Épreuve d'informatique de l'X - 2006 - PSI

Alliance d'entreprises

Question 1 : On rappelle que n est supposé être une constante du programme.

```
1 somme := proc(c::array,a::array)
2 local i, res;
3 global n;
4 res := 0;
5 for i from 0 to n-1 do
6   if a[i] then res := res + c[i] fi;
7 od;
8 return(res);
9 end;
```

Question 2 :

```
1 estReussie := proc(c::array, a::array, Obj::integer)
2    return(evalb(somme(c,a)>=Obj));
3 end;
```

Question 3 : Pour vérifier qu'une alliance est stable, il faut de vérifier que l'alliance est réussie (ligne 4) et que l'objectif n'est pas réalisé dès que l'on enlève un élément de l'alliance. Le calcul de la somme correspondante se fait en retirant du chiffre d'affaire de l'alliance le chiffre d'affaire de l'entreprise (test de la ligne 6). Si le résultat est supérieur à l'objectif, l'alliance n'est pas stable (ligne 6).

```
1 estStable := proc(c,a,Obj)
2 local sommeAlliance, res, i;
3 sommeAlliance := somme(c,a);
4 res := estReussie(c,a,Obj);
5 for i from 0 to n-1 do
6    if a[i] and (sommeAlliance - c[i] >= Obj) then res := false fi;
7 od;
8 return(res)
9 end;
```

Question 4 : Une alliance a plus de chance d'être stable si les chiffres d'affaire des sociétés qui la composent sont importants. Il suffit d'afficher les numéros des entreprises en partant de la fin jusqu'à ce que leur somme dépasse l'objectif. On suppose que l'objectif est réalisable.

```
1  allianceMin := proc(c::array, Obj::integer)
2  local i, s;
3  i := n-1: s := 0:
4  while s < Obj do
5    printf("%d ", i):
6    s := s + c[i]:
7    i := i-1:
8  od:
9  return()
10 end:
```

Question 5 : On balaye le tableau a tant qu'il y a des « `vrai` » pour atteindre la position du premier « `faux` ». Dans le même temps on met « `faux` » à la place de « `vrai` » dans les cases visitées. À la fin de la boucle `while` (lignes 4 à 7) la variable i correspond à l'indice de la première case non visitée. Si i est différent de n, l'opération est l'opération est possible, on met « `vrai` » dans la i-ème case et retourne « `vrai` » (ligne 10). Sinon, l'opération n'est pas possible, on retourne « `faux` » (ligne 9).

```
1  suivantDe := proc(a :: array)
2  local i;
3  i:=0;
4  while  i < n and a[i] do
5    a[i] := false;
6    i := i+1;
7  od;
8  if i = n and a[n-1] = false
9           then return(false)
10          else a[i] := true; return(true)
11 fi;
12 end;
```

Question 6 :

```
1  imprimer := proc(a :: array)
2  local i;
3    for i from 0 to n-1 do
4      if a[i] then printf("%d",i) fi;
5    od;
6  printf("\n");
7  end;
```

Question 7 :

```
1  imprimerStables := proc(c::array,Obj::integer)
2  local i, a;
3  a := array(0..n-1);
4  for i from 0 to n-1 do
5      a[i] := false;
6  od;
7  while suivantDe(a) do
8    if estStable(c,a,Obj) then imprimer(a) fi
9  od;
10 end;
```

Pour l'ordre de grandeur du nombre d'opérations, la boucle des lignes 7 à 9 étant parcourue autant de fois qu'il y a d'alliances possibles, on aura 2^n appels à la fonction `estStable` dont le coût est en $O(n)$ et au plus 2^n appels à la fonction `imprimer` dont le coût est aussi en $O(n)$. Le nombre d'opérations effectuées sera de l'ordre de $O(n.2^n)$.

Question 8 :

```
1  cumuls := proc(c::array,t::array)
2  local i;
3  t[0] := c[0];
4  for i from 1 to n-1 do
5      t[i] := c[i] + t[i-1];
6  od;
7  end;
```

Question 9 : Dans cette question, on suit les consignes de l'énoncé mais la ligne 12 est incorrecte car en Maple, on n'a pas le droit de modifier les valeurs des arguments de la fonction.

On suppose qu'il nous reste à réaliser l'objectif Obj. On complète notre sélection d'entreprises jusqu'à ce que l'objectif soit réalisé. À chaque fois que l'on sélectionne une entreprise, on modifie le tableau a (ligne 11) et l'objectif à réaliser (ligne 12). On retourne enfin le numéro de la dernière entreprise ajoutée (ligne 14).

```
1  completer := proc(t::array, c::array, a::array,
2                                  Obj::integer, k::integer)
3  local i;
4  global n;
5  i := k;
6  while Obj >=0 do
7    i := 0;
8    while t[i] < Obj do
9      i := i+1
10   od;
11   a[i] := true;
12   Obj := Obj - c[i];
13 od;
14 return(i);
15 end;
```

Question 10 : On commence par rechercher le ℓ de l'énoncé (ligne 6). Si $k = i + l + 1$ est égal à n, on s'arrête en retournant la valeur n (ligne 9), sinon on enlève les entreprises dont les numéros sont compris entre i et $i + l$ de notre sélection et on rajoute leur chiffre d'affaire à l'objectif (ligne 12). Puis on rajoute l'entreprise k à notre sélection et on retranche son chiffre d'affaire de l'objectif (ligne 14). Enfin, on retourne la valeur k.

```
1  prelude := proc(t::array, c::array, a::array,
2                                  Obj::integer, i::integer)
3  local l, j, k;
4  global n;
5  l := 0;
6  while i+l+1 < n and a[i+l+1] do l := l+1 od;
7  k := i+l+1;
8  if k = n
9     then return(n);
10    else
11       for j from i to k-1 do
12            a[j] := false; Obj := Obj + c[j]
13       od;
14       a[k] := true; Obj := Obj - c[k];
15       return(k)
16 fi;
17 end;
```

Question 11 : On commence par créer et calculer le tableau des cumuls (lignes 4-5) puis on crée et on initialise le tableau des sélections (lignes 6-7). On vérifie que l'objectif est bien réalisable (ligne 8). On repère le plus petit des k tel que $t[k] \geqslant Obj$ (ligne 10). On sélectionne l'entreprise k (ligne 11), on diminue l'objectif à atteindre du chiffre d'affaire de l'entreprise k (ligne 12) puis on complète notre sélection pour atteindre notre nouvel objectif (ligne 13). La boucle des lignes 14-18, permet de calculer et d'afficher les sélections successives en suivant l'algorithme décrit par l'énoncé.

```
1  imprimerAStables1 := proc(c::array, Obj::integer)
2  local t, a, i, k;
3  global n;
4  t := array(0..n-1);
5  cumuls(c,t);
6  a := array(0..n-1);
7  for i from 0 to n-1 do a[i] := false; od;
8  if t[n-1] < Obj then return("Erreur : objectif irréalisable"); fi;
9  k := 0;
10 while k+1 < n and t[k] < Obj do k := k+1 od;
11 a[k] := true;
12 Obj := Obj - c[k];
13 i := completer(t,c,a,Obj,k);
14 while k < n do
15    imprimer(a);
16    k := prelude(t,c,a,Obj,i);
17    if k < n then i := completer(t,c,a,Obj,k) fi;
18 od;
19 end;
```