                    if not statesearch.movetostateattime(stoptime)then;
                    if not statesearch.movetoposition(statesearch.position - 1) then;
                    if not (statesearch.stateinfos.bonusdistance > 10) then
                     lastbonusdistance:= true;
                end
                else lastbonusdistance:= false;
            end{playerkilled}
        {    else begin
                if not statesearch.movetostateattime(pursuitstoptime)then;
                if not statesearch.movetoposition(statesearch.position - 1) then;
                if((statesearch.stateinfos.bonusdistance < 5) and
                 (statesearch.stateinfos.bonusdistance > -1) ) then
                  lastbonusdistance:= true;
            end;{not hurt, not killed}
      {       if (pursuitbonus(statesearch,'bonus',starttime,pursuitstoptime)
             and lastbonusdistance) then
               bonusmissed:= true;
        end;{no collect bonus}


        (*compute results of the analysis *)
        if (collectbonus = true) then
         inccollectivecode(collectivetable,98);
      {  begin
            writeoutputforoutcomes(collectivetable,colloutfile,name,98);
            write(colloutfile,statesearch.stateinfos.currentlevel:3,'  ');
```

```
            write(colloutfile,'  ',starttime/100:18:2);
            write(colloutfile, '    ', time2string(starttime/100));
            writeln(colloutfile);
        end;
      }{pour l'instant pas n,cessaire dans l'output vertical, seulemetn dans l'horizontal}

        if not statesearch.movetostateattime(stoptime_temp) then;
         { writeln(' problem at nwpower: line 659.46');}(* those 2 lines must stay here, no change. *)
        if subjectdone then break;
    end;{while new bonus visible applied}

    eventSearch.Done;
    stateSearch.Done;

  end;{bonus analysis: whole procedure}

begin
    lastpos:= 0;
    prepared:= false;
    prepare2nd:= false;

    assign(collective_outfile,outputdir + outfilename + textext);
    rewrite(collective_outfile);
    writeln(collective_outfile,' File: ',outfilename + textext);
    writeln(collective_outfile);
    Rewrite(collective_outFile);

    assign(collective2nd,outputdir + 'genreslts' + textext);
    Rewrite(collective2nd);
    initializecolldata(collectivetable,relevantcodes);
    prepareoutput(collective2nd,collectiveTable,prepare2nd,precedingforoutput); (*** prepare collective outputfile ***)

    repeat
        infilename:= findinfile(lastpos,experimentfiles,alldone);
        val(infilename,subjectname,nothing);

        dirinfilename:= inputdir + infilename;
        if alldone then break;

        setName(CollectiveTable,infilename);
        (* initialisation des valeurs *)
        for i:= 1 to ccd_num_of_codes do begin
            if i in relevantcodes then
              setCollectiveCode( Collectivetable,i,0);
        end;
        Analyzehurt1(collective_outfile,collectivetable,dirinfilename,subjectname);
        Analyzehurt2(collective_outfile,collectivetable,dirinfilename,subjectname);
        Analyzelosses(collective_outfile,collectivetable,dirinfilename,subjectname);
        analyzeconfigurations(collective_outfile,collectivetable,dirinfilename,subjectname);
        testbonus(collective_outfile,collectivetable,dirinfilename,subjectname);

        { *** write results in collective outfile}
        write(collective2nd,' ',collectiveTable.name,'  ');
        for x:= 1 to ccd_num_of_codes do begin
            if CollectiveTable.data[x].mask then
              write(collective2nd, getCollectiveCode(CollectiveTable,x):7);
        end;
        writeln(collective2nd);

    until alldone;
    if alldone = true then begin
      writeln ('no more files ');{done means all the files have been analyzed}
      close(collective_outfile);
      close(collective2nd);
    end;
end.
```