USTL - Licence ST-A 1ère année 2005-2006

Algorithmes, Programmation Impérative 1

EXAMEN juin 2006

Durée : 2 heures — Calculatrices et Documents interdits

Exercice 1 : *Connaissance et compréhension du langage* PASCAL.[DURÉE : 20 MINUTES]

On suppose, dans cette question, déclarées les procédures, fonctions et variables suivantes :

```
procedure p( const x : CARDINAL; out S:STRING);
function  f(const s :STRING):STRING;

var x,y : CARDINAL;
    z,t : STRING;
```

Q.1 Dans ce contexte, quelles sont les instructions invalides parmi celles ci-dessous ? Précisez pour quelles raisons !

1. `p(x,z);`
2. `x:=f(z);`
3. `if f(z)=z then p(3,z);`
4. `f(p(x,z));`
5. `x:=p(y,z);`
6. `p(2,'toto');`
7. `p(2,z);`
8. `z:=f(z+t);`
9. `p(x,f(z));`
10. `p(t,y);`
11. `f(t);`
12. `p(x,z+t);`

Q.2 On suppose, dans cette question, déclarées la procédure et les variables suivantes :

```
procedure q(const x : INTEGER ;
            out   y : INTEGER ;
            var   z : INTEGER );
var t : INTEGER;
begin
    t:= x+z;
    z:=2*z;
    y:= z+t;
end {q};
var x,y,z : INTEGER;
```

En supposant les variables initialisées par *{x:=1; y:=2; z:=3}* indiquez la valeur de ces variables après chacune des instructions suivantes ? (Dans chacune des sous-questions, on supposera que préalablement on a *{x:=1; y:=2; z:=3}*.)

1. `q(1,x,y);`
2. `q(x,y,z);`
3. `q(x+y+z,z,y);`

Exercice 2 : *Tableau d'éléments ordonnés.*[DURÉE : 40 MINUTES]

```
type ELEMENT = ... // un type pour lequel <= est défini
     INDICE  = ... // un intervalle entier
     TABLEAU = array[INDICE] of ELEMENT;
```

Q.1 Codez en PASCAL un prédicat `est_trie` qui permet de tester si un tableau `t` est trié.

Q.2 Codez en PASCAL la fonction dont les spécifications sont les suivantes :

```
// retourne le plus grand indice i
// tel que  la tranche
// t[a..i] soit triée
function ind_max(const a : INDICE; const T : TABLEAU):INDICE;
```

On suppose implantées la procédure `fusion` vue en cours, et la fonction `ind_max`. On rappelle la spécification de la procédure `fusion`

```
fusion ( t , a , b , c )
Données : un tableau t, trois indices a, b, c
CU      : les tranches t [a .. b] et
          t [b+1 .. c] sont triées
But     : trier la tranche t[a .. c]
```

On donne le code de la procédure `biduler` :

```
procedure biduler(var t : TABLEAU);
var a,b,m:INDICE;
begin
  a:=low(t);
  m:=ind_max(a,t);
  while m<high(t) do
  begin
    b:=ind_max(succ(m),t);
    fusion(t,a,m,b);
    m:=b;
  end {while};
end {biduler};
```

Q.3 On considère le tableau `t`

i	1	2	3	4	5	6	7
t[i]	4	3	7	1	2	5	3

Détaillez l'exécution de l'instruction `biduler(t)`. On donnera à chaque étape du `tant que`, la valeur de `m` et la valeur de `t`

Q.4 D'une manière générale, à quoi sert la procédure `biduler` ? Justifiez.

Exercice 3 : *Masse critique.*[DURÉE : 1 HEURE]

Le but de ce problème est de réaliser certains éléments nécessaires pour programmer le jeu des masses critiques.

C'est un jeu à deux joueurs, qui se joue avec une grille rectangulaire de $n \times m$ cases . Chaque joueur doit choisir une couleur. Chaque joueur dispose de bâtonnets de sa couleur. Chaque case peut contenir des bâtonnets. Tous les bâtonnets contenus dans une case doivent être de la même couleur.

1. Préparation du jeu

   Au début, la grille est vide, puis chaque joueur pose un bâtonnet de sa couleur, dans des coins diagonalement opposés. Un joueur est désigné aléatoirement pour commencer.

