Examen d'informatique
Durée 1h 30mn

Exercice 1

Ecrire une fonction qui permet de supprimer tous les espaces multiples à l'intérieur d'une chaîne de caractère **S** (s'il y'a 2 espaces consécutifs ou plus, on ne laisse qu'un seul espace).

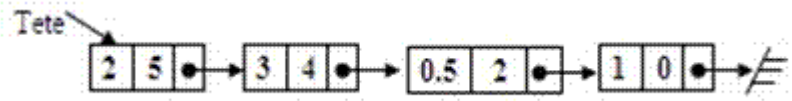
Exemple si S = SMI S4 (4 espaces), après l'appel de la fonction S= SMI S4 (1 espace).

```
/*fonction pour supprimer les caractères espace au milieu en double*/
voidSuppEspaceMilieu (char T[] )
 {
int i,j=0;
for (i=0 ;T[i]!='\0';i++)
            {
          T[j] = T[i];
                  if(T[i]!=' '||T[i+1]!=' ')
                  j++;
            }
      T[j]='\0';
 }
```

Exercice 2

On souhaite représenter les polynômes par des listes linéaires chainées, où chaque terme d'un polynôme est rangé dans un nœud de la liste contenant le degré (**degre**) du terme et le coefficient (**coef**) correspondant.

Exemple Le polynôme $\mathbf{2x^5 + 3x^4 + 1/2x^2+1}$est représentée par la liste suivante :



On vous demande de :

1. Donner les structures de données (**poly**) nécessaires à la représentation des polynômes.

```
typedef struct poly
{
    float coef;
    int degre;
    struct poly *suivant;
}poly;
```

2. Ecrire la fonction ***voidRemplirListe(poly *p)*** qui permet deremplir les champs de la structure.

```
voidRemplirliste(poly *p)
{
printf("\ndonner le coefficient\t");
scanf("%f",&p->coef);

printf("\n donner le degre\t");
scanf(" %d",&p->degre);
}
```

3. Ecrire la fonction ***voidafficherListe(poly *p)*** qui permet d'afficher les éléments de la liste.

```
voidafficherListe(poly *T)
{
   poly *p= T;
  while (p != NULL)
   {
         printf("%.2f * X^ %d ",p->coef,p->degre);

         printf(" +");
         p = p->suivant;
   }
 }
```

4. Ecrire la fonction ***voidafficherdegre(poly*P)*** qui permet d'afficher le degré du polynôme.

```
voidafficherdegre(poly*T)
{
poly *p=T;
printf("\nledegre du polynome = %d\n", p->degre);
}
```

5. Ecrire la fonction ***floatvaleur(poly *p, float x)***permettant de donner la valeur du polynôme P pour x.**Exemple** : Valeur(p, 1) = 6.5

```
float valeur(poly *p, float x)
{
float  s=0;
   int i,c;
while(p!=NULL)
     {
        c=1;
for(i=1;i<=p->degre;i++)
           {
              c=c*x;
           }
         s=s+ (c*p->coef);
         p=p->suivant;
}   return s;  }
```

6.1 Donner le résultat après l'exécution du code ci-dessous en prenant comme entrée les polynômes suivants : (Pour les pointeurs T, p et q vous marquez (NULL) ou (!=NULL))

Avant l'appel de la fonction	Après l'appel de la fonction	T	P	q
P1= $2x^5 + 3x^4 + 3x^2+1$	**$10x^4 + 12x^3 + 6x^1$**	**!= NULL**	**NULL**	**!= NULL**
P2= $2x^5 + 3x^4$	**$10x^4 + 12x^3$**	**!= NULL**	**NULL**	**!= NULL**
P3= $2x^0$	**Liste Vide**	**NULL**	**NULL**	**NULL**

```
poly* test( poly *T)
{
poly *q=NULL,*p=T;
while(p!=NULL)
{
        if(p->degre!=0)
          {
p->coef=p->coef*p->degre;
             p->degre=p->degre-1;
q=p;
p=p->suivant;
}
        else
          {
            if(q==NULL)
              {
                T=NULL;
              }
            else
              {
                 q->suivant=NULL;
              }
                 free(p);
p=NULL;
}
    }
return T;
}
```

