CIL 2008 • Epreuve de Finale
a) Solutions modèles en Pascal

Problème I – Shuttle puzzle 40 points

Avant de se lancer dans la programmation, il fallait avoir trouvé et compris la méthode à utiliser (comme toujours) en étudiant l'exemple donné dans l'énoncé.

Pour p = 3:

```
BBB_NNN
BB_BNNN          1 déplacement 'B'
BBNB_NN
BBNBN_N          2 déplacements 'N'
BBN_NBN
B_NBNBN
_BNBNBN          3 déplacements 'B'
NB_BNBN
NBNB_BN
NBNBNB_          3 déplacements 'N'
NBNBN_B
NBN_NBB
N_NBNBB          3 déplacements 'B'
NN_BNBB
NNNB_BB          2 déplacements 'N'
NNN_BBB          1 déplacement 'B'
15
```

Il suffit donc de compter de 1 à p et de faire autant de déplacements, en alternant la couleur, d'effectuer p déplacements, d'alterner la couleur et, enfin, de compter de p à 1 et de faire autant de déplacements, en alternant la couleur.

Pour le nombre de déplacements nécessaires, nous obtenons donc:

$$\sum_{1}^{p} p \; + \; p \; + \sum_{1}^{p} p \; =$$

$$\frac{p(p+1)}{2} \; + \; p \; + \; \frac{p(p+1)}{2} \; =$$

$$p(p+1) \; + \; p \; =$$

$$p(p+1+1) \; =$$

$$p(p+2)$$

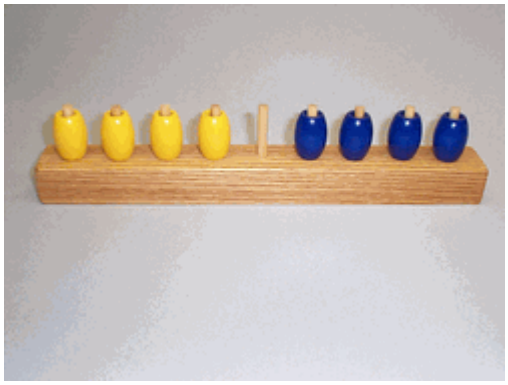
$$Soit, \; pour \; \; p = 3 : 3(3+2) = 15$$

$$Pour \; \; p = 30 : 30(30+2) = 960$$

Le programme suivant décide lui-même, à l'aide de la procédure MOVE, si le déplacement à faire est un glissement ou un saut.

```
program SHUTTEL_AUTOM;

const MAX = 30;

type TBOARD = array [1..2*MAX+1] of char;

var P,I,J : integer;
    BOARD : TBOARD;
    COUL : char;
    NUM : integer;

procedure INITBOARD (var BOARD : TBOARD);
var I : integer;
begin
   for I := 1 to P do
      BOARD[I] := 'B';
   BOARD[P+1] := '_';
   for I := P+2 to 2*P+1 do
      BOARD[I] := 'N'
end;

procedure SHOWBOARD (BOARD : TBOARD);
var I : integer;
begin
   for I := 1 to 2*P+1 do
      write(BOARD[I]);
   writeln
end;

function EMPTY (BOARD : TBOARD) : integer;
var TEMP,I : integer;
begin
   for I := 1 to 2*P+1 do
      if BOARD[I] = '_'
      then TEMP := I;
   EMPTY := TEMP
end;

procedure MOVE (COUL : char; var BOARD : TBOARD);
var E :  integer;
begin
   if COUL = 'B'
   then begin
           E := EMPTY(BOARD);
           if BOARD[E-1] = 'B'
           then begin
                   BOARD[E] := 'B';
                   BOARD[E-1] := '_'
                end
           else if (BOARD[E-1] = 'N') and (BOARD[E-2] = 'B')
                then begin
                        BOARD[E] := 'B';
                        BOARD[E-2] := '_'
                     end
        end
   else begin
```

```
            E := EMPTY(BOARD);
            if BOARD[E+1] = 'N'
            then begin
                    BOARD[E] := 'N';
                    BOARD[E+1] := '_'
                 end
            else if (BOARD[E+1] = 'B') and (BOARD[E+2] = 'N')
                 then begin
                         BOARD[E] := 'N';
                         BOARD[E+2] := '_'
                      end
         end;
   NUM := NUM + 1;
   SHOWBOARD(BOARD)
end;

procedure SWITCH (var COUL : char);
begin
   if COUL = 'B'
   then COUL := 'N'
   else COUL := 'B'
end;

begin
   write('p: ');
   readln(P);
   writeln;
   INITBOARD(BOARD);
   SHOWBOARD(BOARD);
   COUL := 'B';
   NUM := 0;
   for I := 1 to P do
      begin
         for J := 1 to I do
            MOVE(COUL,BOARD);
         SWITCH(COUL)
      end;
   for I := 1 to P do
      MOVE(COUL,BOARD);
   SWITCH(COUL);
   for I := P downto 1 do
      begin
         for J := 1 to I do
            MOVE(COUL,BOARD);
         SWITCH(COUL)
      end;
   writeln(NUM);
   readln
end.
```



