Examen de Programmation Impérative avec Ada – 1ère session – Mai 2016

Auteur : Joëlle Cohen

durée 2h – sans document – 4 pages

justifiez vos réponses

barême en annexe

I. Tableaux

On considère le paquetage générique dont le fichier de spécification PG.ads est

```
generic type item is private;
package PG is
nb : constant integer := 999 ;
subtype intervalle is integer range 0..nb;
type tabG is array(intervalle) of item;
procedure init(t: out tabG);
function nbItem(t:tabG; e:item) return integer;
function inconnue(t:tabG) return item;
end P ;
```

La procédure `init(t)` permet d'**initialiser** `t`.

La fonction `nbItem(t:tabG;e:item)` renvoie le nombre d'occurrences de `e` dans le tableau `t`.

I.1.

Donnez la suite d'instructions permettant d'utiliser et d'initialiser un tableau t de type `tabG` contenant des entiers.

I.2.

Ecrire le code de la fonction `nbItem(t:tabG;e:item)`

I.3.

Voici le code de la fonction `inconnue`.

Expliquez ce que renvoie cette fonction.

```
1  function inconnue(t:tabG) return item is
2  k, n : integer;
3  begin
4  k := 0; n :=0;
5  for i in t'range loop
6          if nbItem(t,t(i))>=nbItem(t,t(k))
7          then k:=i;
8          elsif nbItem(t,t(i))>nbItem(t,t(n))
9          then n:=i;
10 end loop;
11 return t(n);
12 end inconnue;
```

II. Pile

On a un paquetage générique `pile` pour le type générique `item`.

II.1.

Que fait la procédure suivante ? :

```
1  procedure P(p : in out pile ; e : in item) is
2  pTemp : pile;  trouve : boolean := false;
3  begin
4  init(pTemp);
5  while (not vide(p) and then sommet(p)/= e) loop
6    ajouter(pTemp, sommet(p));
7    supprimer(p);
8  end loop;
9  if not vide(p) then trouve := true; end if;
10 if trouve then
11   supprimer(p);
12 end if;
13 while (not vide(ptemp)) loop
14     ajouter(p, sommet(pTemp));
15     supprimer(pTemp);
16 end loop;
17 ajouter(p,e);
18 end P;
```

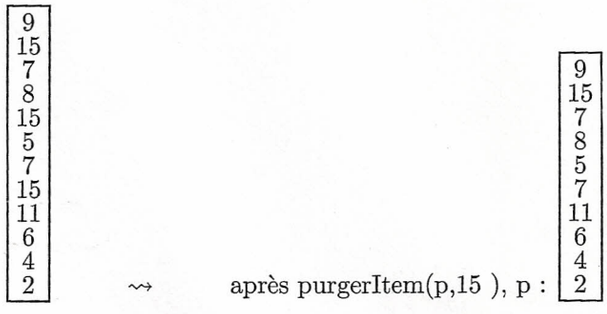
II.2.

Ecrire une procédure de prototype

procedure purgerItem(p : **in out** pile ; e : **in** item)

qui supprime de la pile p toutes les occurrences de e sauf la première en partant du sommet, et sans modifier l'ordre relatif des autres éléments de la pile.

Par exemple,



III. File

On a un paquetage générique `file` pour le type générique `item`.

Ecrire une fonction de prototype

```
function rang(f:file ; x : item) return integer;
```

telle que

- si x est dans f, la fonction `rang(f,x)` renvoie le nombre de valeurs que l'on doit défiler de f pour que x soit le premier élément de f
- si x n'est pas dans f, la fonction `rang(f,x)` renvoie -1

Par exemple si, en partant d'une file d'entiers vide, on enfile successivement 2, 7, 9, 5, 3, 2, 5, 7, alors rang(f,2) renvoie 0 — rang(f,5) renvoie 3 — rang(f,4) renvoie -1.

IV. Liste

```
type cell;
type liste is access cell;
type cell is record
v:integer;
suiv : liste;
end record;
```

IV.1.

Ecrire une fonction récursive de prototype

```
function nbOccurrence (l:liste ; x : integer) return integer;
```

qui renvoie le nombre d'occurrences de `x` dans la liste `l`.

IV.2.

Que fait la procédure suivante

```
1  procedure inconnue2(l : in out liste) is
2  temp : liste;
3  sauv : liste;
4  begin
5  if l/= null  and then l.suiv /=null
6  then
7  temp:=l.suiv;
8  l.suiv := null;
9  while(temp/=null) loop
10 sauv := temp.suiv;
11 temp.suiv:=l;
12 l:=temp;
13 temp:=sauv;
14 end loop;
15 end inconnue2;
```

Annexe

contenu du paquetage `pile` pour le type générique `item`:

```
procedure init (p : out pile) ;
procedure ajouter (p : in out pile ; e :in item ) ; -- empiler
procedure supprimer (p : in out pile) ; -- dépiler
function sommet (p : pile) return integer ;
function vide (p : pile) return boolean ;
```

contenu du paquetage `file` pour le type générique `item`:

```
procedure init (f : out file) ;
procedure ajouter (f : in out file ; e :in item ) ; -- enfiler
procedure supprimer (f : in out file) ; -- défiler
function premier (f : file) return integer ;
function vide (f : file) return boolean ;
```

Barême provisoire sur 40

I : 10 points — **II** : 12 points — **III** : 7 points — **IV** : 11 points