6.2 Déduire le rôle de ce programme.

Le programme permet de calculer la dérivée du polynôme

7. La fonction suivante permet de calculer l'intégrale d'un polynôme P.

Exemple : pour un polynôme **P = $8x^3 + 9x^2$,** l'appel de la fonctionintegrer(T) donne le résultat suivant : **P = $2x^4 + 3x^3$** alors que le résultat attendu est **P = $2x^4 + 3x^3$ + C** (C une constante ∈ R) . Compléter la fonction **voidintegrer(poly *T)** pour obtenir le résultat attendu c'est-à-dire **P = $2x^4 + 3x^3$ +1.**

```
poly * integrer(poly *T)
{
   poly *q,*p=T;
   q=NULL;
   while(p!=NULL)
{
p->coef=p->coef/(p->degre+1);
            p->degre=p->degre+1;
q=p;
p=p->suivant;
      }

poly *nouveau, *courant;
nouveau = (poly*) malloc (sizeof(poly));
nouveau->coef=1;
nouveau->degre=0;
nouveau->suivant=NULL;
if(q==NULL)
            {
               T=nouveau;
            }
else
            {
q->suivant=nouveau;
            }
return T;
}
```

Examen de programmation 2
Durée 1h 30mn

Soit PL un plan, on notera P(x,y) un point de coordonnées x et y dans ce plan.Soit N une constante entière strictement positive. On notera PL(N) l'ensemble des points du plan PL (figure1).

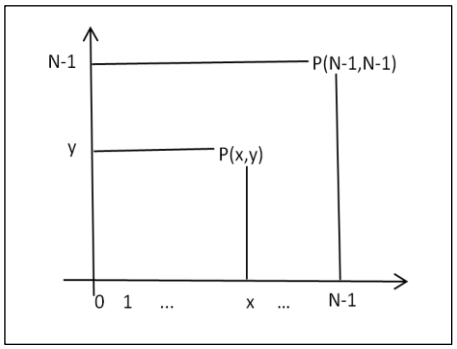


Figure 1

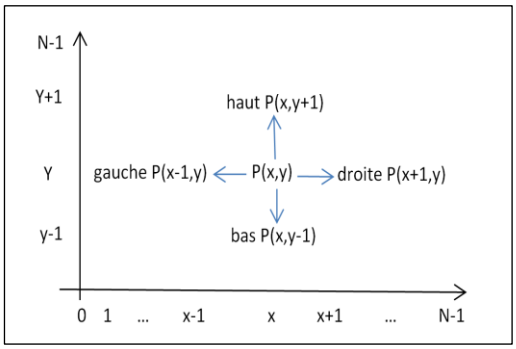


Figure 2

Soient A = $P(x_0,y_0)$ et B= $P(x_k,y_k)$ deux points distincts de PL(N). On appelle chemin de A vers B, un ensemble C de points ordonnés et différents : C={ $P(x_0,y_0)$, $P(x_1,y_1)$, $P(x_2,y_2)$…… $P(x_K,y_K)$}. Chaque point du chemin $P(x_0,y_0)$ est relié au point suivant $P(x_1,y_1)$ par l'une des quatre relations suivantes (figure 2):

Gauche :$P(x_1,y_1)$ est à gauche de $P(x_0,y_0)$ et dans ce cas $x_1= x_0 -1$ et $y_1 = y_0$
Droite : $P(x_1,y_1)$ est à droite de $P(x_0,y_0)$ et dans ce cas $x_1= x_0 +1$ et $y_1 = y_0$
Haut : $P(x_1,y_1)$ est au-dessus de $P(x_0,y_0)$ et dans ce cas $x_1= x_0$ et $y_1 = y_0 + 1$
Bas : $P(x_1,y_1)$ est en dessous de $P(x_0,y_0)$ et dans ce cas $x_1= x_0$ et $y_1 = y_0 – 1$