Problème II - SMS 40 points

```
program SMS;

var F : text;
    STR1,STR2,STR3,S,STR4,STR5,STR6 : string;
    I,BYTES,DEC : integer;
    CAR : char;

procedure MIRR (var S : string);
var INPUT,OUTPUT : string;
    I : integer;
begin
   INPUT := S;
   OUTPUT := '';
   for I := 8 downto 1 do
      OUTPUT := OUTPUT + copy(INPUT,I,1);
   S := OUTPUT
end;

function bin2dec (BIN : string) : integer;
var I,P,BIT,VAL : integer;
begin
   P := 1;
   VAL := 0;
   for I := 7 downto 1 do
      begin
         BIT := strtoint(copy(BIN,I,1));
         VAL := VAL + BIT*P;
         P := P * 2
      end;
   bin2dec := VAL
end;

begin
   assign(F,'SMSIN.TXT');
   reset(F);
   readln(F,STR1);
   close(F);
   {writeln('input : ',STR1);}
   STR2 := '';
   BYTES := 0;
   for I := 1 to length(STR1) do
      case STR1[I] of
         '0' : STR2 := STR2 + '0000';
         '1' : STR2 := STR2 + '0001';
         '2' : STR2 := STR2 + '0010';
         '3' : STR2 := STR2 + '0011';
         '4' : STR2 := STR2 + '0100';
         '5' : STR2 := STR2 + '0101';
         '6' : STR2 := STR2 + '0110';
         '7' : STR2 := STR2 + '0111';
         '8' : STR2 := STR2 + '1000';
```

```
            '9' : STR2 := STR2 + '1001';
            'A' : STR2 := STR2 + '1010';
            'B' : STR2 := STR2 + '1011';
            'C' : STR2 := STR2 + '1100';
            'D' : STR2 := STR2 + '1101';
            'E' : STR2 := STR2 + '1110';
            'F' : STR2 := STR2 + '1111';
            ' ' : begin
                    STR2 := STR2 + ' ';
                    BYTES := BYTES + 1
                  end
         end;
      BYTES := BYTES + 1;
      {writeln('binaire : ',Bytes,' ',STR2);}
      STR3 := '';
      for I := 1 to BYTES do
         begin
            S := copy(STR2,1,8);
            MIRR(S);
            STR3 := STR3 + S + ' ';
            delete(STR2,1,9)
         end;
      {writeln('mirroite : ',STR3);}
      STR4 := '';
      for I := 1 to length(STR3) do
         if STR3[I] <> ' '
         then STR4 := STR4 + STR3[I];
      {writeln('espaces : ',STR4);}
      STR4 := copy(STR4,1,length(STR4)-BYTES mod 7);
      {writeln('zeros enleves : ',STR4);}
      STR5 := '';
      for I := 1 to length(STR4) do
         begin
            STR5 := STR5 + STR4[I];
            if I mod 7 = 0
            then STR5 := STR5 + ' '
         end;
      {writeln('bin 7 bits : ',STR5);}
      STR6 := '';
      while pos(' ',STR5) > 0 do
         begin
            S := copy(STR5,1,7);
            MIRR(S);
            DEC := bin2dec(S);
            CAR := chr(DEC);
            STR6 := STR6 + CAR;
            delete(STR5,1,8)
         end;
      {writeln(STR6);}
      assign(F,'SMSOUT.TXT');
      rewrite(F);
      writeln(F,STR6);
      close(F)
end.
```

