Structures de Données en C
Solution de TD N°3

Exercice 1

Expression1 :

//Q1: La pile d'évaluation de l'expression1 : R1 = (((A + B)*C) + D) avec A = 1, B = 2, C = 4, D = 3

Pile	**Etape13**													**15**
	Etape12												**3 + 12 (**	
	Etape11											**+ 12 (**		
	Etape10										**12 (**			
	Etape9									**4 * 3 ((**				
	Etape8								*** 3 ((**					
	Etape7							**3 ((**						
	Etape6						**2 + 1 (((**							
	Etape5					**+ 1 (((**								
	Etape4				**1 (((**									
	Etape3			**(((**										
	Etape2		**((**											
	Etape1	**(**												
Entrée		**(**	**(**	**(**	**1**	**+**	**2**	**)**	*****	**4**	**)**	**+**	**3**	**)**

//Q2: Le résultat final que contient la pile est: 15

Expression2 :

//Q1: La pile d'évaluation de l'expression2 : R2 = ((A + B) *(C − D)) avec A = 4, B = 2, C = 5, D = 3

Pile	**Etape13**													**12**
	Etape12												**2 * 6 (**	
	Etape11											**3 - 5 (* 6 (**		
	Etape10										**- 5 (* 6 (**			
	Etape9									**5 (* 6 (**				
	Etape8								**(* 6 (**					
	Etape7							*** 6 (**						
	Etape6						**6 (**							
	Etape5					**2 + 4 ((**								
	Etape4				**+ 4 ((**									
	Etape3			**4 ((**										
	Etape2		**((**											
	Etape1	**(**												
Entrée		**(**	**(**	**4**	**+**	**2**	**)**	*****	**(**	**5**	**-**	**3**	**)**	**)**

//Q2: Le résultat final que contient la pile est: 12

Exercice 2

/***********************(Q1) **/**

//Les structures de données nécessaires à cette pile :

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
//---------------------------------------------
typedef struct ElementRepere{
  char NomForme;
  int SurfaceForme;
  ElementRepere*suivant;
}Element;
//---------------------------------------------
typedef struct FileRepere {
     Element *tete;
     Element *queue;
     int nef;
 }File;
//---------------------------------------------
Element* CreerElement( ){
   Element* El;
   El=(Element*)malloc(sizeof(Element));
   El->NomForme=0;
   El->SurfaceForme=0;
   El->suivant=NULL;
   return(El);
}
//---------------------------------------------
File* CreerFile( ){
   File* Fi;
   Fi= (File*)malloc(sizeof(File));
   Fi->tete=NULL;
   Fi->queue=NULL;
   return(Fi);
}
```

/***********************(Q2) **/**

//La fonction Enfiler():

```
void Enfiler(File* Fi, char  nom, int surface){
   Element* El;
   El = CreerElement();
  if(Fi == NULL || El == NULL){
      exit(EXIT_FAILURE);
   }
   El->NomForme = nom;
   El->SurfaceForme = surface;
  //cas d'une file vide
    if(Fi->tete==NULL){
       Fi->tete = El;
       Fi->queue = El;
   }
  //cas d'une file non vide
    else{
```

