Devoir en classe d'Informatique II,2

50 minutes – 30 points

Exercice 1

Présentez l'algorithme de la *multiplication de deux polynômes.* [10 p.]

```
function produit(a, b : poly) : poly;
var
  i, j : integer;
  p : poly;
begin
        if ((a.d = 0) and (a.c[0] = 0)) or ((b.d = 0) and (b.c[0] = 0)) then
    begin
      p.d := 0;
      p.c[0] := 0;
    end
else
  begin
    p.d := (a.d + b.d);
    for i:= 0 to p.d do
      p.c[i] := 0;
    for i := 0 to a.d do
      for j := 0 to b.d do
        p.c[i+j] := p.c[i+j] + (a.c[i] * b.c[j]);
  end;
  result := p;
end;
```

Exercice 2

Ecrivez une procédure itérative qui trie une listbox, contenant des nombres naturels, de manière à ce que tous les nombres pairs se trouvent au début et tous les nombres impairs se trouvent à la fin. Utilisez pour cela ou bien le principe du tri par insertion ou bien le principe du tri par sélection. (On suppose l'existence de la procédure *swap*). [10 p.]

Remarque : L'ordre relatif des nombres pairs ou impairs n'est pas à considérer !

Exemple :

8	9	6	0	1	7	9	4

8	6	0	4	9	1	7	9

Procédure basée sur le principe du tri par sélection:

```
function findEvenIndex(list : TListBox; i : integer) : integer;
var
  j : integer;
begin
  result := i;
  for j := i+1 to list.Items.Count -1 do
    if (StrToInt(list.Items[j]) mod 2 = 0) then
      result := j;
end;

procedure selectionSort(list : TListBox);
var
  i: integer;
begin
  for i := 0 to list.Items.Count - 2 do
    swap(list, i, findEvenIndex(list, i));
end;
```

Procédure basée sur le principe du tri par insertion:

```
procedure insertIntoPosition(list : TListBox; i : integer);
var
  j : integer;
  value : string;
begin
  value := list.Items[i];
  j := i;
  while (j > 0) AND (StrToInt(list.Items[j-1]) mod 2 = 1) do
    begin
      list.Items[j] := list.Items[j-1];
      j := j - 1;
    end;
  list.items[j] := value;
end;

procedure insertionSort(list : TListBox);
var
  i : integer;
begin
  for i := 1 to list.Items.Count - 1 do
    insertIntoPosition(list, i);
end;
```

Exercice 3

On donne la procédure *func* définie par :

```
function swap(s : string; i,j : integer) : string;
var
  tmp : char;
begin
  tmp := s[i];
  s[i] := s[j];
  s[j] := tmp;
  result := s;
end;

procedure func(s : string; p : integer);
var
  i : integer;
begin
  if (p = Length(s)) then
    writeln(s)
  else
    for i := p to length(s) do
      func(swap(s, p, i), p+1);
end;
```

1. Déterminez à l'aide d'un exemple d'exécution : func('cat', 1) [8 p.]

func('cat', 1) appelle:

- *func('cat', 2)*
- *func('act', 2)*
- *func('tac', 2)*

func('cat', 2) appelle:

- *func('cat', 3)*
- *func('cta', 3)*

func('cat', 3) affiche 'cat'

func('cta', 3) affiche 'cta'

func('act', 2) appelle:

- *func('act', 3)*
- *func('atc', 3)*

func('act', 3) affiche 'act'

func('atc', 3) affiche 'atc'

func('tac', 2) appelle:

- *func('tac', 3)*
- *func('tca', 3)*

func('tac', 3) affiche 'tac'

func('tca', 3) affiche 'tca'

2. Expliquez en une seule phrase ce que fait cette procédure. [2 p.]

La procédure func affiche toutes les anagrammes du paramètre d'appel, c.-à-d. toutes les permutations des caractères du string.

Bonus

Dans l'exercice 2, écrivez les deux procédures (principe du tri par insertion et tri par sélection). [4 p.]