Questionnaire d'examen+Corrigé

INF1101 Algorithmes et structure de données
Aut 99

Question 1 – Structure de données (2.0 points)

Dans le cours, nous avons vu différentes structures de données, et ainsi que les algorithmes permettant d'effectuer le retrait d'une donnée dans celle-ci. Dépendant de la structure de données et de l'algorithme associé, l'élément retiré sera :

- un élément quelconque,
- le plus petit élément,
- le dernier élément inséré,
- le plus grand élément,
- le premier élément inséré,
- l'élément du milieu.

Pour chaque structure de données qui suit : liste, pile, file, arbre en monceau, arbre binaire de recherche, identifier quel sera l'élément retiré parmi les choix ci-dessus.

Réponse à la question 1

Liste : retrait d'un élément quelconque

Pile : retrait du dernier arrivé

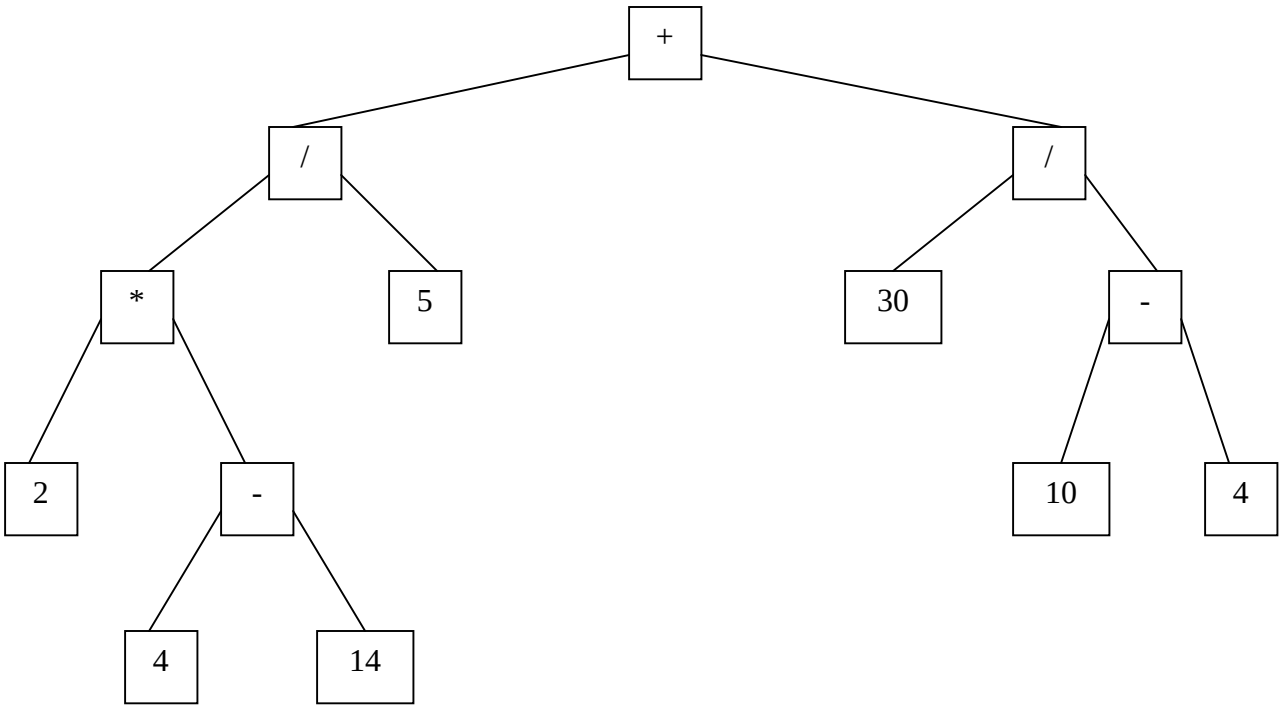
File : retrait du premier arrivé

Arbre en monceau : retrait du plus petit (ou plus grand élément)

Arbre binaire de recherche : retrait d'un élément quelconque

Question 2 -- Arbre binaire (2.5 points)

Soit l'arbre arithmétique suivant :



2.1 Donner l'expression obtenue en utilisant le parcours post-ordre (gauche-droite-père).

2.2 Évaluer l'expression obtenue en 2.1 à l'aide de l'algorithme de la page 425 des notes de cours.