Problème III – Hanoi 50 point

Démonstration par récurrence:

Avant de nous lancer dans la programmation réfléchissons sur le nombre de déplacements nécessaires.

Hauteur H	Déplacements	Nombre de déplacements N
0	-	0
1	1 -> 3	1
2	1 -> 2 1 -> 3 2 -> 3	3
3	1 -> 3 1 -> 2 3 -> 2 1 -> 3 2 -> 1 2 -> 3 1 -> 3	7
4	1 -> 2 1 -> 3 2 -> 3 1 -> 2 3 -> 1 3 -> 2 1 -> 2 1 -> 3 2 -> 3 2 -> 1 3 -> 1 2 -> 3 1 -> 2 1 -> 3 2 -> 3	15

etc.

Quelle relation peut-on trouver entre H et N?

H	N
0	0
1	1
2	3
3	7
4	15

Essayons avec: $N = 2^H - 1$

H	$2^H - 1$
0	0
1	1
2	3
3	7
4	15

$2^H - 1$ pourrait donc faire l'affaire. Faisons une démonstration par récurrence.

Début de la récurrence: (Induktionsanfang)
H = 0
pas de déplacements à faire
N = 0
et $2^0 - 1 = 1 - 1 = 0$

Pas de la récurrence: (Induktionsschritt)
Supposons que notre formule soit correcte pour H
et démontrons quelle reste valable pour H + 1

pour déplacer une tour de hauteur H + 1
de l'emplacement 1 vers l'emplacement 3, nous

1) déplaçons la tour de hauteur H
de l'emplacement 1 vers l'emplacement 2
ce qui nécessite $2^H - 1$ déplacements

2) déplaçons le disque le plus grand
de l'emplacement 1 vers l'emplacement 3
ce qui nécessite 1 déplacement

3) déplaçons la tour de hauteur H
de l'emplacement 2 vers l'emplacement 3
ce qui nécessite $2^H - 1$ déplacements

en tout:

$$2^H - 1 + 1 + 2^H - 1 = 2^H + 2^H - 1 = 2*2^H - 1 = 2^{H+1} - 1$$

C.Q.F.D.

Pour notre programme nous procédons de la même façon.

1) déplacement de H - 1 disques de 1 vers 2
2) déplacement de 1 disque de 1 vers 3
3) déplacement de H-1 disques de 2 vers 3

Cet exercice présente un exemple typique et fameux pour l'utilisation de la récursion.

```
program TOUR_DE_HANOI;

var H,N : integer;

procedure HANOI (H,START,AIDE,ARRIV : integer;
                 var N : integer);
begin
   if H > 0      { hauteur zéro: fin de la récursion }
   then begin
           { déplacement de H-1 disques de START vers AIDE }
           HANOI (H-1,START,ARRIV,AIDE,N);
           { déplacement du disque le plus grand de START vers
           ARRIV }
           writeln(START,' -> ',ARRIV);
           N := N + 1;
           { déplacement de H-1 disques de AIDE vers ARRIV }
           HANOI (H-1,AIDE,START,ARRIV,N)
        end
end;

begin
  readln(H);
  N := 0;
  HANOI(H,1,2,3,N);
  writeln(N);
  readln
end.
```

