UNIVERSITÉ TOULOUSE III PAUL SABATIER

L2 CUPGE 2018/2019

Contrôle Terminal

1h30 - Documents non autorisés

Nota : L'ordre des exercices n'importe pas

Partie 1 – divers (4 points)

1. Qu'est-ce qu'un type abstrait de données ? (1,5 pt)

```
#include <stdio.h>

void f ( int t[], int i, int n ){
      if (i >= 0 && i<n){
            t[i] = t[i]*t[i] ;
      f(t,i-1, n) ;
      }
}

int main () {
      int i, size, n ;
      int t[4] = {1, 2, 3, 4};
      size=4;
      n=2;
      f(t, n, size);
      printf("%d\n", n);
      for (i=0; i<size; i++)
            printf("%d\n", t[i]);
      return 0 ;
}
```

2. Après avoir compilé, quelles impressions sont effectuées par ce programme sur le terminal ? (2, 5 pts)

Partie 2 - Problème de Flavius Josèphe (5 points)

41 personnes sont disposées sur un cercle et on choisit l'origine O.

En commençant par O, on enlève une personne par une suivant le procédé suivant : on compte 1, 2 et la 3ème personne est supprimée (3ème cran). On itère ce procédé autour du cercle et on recommence jusqu'à ce qui ne reste plus qu'une seule personne.

La question : où se placer dans le cercle d'origine pour être le dernier survivant ?

On généralisera ce problème en utilisant N personnes et k crans et on souhaite programmer cet algorithme en C afin d'afficher la position finale. On utilisera pour cela une **liste chaînée circulaire.**

1) Définir la **cellule personne** (1 pt).

2) Fabriquer l'embryon de la liste avec un élément pointé par **debut,** puis faire des insertions successives pour fabriquer la liste complète des N cellules (2 pts).

3) Ecrire la boucle du programme où à chaque itération est enlevée une cellule et qui s'arrête quand il ne reste plus qu'une seule cellule (2 pts).

Partie 3 – en « Formes » … (4 pts)

Définir tout d'abord une « structure » en C **Forme** permettant de représenter une forme qui peut être un **rectangle** ou un **carré**.

Ecrire les fonctions suivantes :

1. `void creer_forme(Forme *f, <arguments à définir>)` qui permet d'initialiser une forme (2 pts)
2. `void additionne_forme(Forme *f, Forme *f_clone)` qui permet de doubler une forme : un rectangle donnera un rectangle d'une longueur (ou largeur) deux fois supérieure et un carré … un rectangle ! (2 pts)

Partie 4 – Maillons de chaîne … (7 pts)

Nous considérons ici des listes chaînées de nombres entiers signés. Elles sont référencées par l'adresse de leur premier maillon ou pointeur de tête.

1. **Création de la structure de liste.** Ecrire la structure en C permettant de gérer une liste chaînée (1 pt)
2. **Création d'une liste.** Ecrire un programme qui lit une suite de nombres à la console et crée la liste chaînée correspondante. (1 pt)
3. **Recherche d'un élément dans une liste.** On considère une liste représentée par son pointeur de tête. Écrire une fonction qui prend une valeur pour argument et renvoie l'adresse du maillon portant la valeur recherchée (1 pt)
4. **Ordonnancement de liste.** Ecrire une fonction qui ordonne une liste donnée en paramètre sans création de maillon (2 pts)
5. **Concaténation de deux listes.** On considère deux listes L1 et L2 non ordonnées. Écrire les instructions qui concatènent ces deux listes et renvoie une liste ordonnée. (2 pts)