2.3 Donner l'expression obtenue en utilisant le parcours pré-ordre (père-gauche-droite).

2.4 Donner l'expression obtenue en utilisant le parcours in-ordre (gauche-père-droite).

2.5 Évaluer l'expression obtenue en 2.4 en appliquant l'ordre de priorité des opérateurs arithmétiques.

Réponse à la question 2.1

2 4 14 - * 5 / 30 10 4 - / +

Réponse à la question 2.2

1

Réponse à la question 2.3

+ / * 2 - 4 14 5 / 30 - 10 4

Réponse à la question 2.4

2 * 4 - 14 / 5 + 30 / 10 - 4

Réponse à la question 2.5

4.2

Question 3 -- Monceau (3.5 points)

On crée un objet de la classe `Monceau`, dont les éléments ont été ajoutés à l'aide de la fonction membre `Ajouter` (cette fonction ajoute les éléments à la fin du tableau). Il en résulte que les éléments du tableau `Items` ne sont pas organisés en monceau. Écrire une fonction membre `OrganiserMonceau()` de la classe `Monceau` qui organisera les objets du tableau `Items` en Monceau. La fonction `OrganiserMonceau()` fera appel aux fonctions membres de la classe Monceau. On considère que le nœud courant est plus petit que ses descendants.

```
#ifndef MONCEAU
#define MONCEAU
#include <stdlib.h> // pour NULL
template <class T>
class Monceau
{
protected:
  unsigned int MaxTaille;
  unsigned int NbNoeuds;
  T * Items;
public:
  Monceau(unsigned int Taille, T Depart);
  // constructeur par défaut
  // taille du tableau  et Depart pour
  // la sentinelle
  // Si depart est tres petit
  // le noeud est inférieur a ses desendants
   // si depart est tres grand
  // le noeud est plus grand que ses descendants
  Monceau(const Monceau &H);
  // constructeur copie
  ~Monceau();
  // destructeur
  Monceau & operator = (const Monceau & H);
  // operateur affectation
  int RetourNbNoeuds(){return NbNoeuds;};
  bool Vide();
  // le monceau est-il vide?
  bool Plein();
  // le monceau est-il plein?
  void InsereInf( T Inserer);
  // inserer un objet T dans le monceau
  // noeud est inférieur a ses descendants
  T EnleveInf();
  // retirer l'objet minimum du monceau
  // noeud est inferieur a ses descendants
  void DescendsInf(unsigned int k);
  // à partir d'un indice k,
  // restructurer le monceau en descendant
  // noeud est inférieur a ses descendants
  void RemonteInf(unsigned int k);
  // à partir d'un indice k
  // restructurer le monceau en remontant
  // noeud est inférieur a  ses descendants
```

Question 3 -- (suite)

```
  T TrouveMinMax();
  // retourner l'objet a la position 1 du monceau
  // sans le retirer du monceau

  void InsereSup( T Inserer);
  // inserer un objet T dans le monceau
  // noeud est superieur a ses descendants
  T EnleveSup();
  // retirer l'objet  T du monceau
  // noeud est superieur a ses descendants
  void DescendsSup(unsigned int k);
  // à partir d'un indice k,
  // restructurer le monceau en descendant
  // noeud est superieur a ses descendants
  void RemonteSup(unsigned int k);
  // à partir d'un indice k
  // restructurer le monceau en remontant
  // noeud est superieur a  ses descendants

  void Ajouter (T Element);
  // ajouter un objet T à la fin du tableau

  T operator [](int);
  // retourne un objet selon un indice;

  void OrganiserMonceau(); // à faire

  friend void HeapSort(Monceau<T> &H);
  // fonction pour trier le tableau
};
// fonctions membres de la classe

template <class T>
Monceau<T>:: Monceau(unsigned int Taille, T Depart): NbNoeuds(0)
{ MaxTaille = Taille;
  Items = new T [MaxTaille];
  if(Items == NULL)
    throw " manque d'espace mémoire";
  else
      Items[0] = Depart;

}
template <class T>
void Monceau <T>:: Ajouter(T Element)
{  if (!Plein())
        Items[++NbNoeuds] = Element;
   else
         throw " Le tableau est plein";
}
```

Réponse à la question 3

```
template <class T>
void Monceau<T>::OrganiserMonceau()
{
  //Organiser le monceau
      for ( int k = NbNoeuds/2; k>=1; k--)
         DescendsInf(k);
}
```

Question 4 ---Technique de programmation (3.5 points)

Soit le labyrinthe suivant qui contient une case départ D et cinq cases de sorties S1, S2, S3, S4 et S5:

	0	1	2	3	4	5	6	7
0								
1		S1				S 2		
2								
3				D				
4								
5		S3					S4	
6				S5				
7								
8								

4.1 Quelle sera la première sortie atteinte si on utilise l'algorithme récursif présenté dans les notes de cours à la page 450 selon l'ordre Nord, Est, Sud, Ouest ?