b) Solutions modèles en C

Problème I – Shuttle puzzle (Frank Mousset) 40 points

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

vector<string> moves;

void ml(string& s, int n) {
    s[n-1] = s[n];
    s[n] = '_';
    moves.push_back(s);
}

void mr(string& s, int n) {
    s[n+1] = s[n];
    s[n] = '_';
    moves.push_back(s);
}

void jl(string& s, int n) {
    s[n-2] = s[n];
    s[n] = '_';
    moves.push_back(s);
}

void jr(string& s, int n) {
    s[n+2] = s[n];
    s[n] = '_';
    moves.push_back(s);
}

void trace() {
    for (int i = 0; i < moves.size(); i++) cout << moves[i] <<
endl;
}

void alljl(string& s, int n, char color) {
    for (int i = n; i < s.length(); i++) {
        if (color == s[i]) jl(s, i);
    }
}

void alljr(string& s, int n, char color) {
    for (int i = n; i >= 0; i--) {
        if (color == s[i]) jr(s, i);
    }
}
```

```
string swapnb(string s) {
    for (int i = 0; i < s.length(); i++) {
        if (s[i] == 'N') s[i] = 'B';
        else if (s[i] == 'B') s[i] = 'N';
    }
    return s;
}

string revstr(string s) {
    string r = "";
    for (int i = s.length()-1; i >= 0; i--) {
        r = r + s[i];
    }
    return r;
}

void do_magic(string& s, int m) {
    int pos_;
    for (int i = 0; i < s.length(); i++)
        if (s[i] == '_') pos_ = i;
    if (pos_ < s.length()/2) {
        ml(s, pos_+1);
        alljl(s, pos_+1, s[pos_]);
    } else {
        mr(s, pos_-1);
        alljr(s, pos_-1, s[pos_]);
    }
}

int main(void) {
    int p, nmoves, movesize;
    cin >> p;
    string s;
    s = "_";
    for (int i = 0; i < p; i++) {
        s = s + "BN";
    }
    moves.push_back(s);
    char m = s[s.length()/2];
    while (m != '_') {
        do_magic(s, m);
        m = s[s.length()/2];
    }
    nmoves = 2*moves.size()+p-1;
    movesize = moves.size();
    vector<string> allmoves(nmoves, "___");
    for (int i = 0; i < movesize; i++) {
        allmoves[i] = moves[movesize-i-1];
        allmoves[nmoves-i-1] = revstr(moves[movesize-i-1]);
    }
    s = moves[0];
    moves.clear();
    alljl(s, 0, 'N');
    for (int i = 0; i < moves.size(); i++) {
        allmoves[movesize+i] = moves[i];
    }
```