2. Réglement
   - On dit qu'une case est *neutre*, lorsqu'elle est vide.
   - On dit qu'une case *appartient* à un joueur lorsqu'elle contient au moins un bâtonnet de sa couleur.
   - La *capacité* d'une case est le nombre de cases adjacentes, ainsi la capacité d'un coin est 2, celle d'une case sur un bord qui n'est pas un coin est 3, enfin la capacité d'une case qui n'est pas sur le bord est 4.
   - Lorsqu'une case c contient un nombre de bâtonnets supérieur ou égal à sa capacité, elle *explose*.
   - Lorsqu'une case c explose, les bâtonnets qu'elle contient sont répartis équitablement dans les cases adjacentes de la case c, le reste est laissé dans la case c.
   - Lorsqu'une case reçoit des bâtonnets, tous les bâtonnets qu'elle contient sont remplacés par autant de bâtonnets de la couleur du joueur qui a joué.
   - Bien sûr, il peut y avoir des réactions en chaîne...

3. Déroulement du jeu

   À tour de rôle, chaque joueur doit poser un bâtonnet dans une case neutre, ou dans une case qui lui appartient.

4. Condition de victoire

   Un joueur gagne lorsque la couleur de l'autre joueur n'est plus représentée sur la grille.

Q.1 Déclarez un type `COULEUR` à deux valeurs.

Q.2 Définissez un type `CASES` permettant de représenter le nombre et la couleur des bâtonnets d'une case.

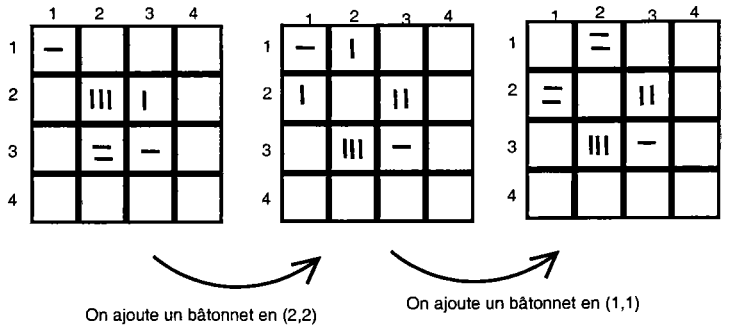

FIG. 1 – exemple d'explosion

Remarque : Dans les figures l'orientation des bâtonnets est utilisée pour représenter la couleur (les sujets sont en noir et blanc)...

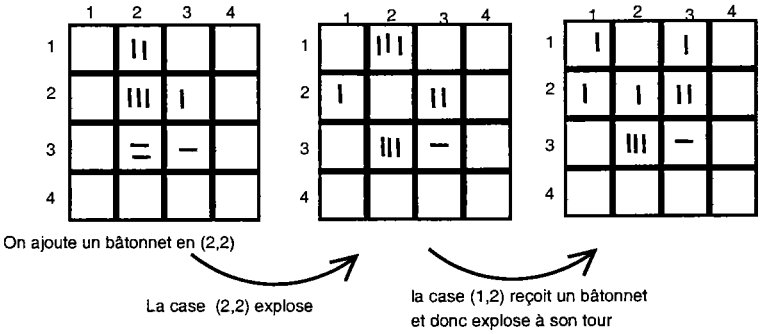

FIG. 2 – exemple de réaction en chaîne

Q.3 Codez en PASCAL une fonction nommée `est_neutre` paramétrée par `c` de type `CASES` et dont le résultat est un booléen, qui vaut vrai si et seulement si la case est neutre.

le type `GRILLE` est défini par

```
const N = .... ; // un entier naturel représentant le nombre de lignes
      M = .... ; // un entier naturel représentant le nombre de colonnes
type IND_LIG = 1..N;
     IND_COL = 1..M;
type GRILLE = array[IND_LIG,IND_COL] of CASES;
```

Q.4 Codez en PASCAL une procédure `vider_grille` qui permet de mettre à 0 le nombre de bâtonnets de chacune des cases de la grille passée en paramètre.

Q.5 Codez en PASCAL une fonction `est_case_valide` paramétrée par deux entiers `i` et `j` , dont le résultat est vrai si et seulement si `i` est compris entre 1 et N, et `j` est compris entre 1 et M

Q.6 Codez en PASCAL une fonction `capacite` paramétrée par deux entiers
- `i` supposé compris entre 1 et N,
- et `j` supposé compris 1 et M

dont le résultat est le nombre de voisins de la case (i, j).

Q.7 Codez en PASCAL une procédure récursive dont les spécifications sont les suivantes :

```
procedure ajouter( var   g : GRILLE;
                   const i : IND_LIG;
                   const j : IND_COL;
                   const n : CARDINAL;
                   const c : COULEUR);
```

qui permet d'ajouter `n` bâtonnets de la couleur `c` dans la case de coordonnées `(i,j)` de la grille `g`. En faisant éventuellement exploser la case si le nombre de bâtonnets atteint la capacité.

Q.8 La procédure écrite précédemment s'arrête-t-elle dans tous les cas ?