**Prof : Elie Hamouche** **année :2019**

**Exercice supplément liste doublement chainée et pile**

**NB :** Dans tout ce qui suit, on utilisera les méthodes usuelles liées aux piles ***Empiler(P,a), Depiler(P), Peek(P) , EstVide() ,EstPleine****()* existent ..

**1**. Ecrire une fonction récursive qui évalue une suite géométrique **Un** de raison **q=1/2** avec :

**U0 = 1**

**Un = q\*Un-1**

$$\sqrt{1/64}$$

$$\sqrt{1/4}$$

$$\sqrt{1/16}$$

$$\sqrt{1/2}$$

$$\sqrt{1/32}$$

$$\sqrt{1/8}$$

 **Début fin**

**Liste doublement chainée dList des** $\sqrt{Un}$

typedef struct node

{

 float data;

 struct node \*next;

 struct node \*prev;

}node;

typedef struct dl

{

 size\_t length;

 struct node \*last;

 struct node \*first;

} Dlist;

float suitegeomR(int n, int q) {

 int res;

 if (n == 0)

 {

 res = 1;

 }

 else if (n == 1) {

 res = (1 / 2);

 }

 else if (n>1) {

 res = (q\*suitegeomR (n - 1)) ;

 }

 else

 {

 return -1;

 }

 return res;

}

1. Ecrire une procédure qui remplit la liste double dL par les termes de √Un  :  **void RemplirDlist (int n)?**

//fonction qui ajoute un élément en tête de Dlist

Dlist\* dlist\_AjoutTete(Dlist \*dl, int od)

{

 if (dl != NULL)

 {

 node \*nouv = (node\*)malloc(sizeof(node));

 nouv->data = od;

 nouv->prev = NULL;

 if (dl->last == NULL)

 {

 nouv->next = NULL;

 dl->first = nouv;

 dl->last = nouv;

 }else{

 dl->first->prev = nouv; // On relie le 1er élément de la liste vers nouv

 nouv->next = dl->first; // On fait pointer next de nouv vers le 1er élément

 dl->first = nouv; // On fait pointer le début de liste vers le nouv

 }

 dl->length++;

 }

 return dl;

}

void RemplirDlist (int n) {

 Dlist \*nouv = (Dlist\*)malloc(sizeof(Dlist));

 // remplir Dlist

 int q =0.5;

 for (int i= 0; i< n; i++){

 nouv= dlist\_AjoutTete(nouv, suitegeomR(i,q)) ;

 }

 dlist\_display(nouv) ;

}

// function récursive qui affiche les éléments de Dlist

 void displayDlistR(Dlist \*dl)

 {

 node \* tmp = dl->first;

 if (tmp != NULL)

 {

 cout << "la donnee :" << tmp->data << "\n";

 displayDlistR (tmp ->next);

 }

}

1. Ecrire une procédure qui **remplace la clé d’un élément p paire par 2 éléments dont les clés sont égales à la clé / 2 .** **getReplaceHalf(Dlist \*dl) :void**

void getReplaceHalf(Dlist \*dl){

 node \*tmp = new node;

 node \*q = new node;

 tmp= dl->first ;

 while (tmp != NULL){

 if (tmp->data %2 == 0){

 q->data= p->data/2 ;

 tmp->data= tmp->data/2 ;

 q->next = tmp->next; //succ de (q) = succ de (tmp)

 tmp->next = q; // succ de (tmp) = q

 tmp = q->next ; // p =succ de (q)

 }

 else{

 tmp =tmp->next;

 }

 }

 cout << "les clés paires sont remplacées par la moitié " << "\n";

 displayDlistR(dl) ;

 }

1. Ecrire une fonction **qui supprime un élément sur 2** de la liste double **Dlist** ?

 void deleteUnSurdeux(Dlist \*dl){

 node \*tmp = new node;

 node \*q = new node;

 tmp= dl->first ;

 while (tmp != NULL && tmp->next !=NULL){

 q->tmp->next ; // q= succ de (tmp)

 tmp->next = q->next; //succ de (tmp) = succ de (q)

 delete(q); // supprimer q

 tmp =tmp->next; //tmp = succ de (tm)

 }

 }

 cout << "un élément sur 2 est supprimé " << "\n";

 displayDlistR(dl) ;

 }

1. Ecrire une fonction ou procédure qui prend en argument une liste double **dList** et **remplira les piles P1 par les 3 premières valeurs de Dlist et P2 par les autres** ? déduire la somme de chacune d’elle ?

//fonction qui accède a la kieme cellule de Dlist

float \*acceskieme(Dlist \*dl,int k){

 node \*tmp = new node;

 tmp= dl->first ;

 int i = 1;

 while (tmp != NULL && i < =k)

 tmp = tmp->next;

 i++;

 }

 return (tmp->data);

}

//fonction qui éclate une Dlist en 2 autres P1 et P2 et moyenne de chacune d’elle

void EclaterDlist (Dlist \*dl) {

 pile \*p1 = new Pile;

 pile \*p2 = new Pile;

 int s1,s2,cpt1,cpt2;

 float \*T1 = new float[3];

 float \*T2 = new float[3];

 cpt1=cpt2= 0;

 s1=s2=0;

 node \* tmp = dl->first;

 while (tmp != NULL) {

 // remplissage 1ere pile

 for(int i = 1;i<=3;i++)

 {

 T1[i] =acceskieme(dl,i);

 p1 =Empiler(p1,T1[i]);

 s1+ = T1[i] ;

 cpt1++ ;

 }

 // remplissage 2eme pile

 for(int j = 3;j<=6;j++)

 {

 T2[j] =acceskieme(dl,j);

 P2 =Empiler(p2,T2[j]);

 S2+ = T1[j] ;

 cpt2++ ;

 }

 tmp=tmp->next;

 }

 cout << "la somme des éléments de la pile p1 : " << s1 << "\n";

 cout << "la somme des éléments de la pile p2 : " << s2 << "\n";

 cout << "la moyenne dans la pile p1 : " << s1/cpt1 << "\n";

 cout << "la moyenne dans la pile p2 : " << s2/cpt2 << "\n";

}