Problème I - TRIANGLE DE PASCAL — 20 points

```
program PASCAL;

const DIM = 20;

type T_PA = array [1..DIM,1..DIM] of integer;

var PA : T_PA;
    I,J,M : integer;

procedure INIT (var A : T_PA);
var I,J : integer;
begin
   for I := 1 to DIM do
      for J := 1 to DIM do
         if (J = 1) or (I = J)
         then A[I,J] := 1
         else A[I,J] := 0
end;

procedure OUTPUT (A : T_PA; M : integer);
var I,J : integer;
begin
   for I := 1 to M do
      begin
         for J := 1 to M do
            if A[I,J] <> 0
            then write(A[I,J],' ');
         writeln
      end
end;

procedure DOTRI (var A : T_PA; M : integer);
var I,J : integer;
begin
   for I := 3 to M do
      for J := 2 to M-1 do
         A[I,J] := A[I-1,J] + A[I-1,J-1]
end;

begin
   INIT(PA);
   readln(M);
   DOTRI(PA,M);
   OUTPUT(PA,M);
   readln
end.
```

Problème II - ACKERMANN — 20 points

```
program Ackermann;

var N,M : integer;

function A (N,M : integer) : int64;
begin
   if N = 0
   then A := M + 1
   else if M = 0
        then A := A(N-1,1)
        else A := A(N-1,A(N,M-1))
```

```
end;

begin
   readln(N,M);
   writeln(A(N,M));
   readln
end.
```

Cette fonction croît vraiment rapidement. Pour tester, nous avons compté le nombre d'appels récursifs. Pour les exemples donnés dans l'énoncé

A(2,2) = 7	on nécessite 27 appels récursifs
A(3,7) = 1021	on nécessite déjà 693.964 appels récursifs

et pour

A(3,12) = 32765	on nécessite 715.664.091 appels récursifs !

Problème III - LE TAQUIN — 60 points

```
program LeTaquin;

type tTaq = array [1..4,1..4] of integer;

var Taq : tTaq;
    Move : char;

procedure initTaquin (var Taq : tTaq);
var i,j,k : integer;
begin
   for i := 1 to 4 do
      for j := 1 to 4 do
         Taq[i,j] := 0;
   randomize;
   for k := 1 to 15 do
      begin
         repeat
            i := random(4) + 1;
            j := random(4) + 1
         until Taq[i,j] = 0;
         Taq[i,j] := k
      end
end;

procedure showTaquin (Taq : tTaq);
var i,j : integer;
begin
   for i := 1 to 4 do
      begin
         for j := 1 to 4 do
            if Taq[i,j] < 10
            then write(Taq[i,j],'   ')
            else write(Taq[i,j],'  ');
         writeln
      end;
   writeln
end;

procedure doMove (var Taq : tTaq; Move : char);
var i,j : integer;
    EmptyX,EmptyY,MoveX,MoveY : integer;
    poss : boolean;
begin
   for i := 1 to 4 do
      for j := 1 to 4 do
         if Taq[i,j] = 0
```

```
          then begin
                  EmptyX := i;
                  EmptyY := j
               end;
   poss := true;
   case Move of
      'i' : begin
               MoveX := EmptyX + 1;
               if MoveX > 4
               then poss := false;
               MoveY := EmptyY
            end;
      'm' : begin
               MoveX := EmptyX - 1;
               if MoveX < 1
               then poss := false;
               MoveY := EmptyY
            end;
      'j' : begin
               MoveX := EmptyX;
               MoveY := EmptyY + 1;
               if MoveY > 4
               then poss := false;
            end;
      'k' : begin
               MoveX := EmptyX;
               MoveY := EmptyY - 1;
               if MoveY < 1
               then poss := false;
            end
   end;
   if poss
   then begin
           Taq[EmptyX,EmptyY] := Taq[MoveX,MoveY];
           Taq[MoveX,MoveY] := 0
        end
end;

function done (Taq : tTaq) : boolean;
var temp : boolean;
    i,j,k : integer;
begin
   temp := true;
   k := 1;
   for i := 1 to 4 do
      for j := 1 to 4 do
         begin
            if (Taq[i,j] <> k) and (Taq[i,j] <> 0)
            then temp := false;
            if Taq[i,j] <> 0
            then k := k + 1
         end;
   done := temp
end;

begin
   initTaquin(Taq);
   showTaquin(Taq);
   repeat
       repeat
          readln(Move)
       until Move in ['i','j','k','m'];
       doMove(Taq,Move);
       showTaquin(Taq)
   until done(Taq);
   writeln('Bravo');
   readln
end.
```