SCIENCES SORBONNE UNIVERSITÉ

Votre numéro d'anonymat : | | | |

Éléments de programmation 2– 1I002

Examen du 18 juin 2019
2h
Aucun document n'est autorisé.

Les calculatrices et autres appareils électroniques sont interdits. Les téléphones mobiles doivent être éteints et rangés dans les sacs.

Le barème sur 66 points (12 questions) n'a qu'une valeur indicative.

Les questions sont indépendantes. Vous pouvez utiliser une fonction que vous n'avez pas programmée.

L'objectif est de modéliser de manière simple le comportement des fourmis pour l'exploitation des ressources alimentaires. Au départ, les fourmis parcourent au hasard leur environnement en partant de leur nid puis, si une fourmi découvre une source de nourriture, elle rentre plus ou moins directement au nid (trajet en sens retour), en laissant sur son chemin une piste de phéromones. Ces phéromones étant attractives, les fourmis passant à proximité vont avoir tendance à suivre cette piste.

Le monde des fourmis sera représenté par un tableau à deux dimensions `int monde[NB_LG][NB_COL]` où `NB_LG` et `NB_COL` sont deux constantes définies par `#define` représentant respectivement le nombre de lignes et le nombre de colonnes du tableau.
L'expression `monde[i][j]` permet d'accéder (en lecture ou en écriture) à la valeur se trouvant à l'intersection de la ligne `i` ($0 \leq$ `i` $<$ `NB_LG`) et de la colonne `j` ($0 \leq$ `j` $<$ `NB_COL`) du tableau `monde`.
Le passage en argument d'un tableau à deux dimensions se fera ici avec ses dimensions entre les crochets des deux dimensions, par exemple `void initMonde(int monde[NB_LG][NB_COL])`.

Première partie : tirages aléatoires, tableaux 1D

Question 1 (6 points)

Ecrire la fonction

```
int *somCumulee(int t[], int n)
```

qui retourne un tableau d'entiers contenant la somme cumulée des valeurs contenues dans le tableau `t` de taille `n`. La case `i` du tableau retourné contiendra la somme du contenu des cases 0 à `i` du tableau `t`. Par exemple, pour `t={1,2,0,3}`, la fonction retournera un tableau contenant `{1,3,3,6}`.

Réponse :

```
int *somCumulee(int t[], int n){
```

```
}
```

Question 2 (6 points)

Ecrire la fonction

```
int tirerSelonScore(int t[], int n)
```

qui retourne un entier (indice) entre 0 et n (exclu), tiré au hasard de manière non équiprobable en fonction du tableau t de taille `n`. Vous utiliserez un tableau des sommes cumulées nomé `tCumul`, calculé avec la fonction précédente. Vous tirerez ensuite aléatoirement une valeur `a` comprise entre 0 (inclus) et la somme de toutes les valeurs de `t` (exclue) puis vous calculerez l'indice de la case de `tCumul` contenant la première valeur strictement supérieure à `a`. Vous retournerez cet indice.

Par exemple, avec le tableau `t={1,2,0,3}`, si vous tirez aléatoirement 2, vous devrez retourner 1 car la case d'indice 1 du tableau des sommes cumulées `{1,3,3,6}` contient 3 qui est la première valeur supérieure à 2. Attention à éviter les fuites mémoire.

Cette fonction permet donc de tirer aléatoirement selon une distribution quelconque : avec le tableau `t={1,2,0,3}`, il y a 1 chance sur 6 de tirer l'indice 0, 2/6 de tirer 1, 0/6 de tirer 2 et 3/6 de tirer 3. Vous remarquerez qu'il n'est pas possible avec cette procédure de tirer l'indice 2 car cette case contient 0.