4.2 Quelle sera la première sortie atteinte si on utilise l'algorithme ci-dessous implanté à l'aide d'une pile?

```
Marquer la case de départ D
Insérer dans la pile la case de départ D
Tant qu'une sortie n'est pas atteinte et que la pile n'est pas vide
        Retirer la case au sommet de la pile
        Pour les quatre voisins de la case retirée ( dans l'ordre N, E, S, O)
                Si le voisin n'est pas un mur et qu'il n'est pas marqué
                        Marquer le voisin
                        Insérer le voisin dans la pile
```

4.3 Quelle sera la première sortie atteinte si on utilise l'algorithme ci-dessous implanté à l'aide d'une file?

```
Marquer la case de départ D
Insérer dans la file la case de départ D
Tant qu'une sortie n'est pas atteinte et que la file n'est pas vide
        Retirer la case au début de la file
        Pour les quatre voisins de la case retirée ( dans l'ordre N, E, S, O)
                Si le voisin n'est pas un mur et qu'il n'est pas marqué
                        Marquer le voisin
                        Insérer le voisin dans la file
```

Réponse à la question 4.1

S1

Réponse à la question 4.2

S3

Réponse à la question 4.3

S5

Question 5 – Liste doublement chaînée (4.0 points)

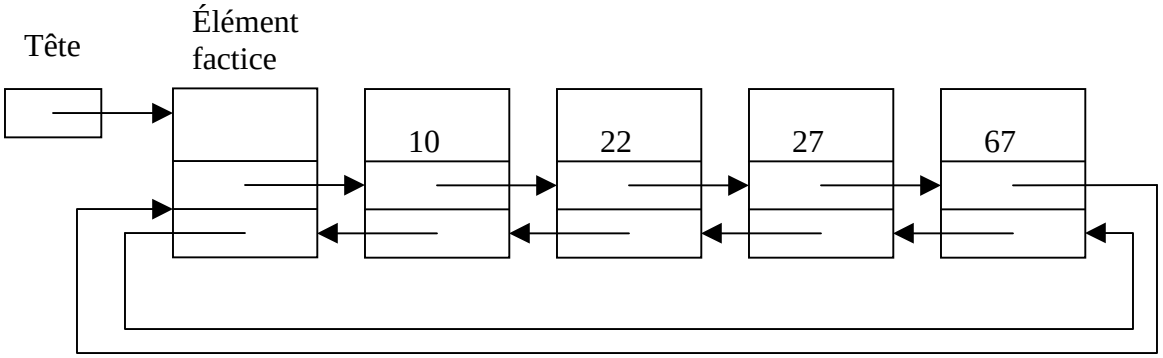
La classe `LChaindc` représente une liste doublement chaînée circulaire avec élément factice, écrire une fonction membre de cette classe qui retire et retourne en paramètre le dernier élément de la liste.

La signature de la fonction est :

```
bool ListRetraitFin(T & ObjetFin);
```

`ObjetFin:` le dernier élément de la liste;

La fonction retourne "`false`" si la liste est vide, sinon elle retourne "`true`".



D'après l'exemple, la fonction `ListRetraitFin` doit retirer et retourner en paramètre l'objet 67 de la liste.

```
#ifndef LCHAINEDC
#define LCHAINEDC
#include <assert.h>  // pour assert()
#include <stddef.h>  // pour NULL
#include <stdlib.h>  // pour exit()
template<class T>
class LChaindc
{protected:
     class Noeud  // un élément de la liste
   {public:
      T Element;
         Noeud * Suivant,    // pointeur au prochain élément
                 * Precedent;  // pointeur à l'élément précedent
    };  //fin de Noeud

     Noeud * Tete;
     int  Taille;          // nombre de noeuds de la liste
    Noeud * Courant;    // pointeur courant de la liste


     Noeud * RechercheNoeud(T ClasseCherchee);
     // recherche d'un élément dans la liste
```

