DEPARTEMENT D'INFORMATIQUE

Faculté MI **ASDD** **première année Ingénieur S2**

Examen ASDD 2022-2023 **1h30h**

Exercice 1 : (4 pts)

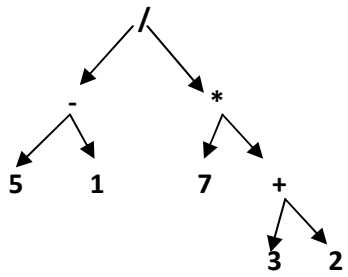
Etant donnée une file de données représentée par une liste chainée à deux pointeurs tête et queue. Chaque valeur est constituée d'une valeur supplémentaire qui défini sa priorité. Cette priorité soit égale 1 soit égale 0. A chaque défilement la primitive cherche dans la file le premier élément de priorité 1 pour le défiler si non trouvé elle défile celui qui est en tête de la file.

1) Donner la déclaration de cette file.

2) Ecrire la primitive défiler.

Exercice 2 : (9 pts)

Arbre A) :



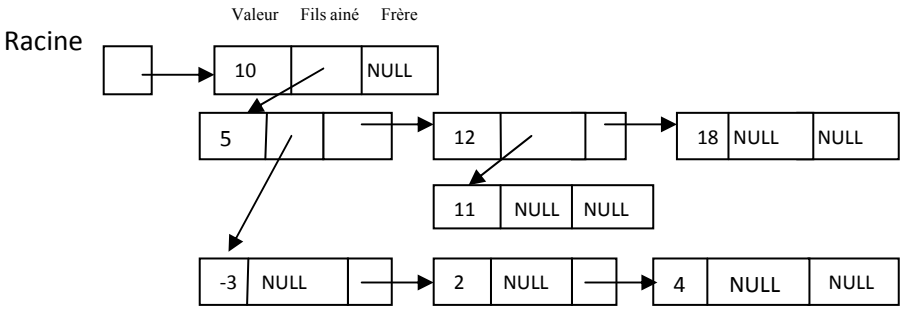
1) Ecrire la fonction qui affiche les valeurs des feuilles de gauche à droite (l'ordre 5 1 7 3 2).

2) Ecrire la fonction qui affiche les valeurs des feuilles dans l'ordre inverse (2 3 7 1 5).

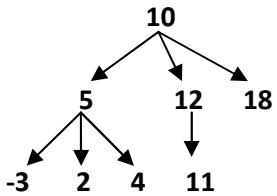
3) Ecrire la fonction récursive qui permet de copier un arbre vers sa version linéaire parenthésée sous forme de chaine de caractères S, en introduisant les parenthèses pour encadrer les sous-arbres opérationnels comme dans l'exemple suivant : **S =''/(-(5 1) *(7 +(3 2)))'' est la version linéaire de l'arbre A.**

Exercice 3 : (7 pts)

La représentation sous forme de listes chainées suivante représente une implantation d'un arbre 3-aire (Arbre B) qui suit.



Arbre B :



Chaque élément de la liste chainée est constituée de champs valeur d'un pointeur vers son fils ainé et d'un pointeur vers la liste des ses frères (dans l'exemple l'élément descendant de 10 qui a la valeur 5 est chainé avec son fils ainé (à valeur -3) par un pointeur et chainé à la liste de ses frères (12, 18) par un autre pointeur.

1) Ecrire une fonction qui calcule (la taille) le nombre de nœuds d'un arbre n-aire.

2) Ecrire une fonction qui affiche l'arbre n-aire par parcours préfixé.

3) donner l'affichage du parcours post-fixé de l'exemple Arbre B.

DEPARTEMENT D'INFORMATIQUE

Faculté MI **ASDD** **première année Ingénieur S2**

corrigé

Exercice 1 (4pts)

```
typedef struct file {
                        int val,prio ;
                        struct file *suivant ;
                    } file ;
file *tete,*queue ;

void Defiler(file **t, file**q, int *v )
        {  file *d ;
           if (*t !=NULL){
                        if((*t)->prio ==1)
                                {*v= (*t)→val ;
                                 d=*t ;
                                 *t=(*t)→suivant ;
                                 free(d) ;
                                 }
                        else { file *prec=*t,*p=(*t)->suivant ;
                               While(p !=NULL && p->prio ==0)
                                {prec=p; p=p->suivant;
                                }
                               if(p==NULL)
                                  {*v= (*t)→val ;
                                  d=*t ;
                                  *t=(*t)→suivant ;
                                  free(d) ;
                                  }
                               else{*v= p→val ;
                                    prec->suivant=p->suivant ;
                                    if(*q==p) *q=prec ;
                                    free(p) ;
                                   }

                             }
                        }
            }
```

0.5pt

0.5pt

1pt

0.5

1.5pt

Exercice 2(9pts)

```
void affprefixe (Noeud *a)
{     if  (a != NULL) {
                    if( a->FG==NULL &&  a->FD==NULL)
                        printf(" %c ",a->val);
                    else
                        { affprefixe ( a->FG) ;
                        affprefixe ( a->FD) ;
                        }
                     }
}
```

2pts

```
void affpostfixe (Noeud *a)
{     if  (a != NULL) {
                      if( a->FG==NULL &&  a->FD==NULL)
                          printf(" %c ",a->val);
                      else
                        { affpostfixe ( a->FD) ;
                          affpostfixe ( a->FG) ;
                        }
                        }
}
```

2pts

```
void lineaire (Noeud *a, char S[])
 { static int i=0;
   if  (a != NULL) {S[i]= a->val;i++;
                    if(a->FG!=NULL || a->FD!=NULL)
     // au moins un sous-arbre à encadrer|| pour opérateur unaire comme -(5)
                              {  S[i]= '(' ;i++;
                                lineaire(a->FG, S);
                                lineaire(a->FD, S);
                                S[i]= ')' ;i++;
                               };

                   }
   S[i]= '\0';
}
```

5pts

Exercice 3(7pts)

```
int taille (Noeud *N)
{
      if (N == NULL) return 0;
      return 1+ taille( N->Fia)+ taille ( N->Fr);
}
```

3pts

```
void affprefixe (Noeud *a)
{     if  (a != NULL)    { printf(" %d ",a->val);
                           affprefixe ( a->Fia) ;
                           affprefixe ( a->Fr) ;

                          }
}
```

3pts

Bonus

Affichage -3 2 4 5 11 12 18 10 1pt

```
void affpostfixe (Noeud *a)
{     if  (a != NULL) {
                       affpostfixe ( a->Fia) ;
                       printf(" %d ",a->val);
                       affpostfixe ( a->Fr) ;
                      }
}
```