*Partie théorique*

## Question 1

```
//1
procedure Derivee_Poly(lbA,lbD:TListBox);
var dA,i:integer;
    x:extended;
begin
  lbD.Clear;
  lbD.Items[0]:='0'; //la dérivée d'un polynôme constant est 0
  dA:=lbA.Items.Count-1;
  for i:=1 to dA do begin
                      x:=StrToFloat(lbA.Items[i]);
                      lbD.Items[i-1]:=FloatToStr(i*x);
                    end;
end;

//2
procedure Primitive_Poly(lbA,lbPrim:TListBox);
//La procédure donne la primitive à terme constant 0.
var dA,i:integer;
    x:extended;
begin
  lbPrim.Clear;
  lbPrim.Items[0]:='0';
  dA:=lbA.Items.Count-1;
  for i:=0 to dA do begin
                      x:=StrToFloat(lbA.Items[i]);
                      lbPrim.Items[i+1]:=FloatToStr(x/(i+1));
                    end;
end;
```

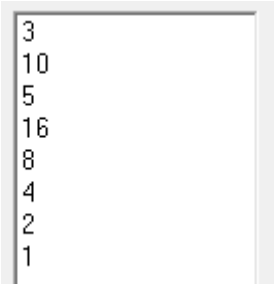
## Question 2

```
//1
function prime(n:int64):boolean;
//Le type int64 permet de traiter des entiers plus grands (64 bits)
var i:integer;
begin
if n<=2
then result:=n=2
else if n mod 2 = 0
     then result:=false
     else begin
            i:=3;
            result:=true ;
            while (i*i<=n) and result do
              if n mod i = 0 then result:=false else i:=i+2;
          end
end;

//2
function nextPrime(n:int64):int64;
begin
if prime(n) then result:=n else result:=nextPrime(n+1);
end;
```

## Question 3

(1)




(2) Version itérative :

```
procedure co_it(x:integer;lbA:TListBox);
begin
  while x>1 do
    begin
      lbA.Items.Add(inttostr(x));
      if x mod 2 = 0 then x:=x div 2 else x:=3*x+1
    end;
  lbA.Items.Add('1');
end;
```

***Partie pratique : Le jeu de taquin***

```
// Question 2
procedure gameInit(sgGame:TStringGrid;edtCol,edtRow:TEdit);
var i,j,r,c,n:integer;
begin
  r:=strtoint(edtRow.Text);
  c:=strtoint(edtCol.Text);
  sgGame.ColCount:=c;
  sgGame.RowCount:=r;
  n:=1;
  for j:=0 to r-1 do
    for i:=0 to c-1 do
      begin
        sgGame.Cells[i,j]:=inttostr(n);
        n:=n+1
      end;
  sgGame.Cells[c-1,r-1]:=''
end;

// Question 3
procedure emptyCell(sgGame:TStringGrid;var c,r:integer);
var i,j:integer;
begin
  for i:=0 to sgGame.ColCount-1 do
    for j:=0 to sgGame.RowCount-1 do
      if sgGame.Cells[i,j]='' then begin c:=i; r:=j end
end;
```

```
// Question 4
procedure MoveUp(sgGame:TStringGrid);
var c,r:integer;
begin
  emptyCell(sgGame,c,r);
  if r<sgGame.RowCount-1 then
    begin
      sgGame.Cells[c,r]:=sgGame.Cells[c,r+1];
      sgGame.Cells[c,r+1]:=''
    end
end;

procedure MoveDown(sgGame:TStringGrid);
var c,r:integer;
begin
  emptyCell(sgGame,c,r);
  if r>0 then
    begin
      sgGame.Cells[c,r]:=sgGame.Cells[c,r-1];
      sgGame.Cells[c,r-1]:=''
    end
end;

procedure MoveRight(sgGame:TStringGrid);
var c,r:integer;
begin
  emptyCell(sgGame,c,r);
  if c>0 then
    begin
      sgGame.Cells[c,r]:=sgGame.Cells[c-1,r];
      sgGame.Cells[c-1,r]:=''
    end
end;

procedure MoveLeft(sgGame:TStringGrid);
var c,r:integer;
begin
  emptyCell(sgGame,c,r);
  if c<sgGame.ColCount-1 then
    begin
      sgGame.Cells[c,r]:=sgGame.Cells[c+1,r];
      sgGame.Cells[c+1,r]:=''
    end
end;

// Question 5
procedure TfrmMain.FormCreate(Sender: TObject);
begin
  randomize;
  gameInit(sgGame,edtCol,edtRow)
end;
```

```
procedure TfrmMain.btnResetClick(Sender: TObject);
begin
   gameInit(sgGame,edtCol,edtRow);
end;

// Question 6
procedure TfrmMain.btnUpClick(Sender: TObject);
begin
  MoveUp(sgGame);
end;

procedure TfrmMain.btnDownClick(Sender: TObject);
begin
  MoveDown(sgGame);
end;

procedure TfrmMain.btnRightClick(Sender: TObject);
begin
  MoveRight(sgGame);
end;

procedure TfrmMain.btnLeftClick(Sender: TObject);
begin
  MoveLeft(sgGame);
end;

// Question 7
procedure TfrmMain.btnStartClick(Sender: TObject);
var i,r:integer;
begin
for i:=1 to 100*strtoint(edtCol.Text)*strtoint(edtRow.Text) do
    begin
      r:=random(2);
      if r=0 then MoveUp(sgGame) else MoveDown(sgGame);
      r:=random(2);
      if r=0 then MoveRight(sgGame) else MoveLeft(sgGame);
    end
end;
```