Question 5 – (suite)

```
public:
// constructeurs et destructeurs:
      LChaindc();
      LChaindc(LChaindc& L);
      ~LChaindc();

// fonctions membres:
      bool ListVide();

      int ListLongueur()
      { return Taille;
      }; // fin ListLongueur

      bool ListInsere(T ClasseInsere);

      bool ListRetrait(T ClasseRetrait);

      bool ListRetraitFin(T & ObjetFin); // à faire

      bool ListRecherche(T & ClasseCherchee);
      // cherche un élément dans la liste selon une clef
      // et retourne l'élément dans ClassCherchee

      void DebutNoeud();
      // se placer au debut de la liste

      void FinNoeud();
      // se placer a la fin de la liste

      void  ProchainNoeud();
      // recherche le prochain noeud de la liste

      void  PrecedentNoeud();
      // recherche du precedent noeud de la liste

      T NoeudCourant()
      { return Courant->Element;
      }// retourne le noeud courant

}; // fin Lchaindc
```

Réponse question 5

```
template<class T>
bool LChaindc<T>::ListRetraitFin(T & ObjetRetrait)
{    if (!ListVide())
     {    Noeud * Supprime = Tete->Precedent;
          ObjetRetrait = Tete->Precedent->Element;
          Tete->Precedent= Tete->Precedent->Precedent;
          Tete->Precedent->Suivant= Tete;
          delete Supprime;
          --Taille;
          return true;
     }
   else
    return false;
}
```

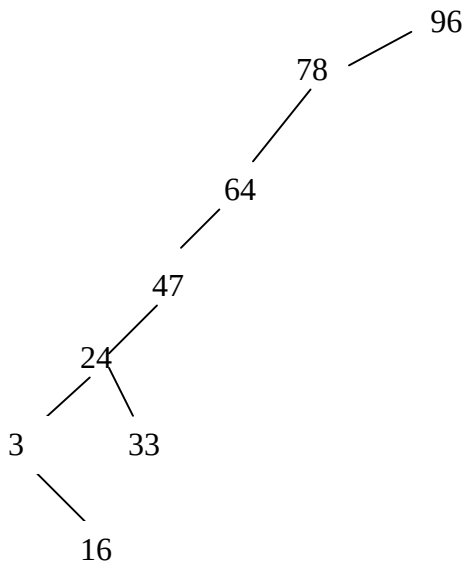
Question 6 – Arbre binaire de recherche (1.5 points)

Un certain programme construit un arbre binaire de recherche à partir de fiches lues d'un fichier sur disque; pour les fins de cette question, seules les valeurs de clefs sont intéressantes; les fiches sont insérées une à une dans un arbre binaire de recherche, et l'ordre d'arrivée des clefs est:
96, 78, 64, 47, 24, 33, 3, 16

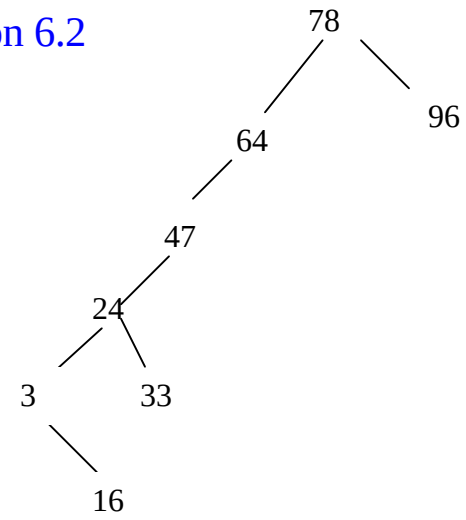
6.1 Dessinez l'arbre de recherche tel qu'il sera après cette série d'insertions.

6.2 Le programme effectue ensuite le retrait de l'élément situé à la racine de cet arbre, et un peu plus tard insère à nouveau ce même élément; dessinez l'arbre de recherche tel qu'il sera après ces deux opérations.