```
    for (int i = 0; i < allmoves.size(); i++) {
        cout << allmoves[i] << endl;
    }
    cout << nmoves-1 << endl;
    getchar();
    getchar();
}
```

Problème II - SMS 40 points

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define kCODEARRAYLEN    (3*85 + 1)   //   1 character:   2digits
+ space + 0x00
                                      // 85 chars in total

typedef unsigned char uchar;

void    HexToBin(char    *CodeArray,    uchar    *BinArray,    int
*BinArrayLen)
{
   char *Ptr;

   Ptr=CodeArray;
   *BinArrayLen=0;
   while((BinArray[(*BinArrayLen)++]=strtol     (Ptr,     &Ptr,
16))!=0);

   (*BinArrayLen)--;
}


void   BinToHex(uchar   *BinArray,   int   *BinArrayLen,   char
*CodeArray)
{
   char         *Ptr;
   int          i;
   unsigned int temp;

   Ptr=CodeArray;
   for (i=0; i<*BinArrayLen; i++)
   {
     temp= (uchar) BinArray[i];
     sprintf(Ptr, "%02x ",temp );
     Ptr+=3;
   }
}
```

```
uchar Mirror8Bits0(uchar b)
{
  char c;

  c  = ((b >> 1) & 0x08) | ((b << 1) & 0x10);
  c |= ((b >> 3) & 0x04) | ((b << 3) & 0x20);
  c |= ((b >> 5) & 0x02) | ((b << 5) & 0x40);
  c |= ((b >> 7) & 0x01) | ((b << 7) & 0x80);
  return(c);
}


void Mirror8Bits(uchar *BinArray, int BinArrayLen)
{
  int i;

  for (i=0; i<BinArrayLen; i++)
    BinArray[i]=Mirror8Bits0(BinArray[i]);
}


int Calc7BitChunks(uchar *BinArray, int BinArrayLen)
{
  uchar b;
  int   BitCount, Mask;

  b=BinArray[BinArrayLen-1];
  BitCount=BinArrayLen*8;
  if ((b & 0x01) == 0)
  {
    BitCount--;
    Mask=2;
    while((BitCount % 7!=0) && ((b & Mask) == 0))
    {
      Mask=Mask<<1;
      BitCount--;
    }
  }
  return(BitCount / 7);
}

void  ExpandAndMirrorArray(uchar  *BinArray,  int  BinArrayLen,
char *CodeArray, int CharCount)
{
  uchar TmpA[kCODEARRAYLEN*8],
        c;
  int   i,
        j;

  // Create array of bits whe
  for (i=0; i<BinArrayLen; i++)
  {
    c=BinArray[i];
    for (j=0;j<8;j++)
      TmpA[i*8+j]=(c & (0x80>>j))>0;
  }
```

```
// Copy bits to target array using 7 bit chunks.Bit 0x80 is 0
  for (i=0; i<CharCount; i++)
  {
    c=0;
    for (j=0; j<7; j++)
      c |= TmpA[i*7+j]<<j;
    CodeArray[i]= c;
  }
  CodeArray[CharCount]=0x00;
}


void DecodeSMS(char *CodeArray)
{
  uchar BinArray[kCODEARRAYLEN];
  int   BinArrayLen,
        CharCount;

  // Convert to Binary
  HexToBin(    CodeArray, BinArray,   &BinArrayLen);
  // Mirror byte by byte (8 bits)
  Mirror8Bits(  BinArray,  BinArrayLen);
  // Calc number of 7bit chunks
  CharCount=Calc7BitChunks(BinArray, BinArrayLen);
  // Expand Array to array of bytes with 7bit mirroring
  ExpandAndMirrorArray(BinArray,    BinArrayLen,    CodeArray,
CharCount);
}


void main(void)
{
  FILE *f;
  char  CodeArray[kCODEARRAYLEN];

  f=fopen("SMSIN.TXT", "r");
  fgets(CodeArray, kCODEARRAYLEN,f);
  fclose(f);

  DecodeSMS(CodeArray);

  f=fopen("SMSOUT.TXT", "w");
  fputs(CodeArray, f);
  fclose(f);
}
```

Problème III - Hanoï (Frank Mousset) 50 points

```
#include <iostream>

using namespace std;

static int nmoves;

void move(int n, int from, int to) {
    int t;
    if (n == 1) {
        cout << from << " -> " << to << endl;
        nmoves++;
    } else {
        t = 6 / (from * to);
        move(n-1, from, t);
        cout << from << " -> " << to << endl;
        nmoves++;
        move(n-1, t, to);
    }
}

int main(void) {
    int n;
    cin >> n;
    if (n < 1) {
        cout << "Veuillez choisir n positif." << endl;
        getchar();
        getchar();
        return 1;
    }
    nmoves = 0;
    move(n, 1, 3);
    cout << nmoves << endl;
    getchar();
    getchar();
    return 0;
}
```