Exercice 1 (15 points) :

Soit un tableau T contenant des entiers positifs et négatifs.

Ecrire un programme qui permet d'extraire de T le sous-tableau U dont la somme est la plus proche à 0. Il peut y avoir plusieurs sous-tableaux de ce type, il suffit dans ce cas d'en générer un seul.

Exemples :

- *Entrée : **T = {-1,3,4, -7,12}**, Sortie **U= {3, 4, -7}***
- *Entrée : **T = {-1,1,4,-3,3,7}**, Sortie **U= { -1,1}** ou bien **U = {-3,3}***
- *Entrée : **T = {-2,5,7,9,1}**, Sortie **U= {1}***

Code :

```
#include <stdio.h>
#include <stdlib.h>

void plusproche (int * t , int n ){
   int i,j,k,som,min,im,jm;

   for ( i = 0; i < n; i++)
   {
      for ( j = i; j < n; j++)
      {
         som=0;
         for (k = i; k <= j; k++)
         {
            som = som+t[k];
            if (som<min && som >=0)
            {
              min=som;
              im=i;
            jm=j;
            }

         }

      }
```

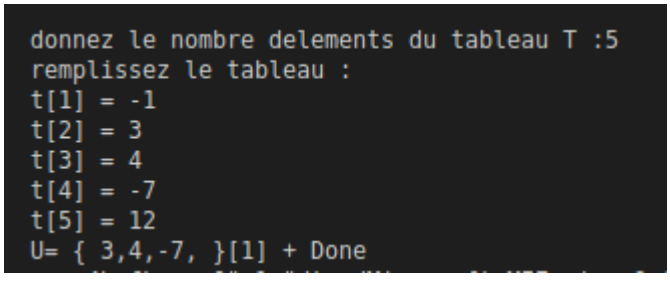
```
    }
    printf("U= { ");
    for(i=im;i<=jm;i++){
        printf ("%d,", t[i]);
    } printf(" }");
}
int main(){
    int i,n;

    printf ("donnez le nombre delements du tableau T :");
    scanf("%d",&n);
    int *t;
    t=(int *)malloc(n*sizeof(int));
    printf ("remplissez le tableau :\n");

    for(i=0;i<n;i++){
      printf("t[%d] = ",i+1);
      scanf("%d",&t[i]);
    }

    plusproche(t,n);
}
```

Execution :



Exercice 6 (18 points)

On considère un polynôme de degré n à coefficients a_i et à variable x réels donné par le schéma usuel suivant :

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

Pour représenter ce polynôme on peut utiliser, parmi les solutions, une liste doublement chaînée dont les nœuds représentent les termes du polynôme. Chaque terme est un monôme défini par son coefficient et son degré.

Exemple :

Le polynôme $P(x) = 4x^7 - 7x^3 + 6$

Est représenté par la liste P= {(4,7) ;(-7,3) ;(6,0)}

1. Ecrire la déclaration de la structure qui représente un monôme.
2. Ecrire la déclaration de la structure qui représente un polynôme sous forme d'une liste doublement chaînée.
3. Ecrire la fonction qui permet de calculer la somme de deux polynômes P et Q.
4. Ecrire la fonction qui permet de calculer le produit deux polynômes P et Q.
5. Ecrire la fonction qui permet de trouver les solutions réelles dans un intervalle [a, b], si elles existent, de l'équation P(x)=0 où P est un polynôme donné.

Code :

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
   int coef;
   int pui;
}mon;

struct node{
   mon n;
   struct node* suiv;
   struct node* prec;
```

```
};
struct node *head,*tail;
void insert_tail (struct node* head, int coef,int pui){
   struct node *p,*nv;
   nv=(struct node*)malloc(sizeof(struct node));
   nv->n.coef=coef;
   nv->n.pui=pui;
   nv->prec=NULL;
   nv->suiv=NULL;
   if (head==NULL)
   {
      head=nv;
      tail=nv;
      nv->suiv=head;
   }
   else {
      p=head;
      while (p!=NULL && p->n.pui > pui)
      {
         p=p->suiv;
      }
      nv->suiv=p->suiv;
      nv->prec=p;
      p->suiv=nv;
   }
}
 struct node* create_polynome (struct node* head,int n){
  struct node* nv;
  int i,c,p;
  for ( i = 0; i < n; i++)
  {
   printf("terme %d : ",i+1);
   printf("donnez le coefficient : ");
   scanf("%d",&c);
   printf("donnez la puissance : ");
   scanf("%d",&p);
   insert_tail(head,c,p);
  }
}

struct node* addition(struct node* head, struct node* p1, struct node* p2){
   struct node* p3;
   p3=(struct node*)malloc(sizeof(struct node));
   p3->prec=NULL;
   p3->suiv=NULL;
   while(p1 != NULL && p2 != NULL){
      if(p1->n.pui == p2->n.pui){
         insert (head,p1->n.coef+p2->n.coef,p1->n.pui);
         p1=p1->suiv;
```

```
            p2=p2->suiv;
        }
        else if (p1->n.pui > p2->n.pui){
            insert (head,p1->n.coef,p1->n.pui);
            p1=p1->suiv;
            p2=p2->suiv;
        }
        else{
             insert (head,p2->n.coef,p2->n.pui);
            p1=p1->suiv;
            p2=p2->suiv;
        }

    }
    if (p1==NULL || p2== NULL){
        while(p1 != NULL){
             insert (head,p1->n.coef,p1->n.pui);
            p1=p1->suiv;
        }
        while(p2 != NULL){
             insert (head,p2->n.coef,p2->n.pui);
            p2=p2->suiv;
        }
    }
   return p3;
    }

struct node* Multiplication(struct node* head, struct node* p1, struct node* p2){
    struct node* p3;
    p3=(struct node*)malloc(sizeof(struct node));
    p3->prec=NULL;
    p3->suiv=NULL;
    while(p1 != NULL && p2 != NULL){
        if(p)

}
```