Réponse question 6.1

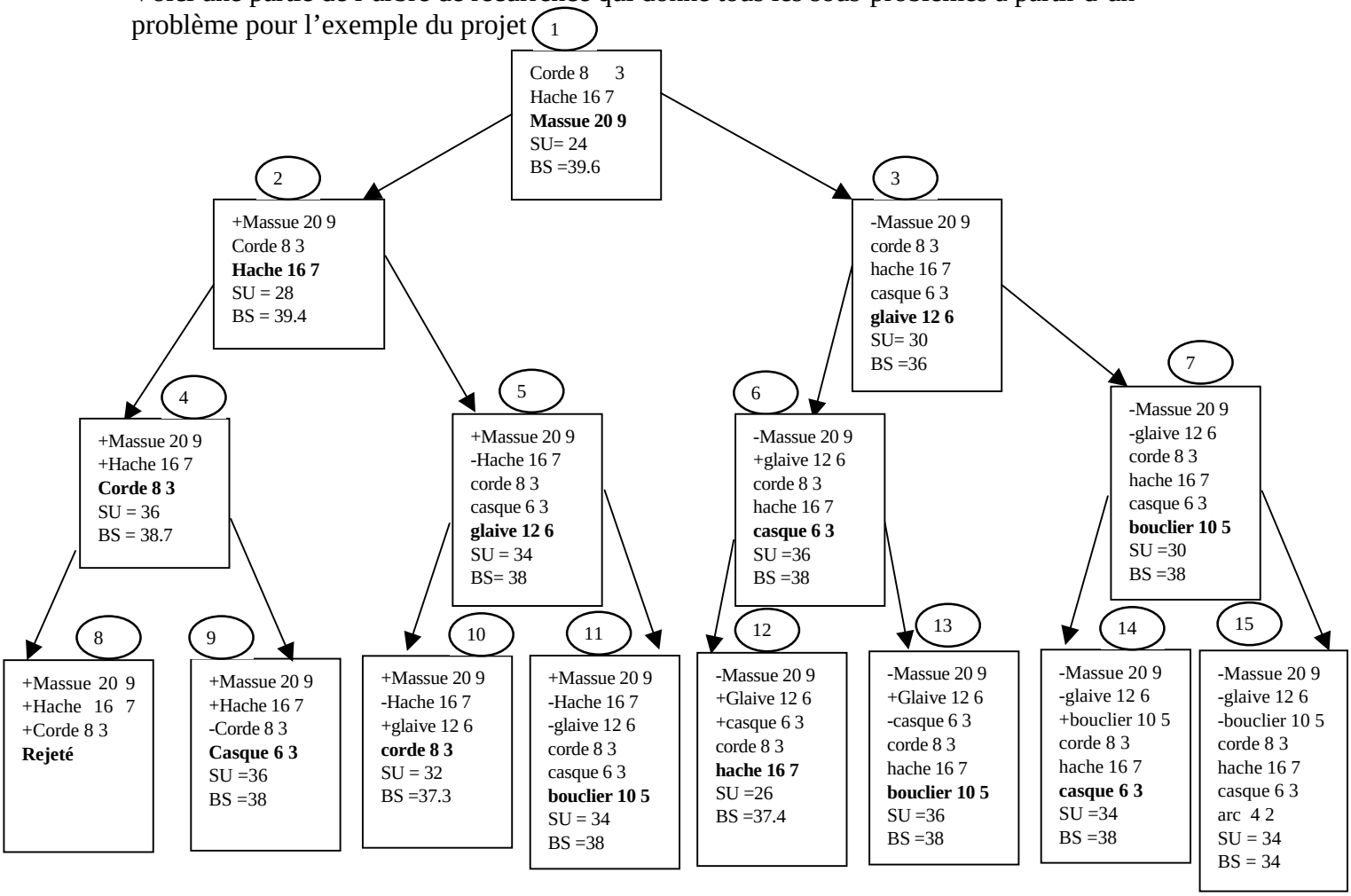


Réponse question 6.2



Question 7 --- Sac à dos (3.0 Points)

Voici une partie de l'arbre de récurrence qui donne tous les sous-problèmes à partir d'un problème pour l'exemple du projet



Dans le projet, la stratégie employée est la recherche avec le meilleur c'est-à-dire, qu'on retire et qu'on traite toujours de la liste de problèmes celui qui a la plus grande borne supérieure.

7.1 Si on avait utilisé une recherche en profondeur, énumérer dans quel ordre les problèmes auraient été traités.

7.2 Si on avait utilisé une recherche en largeur, énumérer dans quel ordre les problèmes auraient été traités.

Réponse à la question 7.1

1 2 4 8 9 5 10 11 3 6 12 13 7 14 15

ou

1 3 7 15 14 6 13 12 2 5 11 10 4 9 8

Réponse à la question 7.2

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

ou

1 3 2 7 6 5 4 15 14 13 12 11 10 9 8