```
    Fi->queue->suivant = El;
    Fi->queue = El;
   }
   Fi->nef++;
 }
```

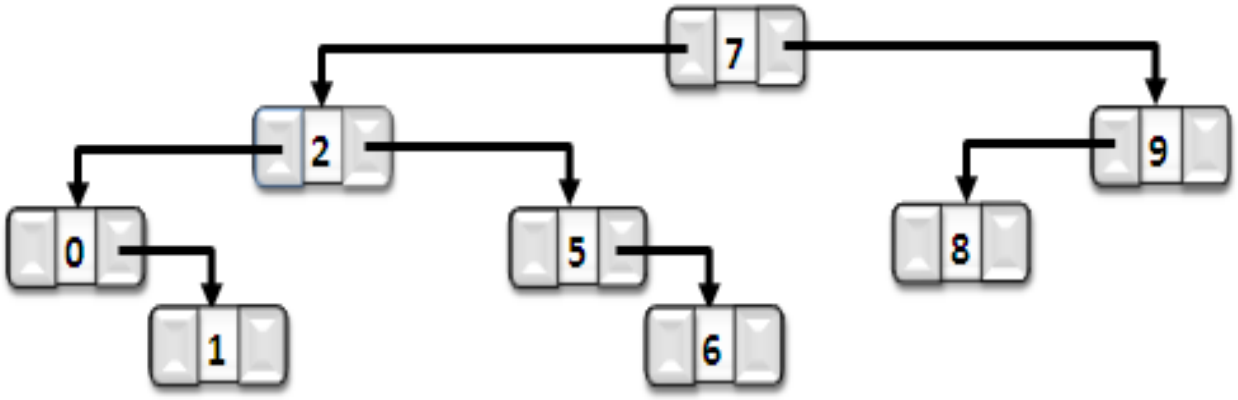
/***********************(Q3) **/**

//La fonction Defiler():

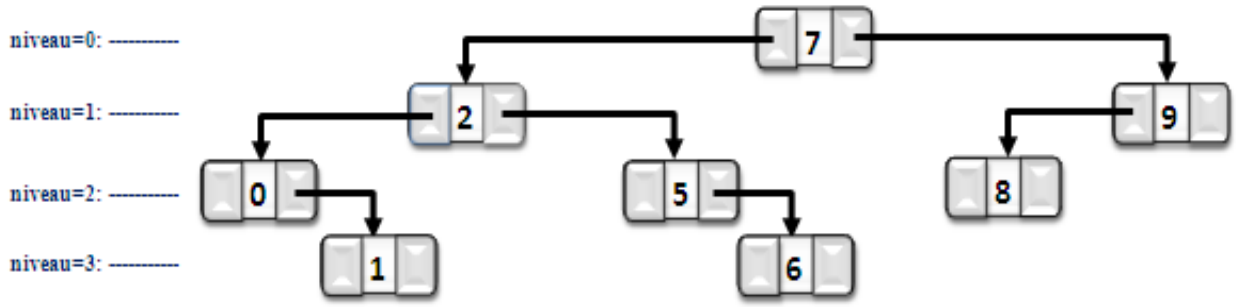
```
int Defiler(File* Fi){
  char nsd;
  int ssd;
  Element* ptr;
  if(Fi == NULL){
     exit(EXIT_FAILURE);
  }
   //cas d'une file vide
    if(Fi->tete == NULL){   //
     return(-1);
  }
  //cas d'une file non vide
 else{
    ptr = Fi -> tete ;
    Fi->tete = Fi ->tete->suivant;
    nsd=ptr ->NomForme;
    free(ptr);
    Fi -> nef--;
    if(Fi->nef==0){   /
          Fi->queue=NULL;
    }
    return(nsd);
  }
}
```

Exercice 3

//(Q1) La représentation de l'arbre binaire de recherche obtenu en insérant successivement les entiers suivants 7, 2, 9, 0, 5, 6, 8 et 1 est :



//(Q2) L'affichage du niveau de chaque nœud est (voir la figure) :

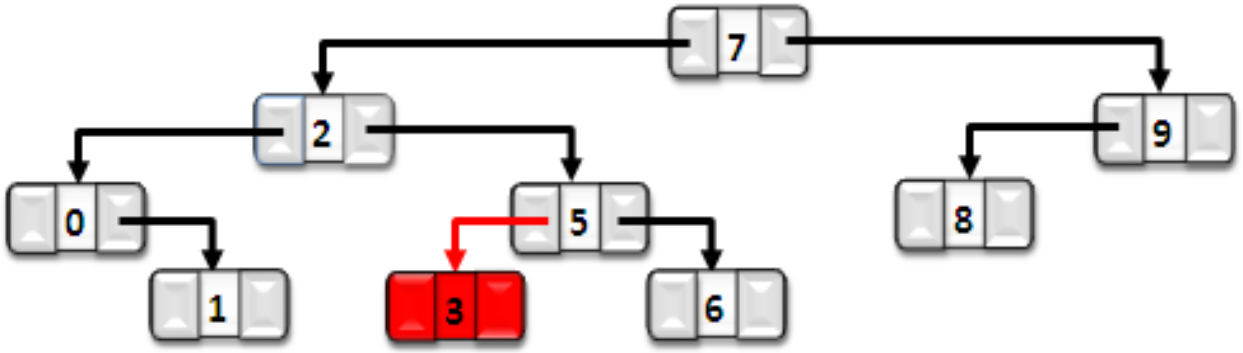


//(Q3) L'affichage du contenu de cet arbre par le parcours Infixe (GRD) est: 0,1,2,5,6,7,8,9

//(Q4) L'affichage du contenu de cet arbre par le parcours Préfixe (RGD) est: 7,2,0,1,5,6,9,8

//(Q5) L'affichage du contenu de cet arbre par le parcours Postfixe (GDR) est: 1,0,6,5,2,8,9,7

//(Q6) La représentation graphique de l'arbre obtenu après l'insertion de l'entier : est:



//(Q7) La hauteur de l'arbre obtenu est : 3