Remarque : soit A=P(2,3) et B= (5,5) deux points de PL(N), entre A et B il peut exister plusieurs chemins différents. C1 et C2 sont deux exemples de chemins entre A et B :

C1={P(2,3), P(3,3), P(4,3), P(4,4), P(4,5), P(5,5)}
C2={P(2,3), P(2,4), P(3,4), P(3,5), P(4,5), P(5,5)}

Question 1

Définir un nouveau type **point** sous forme d'une structure qui permet de représenter les coordonnées d'un point du plan PL(N). Utiliser la variables x pour l'abscisse et y pour l'ordonné.

```
typedefstruct point
 {
        intx,y;
 }point;
```

Question 2

Soient A et B deux points de PL(N), pour représenter l'ensemble des points d'un chemin de A vers B on utilise un tableau C de type **point** déclaré comme suit : **point C[20]**.

Ecrire une fonction **voidinitialiserC()** qui permet d'initialiser tous les éléments du tableau C par les points P(-1,-1).

```
for (i=0;i<N;i++)
        {
            c[i].x=-1;
            c[i].y=-1;
        }
```

Question 3

Soient A et B deux points de PL(N), on appelle **cheminHV** le chemin de A vers B construit de deux types de déplacements, d'abord le **cheminH** qui consiste à se déplacer à partir de A horizontalement (soit à gauche, soit à droite) jusqu'à arriver au point ayant la même abscisse que B, puis le**cheminV**qui consiste à se déplacer verticalement (soit en haut, soit en bas) jusqu'à arriver au point B.

Exemple

Soit A=(5,2) et B=(0,4) de PL(N), le chemin entre A et B selon le principe décris ci-dessus est : C[0]=P(5,2), C[1]=P(4,2), C[2]=P(3,2), C[3]=P(2,2), C[4]=P(1,2), C[5]=P(0,2), C[6]=P(0,3), C[7]=P(0,4), C[8]=P(-1,-1), C[9]=P(-1,-1)…….C[19]=P(-1,-1).

Ecrire une fonction **voidcheminH (point A, point B)** qui permet de créer uniquement le premier type de déplacementde A vers B, c'est-à-dire le **cheminH**.

```
void cheminH(point A,point B)
  {
    int i,j;
         c[0]=A;
         j=0;
    if(A.x<B.x)
        for (i=A.x;i<=B.x;i++)
         {
            c[j].x=i;
            c[j].y=A.y;
            j++;
         }
    else
       for (i=A.x;i>=B.x;i--)
         {
            c[j].x=i;
            c[j].y=A.y;
            j++;
         }
}
```

Question 4

On considère le code suivant :

```
point A={1,1};
point B={5,7};
pointtabC[3][10];
int distance(intnum)
{
      int d=0;
          while(tabC[num][d].x!=-1 &&tabC[num][d].y!=-1 && d<10)
                    d++;
          return d-1;
}
int distanceM()
{
    int i, k=0;
    for(i=0;i<3;i++)
          if(distance(k)>distance(i))
                    k=i;
    return distance(k);
}
```

a. Donner la valeur qui serait renvoyer suite à l'appel de la fonction **distanceM()** avec :

tabC =

2,1	2,2	3,2	3,3	3,4	3,5	4,5	5,5	5,4	-1,-1
2,1	3,1	4,1	5,1	5,2	5,3	5,4	-1,-1	-1,-1	-1,-1
2,1	2,2	2,3	2,4	3,4	4,4	5,4	-1,-1	-1,-1	-1,-1

b. A quoi correspond la valeur retournée.

a. La valeur retournée = 6

b. La valeur retournée correspond au chemin le plus court

Question 5

Le chemin d'un point A vers un point B peut être aussi représenté par une listé chainée de point. Définir un nouveau type **cheminListe** qui permettra de représenter les points d'un chemin entre A et B sous forme d'une liste chaînée. Utiliser la structure **point** déjà déclaré dans la première question.

```
typedefstructtypechemin
  {
        point p;
        structtypechemin *suivant;
}cheminListe;
```

Question 6

Soient A et B deux points du PL(N), on suppose avoir défini une fonction de prototype **cheminListe *AversB(point A, point B),** qui permet de créer un chemin quelconque de A vers B sous forme d'une liste chaînée de type **cheminListe** et de retourner l'adresse de son premier élément.