Réponse :

```
int tirerSelonScore(int t[], int n){

}
```

Deuxième partie : le monde des fourmis, tableaux 2D

Le monde dans lequel évoluent les fourmis sera représenté par un tableau statique à deux dimensions `int monde[NB_LG][NB_COL]` déclaré dans la fonction `main`. Chaque case contiendra la quantité (entière) de phéromones. Les cases du pourtour contiendront 0 phéromone (il est alors interdit et impossible d'y aller pour les fourmis) et les autres cases contiendront (au départ) 1 phéromone. Les fourmis déposeront plus tard d'autres phéromones dans les cases. Le nid des fourmis est en `(1,1)` et la nourriture est en `(NB_LG - 2, NB_COL - 2)`.

Question 3 (6 points)
Ecrire la fonction

```
void initMonde(int monde[NB_LG][NB_COL])
```

qui initialise toutes les cases du `monde` à 1 sauf celles du pourtour qui sont mises à 0.

Réponse :

```
void initMonde(int monde[NB_LG][NB_COL]){

}
```

Troisième partie : gestion des fourmis, listes

Les fourmis sont représentées par la structure `tyFourmi` donnée ci-dessous. Elles seront stockées sous forme d'une liste.

```
typedef struct _tyFourmi{
   int i; // coordonnee i (ligne) de la fourmi
   int j; // coordonnee j (colonne) de la fourmi
   char sens;  //sens du trajet ('A' pour aller ou 'R' pour retour)
   int energie; //energie de la fourmi
   struct _tyFourmi *suiv; //pointeur vers la fourmi suivante
} tyFourmi;
```

Une fourmi est dans le sens aller lorsqu'elle cherche de la nourriture, et dans le sens retour lorsqu'elle en a trouvé et qu'elle rentre au nid pour l'y déposer.

Question 4 (6 points)
Ecrire la fonction

```
tyFourmi *insererEnTete(tyFourmi *listeFourmis)
```

qui renvoie la liste obtenue en créant une nouvelle fourmi et en l'ajoutant en tête de la liste `listeFourmis`. La fourmi sera placée dans le nid (en 1,1), son énergie sera un entier tiré aléatoirement entre 1 et la quantité d'énergie maximale (incluse) définie par la constante `MAX_E` (définie par **`#define`**), elle sera dans le sens aller (`'A'`).

Réponse :

```
tyFourmi *insererEnTete(tyFourmi *listeFourmis){

}
```

Le déplacement d'une fourmi dans une des 4 cases l'entourant (haut, gauche, bas, droite) dépend de la quantité de phéromones sur ces cases. Si la fourmi est en sens aller, toutes les directions sont possibles mais si la fourmi est en sens retour, elle ne peut que se diriger vers le nid donc vers le haut en diminuant i ou vers la gauche en diminuant j.

Question 5 (3 points)

Ecrire la fonction

```
void copiePheromone(int monde[NB_LG][NB_COL], tyFourmi fourmi, int tDir[])
```

qui recopie dans `tDir`, tableau de 4 entiers, les quantités de phéromones des 4 cases du monde autour de la fourmi dans l'ordre haut, gauche, bas, droite si elle est en sens aller, et seulement haut et gauche (les deux autres seront mises à 0) si la fourmi est en sens retour.

NB : On rappelle que la fourmi ne peut (par construction) se trouver au bord.

Réponse :

```
void copiePheromone(int monde[NB_LG][NB_COL], tyFourmi fourmi, int tDir[]){
}
```

Question 6 (6 points)

Ecrire la fonction

```
void bougerUneFourmi(tyFourmi *unefourmi, int monde[NB_LG][NB_COL])
```

qui déplace la fourmi au hasard en fonction des quantités de phéromones sur les cases alentours (en haut, à gauche, en bas, et à droite) et du sens du déplacement (aller ou retour). Par exemple, soit une fourmi entourée de cases contenant des phéromones :

```
  1
2 * 3
  5
```

En sens aller, la probabilité qu'elle aille en haut est de 1/11, à gauche de 2/11, en bas de 5/11 et à droite de 3/11. En sens retour, ces mêmes probabilités sont : 1/3, 2/3, 0 et 0.

