

Dimanche 20 novembre 2016

Devoir surveillé (S3)
Module : Informatique
Durée : 2h00

Questions de cours (6 pts)

- Quels avantages offrent les listes chaînées par rapport aux tableaux ?
- Qu'est ce qu'une fonction récursive terminale.
- Quel est le mécanisme qui permet de s'assurer qu'une fonction récursive prendra fin ?
- Donnez un exemple du monde réel qui peut être modélisé par une PILE.
- Donner le résultat d'exécution des deux programmes suivants :

P1.c

```
#include <stdio.h>
int truc(int n, int v){
    if (n == 0)
        return v;
    return truc(n/10, v * 10 + (n % 10));
}

int main(){
    int n=6532, res;
    res = truc(n, 0);
    printf("%d", res);
    return 0;
}
```

P2.c

```
#include <stdio.h>
int syracuse(int n, int m){
    if (n == 0)
        return m;
    else
        if(n % 2 == 0)
            return syracuse(n - 1, m)/2;
        else
            return 3 * syracuse(n - 1, m) + 1;
}

int main(){
    int n, m, res;
    n = 4;
    m = 2;
    res = syracuse(n, m);
    printf("%d", res);
    return 0;
}
```

Exercice 1 (7 pts)

1. Etant données deux listes chaînées A et B d'entiers, écrire une fonction nommée *appartient* qui permet de vérifier si tout les éléments de la liste B appartiennent à la liste A. Cette fonction doit avoir comme arguments les deux listes et devra retourner 1 si tout les éléments de B appartiennent à A et 0 sinon. **On suppose que les deux listes ne peuvent pas contenir la même valeur plusieurs fois.**

2. Une liste chaînée est dite **ordonnée** si elle contient une suite ordonnée d'entiers, c'est-à-dire que la valeur de chaque élément de la liste doit être égale à la valeur de l'élément suivant moins un. Ecrire une fonction nommée *ordonner* qui permet d'ajouter des éléments dans une liste chaînée trié par ordre croissant afin quelle soit **ordonnée**. Cette fonction doit avoir comme arguments une liste chaînée trié par ordre croissant et de devra retourner la liste obtenue après, ajout des éléments. **La liste est supposée trier, ne faite pas de tri.**

Exemples :

- Si la liste contient les valeurs <2, 3, 5, 7>, les valeurs 4 et 6 doivent être ajoutées à la liste.
- Si la liste contient les valeurs <5, 8, 10, 12>, les valeurs 6, 7, 9 et 11 doivent être ajoutées à la liste.

3. Etant donné une file F et deux piles P et Q. La file F contient une suite d'entiers, P et Q sont initialement vides. Ecrire une fonction nommée *transferer* qui met dans P les entiers impairs de F et laisse dans F les entiers pairs. Notons que les éléments de F doivent être dans le même ordre qu'au début et les éléments de P dans un ordre inversé. Par exemple, si la file F contient les éléments [3,11,8,5,16,7,4] (**3 est le premier élément de F**), la file F sera [8, 16, 4] et la pile P sera [7,5,11,3]. (Utiliser les fonctions de manipulation des piles et des files sans donner leurs définitions).

Exercice 2 (3 pts)

Soit les deux programmes suivants:

```

P1.c
#include<stdio.h>
#include<stdlib.h>

struct element{
    int donnee;
    element * suivant;
};

element * operation(element * debut){
    element * ptmp, * tmp;

    ptmp = debut;
    tmp = debut->suivant;
    while(tmp != NULL){
        if(ptmp->donnee == tmp->donnee){
            ptmp->suivant = tmp->suivant;
            free(tmp);
            tmp = ptmp->suivant;
        }
        else{
            ptmp = tmp;
            tmp = tmp->suivant;
        }
    }
    return debut;
}

```

```

P2.c
#include<stdio.h>
#include<stdlib.h>

struct pile{
    int donnee;
    pile * precedent;
};

pile * mystere(pile *A, pile *B){
    pile * C;
    int x;

    C = NULL;
    while(est_vider(A)==0 && est_vider(B)== 0){
        if(sommet(A) <= sommet(B))
            x = depiler(&A);
        else
            x = depiler(&B);
        C = empiler(C, x);
    }
    while(est_vider(A) == 0){
        x = depiler(&A);
        C = empiler(C, x);
    }
    while(est_vider(B) == 0){
        x = depiler(&B);
        C = empiler(C, x);
    }
    return C;
}

```

1. Soit une liste chaînée L contenant les valeurs suivantes <7, 3, 3, 9, 6, 6, 6, 2>. Donnez le résultat d'exécution de la fonction **operation** pour la liste L.
2. Donnez la pile obtenue après exécution la fonction **mystere** pour les piles **A = [8, 7, 3, 2]** 2 est le sommet de la pile A et **B = [16, 12, 11, 6, 4]** 4 est le sommet de la pile B. (On suppose qu'on fait appel aux fonctions de manipulation des piles).

Exercice 3 (4 pts)

On se donne deux piles P et Q et une file F. La pile P contient une suite d'entiers, Q et F sont initialement vides. Ecrire une fonction nommée *rotation* qui permet d'effectuer la rotation de *n* éléments de la pile P, c'est-à-dire placer les n derniers éléments de la pile au début. Cette fonction doit avoir comme arguments la pile P et l'entier n et devra retourner la pile obtenue après rotation. (Utiliser les fonctions de manipulation des piles et des files sans donner leurs définitions).

Exemples :

Si la pile P contient les éléments [2,6,9,8,12] (**12 est le sommet de P**) et n = 2, la pile P sera [8,12,2,6,9]

Si la pile P contient les éléments [2,6,9,8,12] (**12 est le sommet de P**) et n = 3, la pile P sera [9,8,12,2,6]