On considère le code suivant :

```
cheminListe *q,*p;
q=AversB(A,B);
// premier cas
p=q;
while(p!=NULL)
p=p->suivant;
// deuxième cas
p=q ;
while(p->suivant!=NULL)
p=p->suivant;
```

Premier cas : P=NULL

Deuxième cas : <> NULL

Indiquer pour les deux cas si la valeur finale de p est NULL ou non suite à l'exécution de ce code.

Question 7

Soit la déclaration suivante : **cheminListe *q ;**

Ecrire l'instruction pour réserver un espace mémoire de type **cheminListe** et de stoker son adresse dans le pointeur q.

```
q=(cheminListe*)malloc(sizeof(cheminListe));
```

Question 8

Soient A, B et R trois points distincts de PL(N), en utilisant la fonction **AversB(point A, point B)**, écrire une fonction **cheminListe *cheminRepere(point A,pointB,point R)** qui permet de créer un chemin de A vers B en passant par un point repère R. la fonction doit retourner l'adresse du premier élément de ce chemin.

Exemple : soient A=P(1,2), B=P(4,2) et R=P(0,1), l'appel de la fonction **cheminRepere(A,B,R)** peut retourner par exemple l'adresse du chemin C suivant:

C= P(1,2), P(0,2), P(0,1), P(1,1), P(2,1), P(2,2), P(3,2), P(4,2)

```
cheminListe *cheminRepere(point A, point B, point R){
        cheminListe *q,*q1,*p;
        q=AversB(A,R);
        q1=AversB(R,B);
        p=q;
        while(p->suiv!=NULL)
                p=p->suiv;
           p->suiv=q1->suiv;
           free(q1);
        return q;
  }
```

Opérations sur les polynômes à coefficients réels en utilisant les tableaux

Exercice:
Un monôme est une expression de la forme : ax^n ou **a** est un nombre réel et **n** un entier naturel.
Le nombre **a**est appelé coefficient du monôme et le nombre **n** est appelé le degré du monôme.
Exemple :

- $3x^2$ est un monôme du second degré et de coefficient 3

La somme de plusieurs monômes est un polynôme.
Exemples :

- $3x^2 - 5x + 7$ est un polynôme de degré 2
- $-x^3 + 4x - 9$ est un polynôme de degré 3
- 3 est un polynôme de degré 0

Ce problème s'intéresse aux algorithmes réalisant quelques traitements sur les polynômes à coefficients réels.
La représentation des polynômes peut se faire à l'aide des tableaux à une dimension (vecteur).
Exemple :
Le polynôme $P=5x^6+3x^3-8x+2$ peut être représenté par le tableau A=[2,-8,0,3,0,0,5]
Le tableau A suivant A=[0,2,0,0,-7,3] représente le polynôme $P=3x^5-7x^4+2x$
NB: on appelle polynôme tableau A, un tableau A de réels contenant les coefficients d'un polynôme P.

Question 1 : écrire une fonction d'entête ***LecturePolynome(float A[ ], int n)***qui permet desaisir les coefficients d'un polynôme de degré n dans un tableau passé comme paramètre.
Exemple :
Si le polynôme a lire est : $3x^5+6x^3+2$
alors le polynôme tableau sera A=[2,0,0,6,0,3]

Question 2 : écrire une fonction d'entête ***ValPolynome_x (floatA[ ], int n, float x)***qui prend trois paramètres, un polynôme tableau, son degré n et un nombre réel x et qui retourne la valeur du polynôme tableau en ce point x.
Exemple :
Si le tableau polynôme A =[3,2,0,4] et x = 2
alors la valeur retourné par la fonction est : $4*2^3+0*2^2+ 2*2^1+3=39$