Les étapes à réaliser dans cette fonction sont :

— recopier les phéromones (cf. fonction `copie_pheromone`) ;
— tirer au hasard la "direction" de son déplacement (cf. fonction `tirerSelonScore`) ;
— déplacer la fourmi d'une case selon cette direction.

Réponse :

```
void bougerUneFourmi(tyFourmi *unefourmi, int monde[NB_LG][NB_COL]){

}
```

Question 7 (6 points)

Ecrire la fonction

```
int majFourmi(tyFourmi *uneFourmi, int monde[NB_LG][NB_COL])
```

qui met à jour les champs de la fourmi dans l'ordre indiqué :

1. diminue l'energie de la fourmi de 1 ;
2. si la fourmi atteint la nourriture (case `(NB_LG - 2, NB_COL - 2)`), elle passe en sens retour ;
3. si la fourmi est en sens retour, elle dépose une phéromone sur la case où elle est ;
4. si la fourmi atteint le nid (case (1,1)), son énergie est remise au maximum `MAX_E` et elle passe en sens aller.

La fonction renvoie 1 (qui correspond à la quantité de nourriture ramenée) si la fourmi atteint le nid alors qu'elle est en sens retour, sinon elle renvoie 0 (pas de nourriture ramenée).

Cette fonction est normalement appelée après un déplacement de la fourmi. Les champs `i` et `j` de la fourmi contiennent donc les coordonnées de la case atteinte.

Réponse :

```
int majFourmi(tyFourmi *uneFourmi, int monde[NB_LG][NB_COL]){

}
```

Question 8 (3 points)

Ecrire la fonction

```
int longueurListe(tyFourmi *pListe)
```

qui retourne le nombre d'éléments de la liste `pListe` donnée en argument.

Réponse :

```
int longueurListe(tyFourmi *pListe)){

}
```

Question 9 (6 points)

Ecrire la fonction

```
int bougerLesFourmis(tyFourmi *pListe, int monde[NB_LG][NB_COL])
```

qui permet de déplacer et de mettre à jour toutes les fourmis de la liste. Cette fonction retourne la quantité de nourriture ramenée par l'ensemble des fourmis de la liste.

Réponse :

```
int bougerLesFourmis(tyFourmi *pListe, int monde[NB_LG][NB_COL]){

}
```

Question 10 (6 points)

Ecrire la fonction

```
tyFourmi *supprimerUneFourmi(tyFourmi *pListe, tyFourmi *aSupprimer)
```

qui supprime de la liste `pListe` la fourmi pointée par `aSupprimer` et retourne la tête de la liste sans cette fourmi. Nous faisons l'hypothèse que la fourmi pointée par `aSupprimer` est présente dans la liste.

Réponse :

```
tyFourmi *supprimerUneFourmi(tyFourmi *pListe, tyFourmi *aSupprimer){

}
```

Question 11 (6 points)

Ecrire la fonction

```
tyFourmi *supprimerFourmisMortes(tyFourmi *pListe)
```

qui supprime de la liste toutes les fourmis dont l'énergie est inférieure ou égale à 0 et retourne la tête de la liste sans ces fourmis.

Réponse :

```
tyFourmi *supprimerFourmisMortes(tyFourmi *pListe){

}
```

Question 12 (6 points)

Ecrire la fonction `main` dans laquelle vous :

— déclarerez et initialiserez le monde des fourmis ;
— créerez un liste de 10 fourmis ;
— tant qu'il y a des fourmis :
  — déplacerez et mettrez à jour toutes les fourmis de la liste ;
  — supprimerez les fourmis d'énergie inférieure ou égale à 0 ;
— afficherez la quantité totale de nourriture ramenée après que toutes les fourmis soient mortes.

Réponse :

```
main{

}
```