Question 3 : écrire une fonction d'entête ***DeriveePolynome(float A[ ], int n , float D[ ])***qui prend en argument un polynôme tableau A, son degré n et un polynôme tableau D et qui permet de mettre le polynôme dérivé du polynôme tableau A dans le polynôme tableau D.
Exemple :
Si A=[3,2,0,8,0,4] son polynôme dérivé est D=[2,0,24,0,20]

Question 4 : écrire une fonction ***SommePolynome(float A[ ],int p, float B[ ],int q, float S[ ])*** qui calcule la somme de deux polynômes tableaux A et B de degrés a et b respectivement et qui met le résultat dans le polynôme tableau S.
Exemple :
Si A=[5,4,6,-1,0,7] et B=[2,8,-5,3] alors S=[7,12,1,2,0,7]

Solution

```
#include<stdio.h>
#include<stdlib.h>
  float A[32];
  float B[32];
  float D[32];
float S[32];
   /* A,B,S sont des polynomes */
//Lecture d'un polynome
void LecturePolynome(float A[], int n)
 {
   int i;
for(i=n;i>=0;i--)
      {
printf("nonner le coefficients d'ordre %d\n",i);
scanf("%f",&A[i]);
      }
 }
//Affichage d'un polynome
void AffichePolynome(float A[],int n)
  {
  int i;
  float v;
    for (i=n;i>=0;i--)
    {
      if (A[i]==0)
       continue;
      if (A[i] < 0)
       {
         v=-A[i];
printf(" -");
       }
      else
       {
         v=A[i];
         if (i!=n)
printf(" +");
       }
      if (v!=1. || i==0)
printf(" %g",v);
      if (i> 1)
printf(" X^%d",i);
      else
        if (i==1)
printf(" X");
}
```

```
printf(" ;\n");
 }

/*Calcul de la fonction polynômiale en un point x*/
float ValPolynome_x(float A[],int n,float x)
  {
   float r=0;
   int i;
   for (i=n;i>=0;i--)
     r=A[i]+x*r;
   return r;
}

//Dérivation d'un polynôme
intDeriveePolynome (float A[], intn,float D[])
{
 int i;
   if (n==-1)
     return -1;
   for (i=n;i>=1;i--)
     D[i-1]=i*A[i];
return n-1;
 }

// Addition de deux polnomes
int SommePolynomes (float A[],int p , float B[],int q,float S[])
{
 int i=0,j;
   if (p < q)
    {
      for (i=0;i<=p;i++)
         S[i]=A[i]+B[i];

      for (j=i;j<=q;j++)
         S[j]=B[j];
     return q;
    }
   if (q < p)
    {
      for (i=0;i<=q;i++)
        S[i]=A[i]+B[i];
      for (j=i;j<=p;j++)
         S[j]=A[j];
      return p;
    }
/* on a p=q, on a donc, pour tout i, S[i]=A[i]+Q[i] */
   if (p==q)
    {
     for (i=0;i<=p;i++)
```

```
      S[i]=A[i]+B[i];
       return p;
}
  }
main()
{
intn,p,q,r,s;    /* p est le degre du polynome P
             /* q est le degre du polynome Q */
floatvaleur,x;

printf("donner le degre du polynome \n");
scanf("%d",&n);

LecturePolynome(A,n);
AffichePolynome(A,n);

printf("nonner la valeur du point x\n");
scanf("%f",&x);
valeur= ValPolynome_x(A,n,x);
printf("valeur du polynome en %.2f= %.2f\n",x,valeur);

printf("la derive du polynome est:\n");
r=DeriveePolynome (A,n,D);
AffichePolynome(D,r);

printf("somme de deux polynome A et B\n");
printf("donner le degre nu polynome A\n");
scanf("%d",&p);
LecturePolynome(A,p);
AffichePolynome(A,p);

printf("donner le degre du polynome B\n");
scanf("%d",&q);
LecturePolynome(B,q);
AffichePolynome(B,q);

  s=SommePolynomes (A,p,B,q,S);
printf("S = ");
AffichePolynome(S,s);
}
```