```c
#include <conio.h>
#include<stdio.h>
#include <string.h>
#include<stdlib.h>
 #define N 5
 char tmotscles[N][80]={"entier","reel","repeter","si","sinon"};
typedef struct point{
    int x,y;
}point;
//-----------------------------------------------------------------------------
//question A1
int comment(char instr[]){
    if(instr[0]=='\\' && instr[1]=='*' && instr[strlen(instr)-2]=='*' && instr[strlen(instr)-1]=='\\')
    return 1;
    else
        return 0;
}
//-----------------------------------------------------------------------------
// question A2
 void supprim_espaces(char instr[]){
int i,j,compte;
i=0;
while (instr[i]!='\0')
{j=i;
if (instr[i]==' ')
    while(instr[j]!='\0')
{
    instr[j]=instr[j+1];
    j++;
}
if(i==j)
    i++;
else
    instr[j]='\0';
}
//instr[i]='\0';
 }
 //-----------------------------------------------------------------------------

 //-----------------------------------------------------------------------------

 // question B1

 int motcle(char mot[]){
     int i=0;
    while(strcmp(mot,tmotscles[i])!=0 && i<N)
        i++;
    if(i==N)
        return 0;
    else
        return 1;

 }
 //-----------------------------------------------------------------------------
 // question B2 le mieux est de construire les fonctions suivantes estalphabet -charminuscule-minuscule -contientespace
 char charminuscule(char x){
 if(x>='A' && x<='Z')
    x=x+32;
 return x;
 }
 //========
 char * minuscule(char T[]){
 int i;
 for (i=0;i<strlen(T);i++)
    T[i]=charminuscule(T[i]);
 return T;
  }
 //============
 int contientespace(char T[]){
 int i;
 for (i=0;i<strlen(T);i++)
    if(T[i]==32)
        return 1;
 return 0;
 }
 //===========
 int estalphabet(char T[]){
 int i;
 char m;
 for (i=0;i<strlen(T);i++){
 m=charminuscule((T[i]));
    if(m<'a'  || m >'z')
```

```
        return 0;
 }
 return 1;
  }
  //============================
  int commenceparchiffre(char T[]){
   return (T[0]>='0' && T[0]<='9');
  }
  //============
  int identificateur(char id[]){
  if(!commenceparchiffre(id) && !contientespace(id)&& strlen(id)<=80 && !motcle(id))
     return 1;
     else
     return 0;
  }
 //-------------------------------------------------------------------
 //question c1
 //-----------------------------------------------------
 int analyseinstruction(char instr[]){
 int i=0;
 char s[80];
 if(comment(instr))
     return 1;
 else {
        while(instr[i]!=' ' && i<strlen(instr))
        {s[i]=instr[i];
        i++;}
 if(instr[i]==' '){
        s[i]='\0';
     if(motcle(s) && instr[strlen(instr)-1]==';')
        return 1;
 }

 return 0;
 }
 }
 //-----------------------------------------------
 //1er variante sans mémorisation
 void analyseSource(char source[]){
 FILE * fich;

 int nb=0;
 int succes=1;
 char ligne[200];
 fich=fopen(source,"r");

 while(fgets(ligne,200,fich)){
     if(!analyseinstruction(ligne)){
        succes=0;
        printf("%d\n",nb+1);
     }
     nb++;
 }
 fclose(fich);
 if(succes)
     printf("Succès : 0 Erreurs\n");
  }
 //-------------------------------------------------------------------------
 //2eme variante
 void analyseSource1(char source[]){
 FILE * fich;
 int lnerreur[200];
 int nb,j;
 int succes=1;
 char ligne[200];
 for (nb=0;nb<200;nb++)
     lnerreur[nb]=-1;
 nb=0;j=0;
 fich=fopen(source,"r");

 while(fgets(ligne,200,fich)){
     if(!analyseinstruction(ligne)){
        succes=0;
        lnerreur[j]=nb+1;
        j++;
     }
     nb++;
 }
 fclose(fich);
 if(succes)
     printf("Succès : 0 Erreurs\n");
 else{
   printf("les numéros des lignes correspondantes à des instructions incorrectes sont:");
 j=0;
```

```
while(lnerreur[j]!=-1 && j<200)
{
    printf("%d",lnerreur[j]);
    if(lnerreur[j+1]!=-1 && j+1<200)
        printf(",");
    j++;
}
}
}
//#########################################################################################
//deuxieme probleme
//#########################################################################################
//question  A1

point c[N];
//-----------------------------------------
void initialiserC(){
int i;point P;
P.x=-1;
P.y=-1;
for (i=0;i<N;i++)
{
    c[i]=P;
}
}
//-----------------------------------------------------
//question A2
void cheminHV(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++;
        }

if(A.y<B.y)
    for (i=A.y;i<=B.y;i++)
        {c[j].y=i;
         c[j].x=B.x;
        j++;}
else
        for (i=A.y;i>=B.y;i--)
        {c[j].y=i;
         c[j].x=B.x;
        j++;
        }


}
//-------------------------------------------------------------------------
/*
On prend des points pour que le procedures puissent s'executer correctement dans codeBlocks
Point A,B  et le tableau tabC
*/
point A={1,1};
point B={5,7};
#define NC 3

point tabC[NC][11]={{{1,1},{1,2},{1,3},{1,4},{1,5},{1,6},{1,7},{2,7},{3,7},{4,7},{5,7}},
            {{1,1},{2,1},{3,1},{4,1},{5,1},{5,2},{5,3},{5,4},{5,5},{5,6},{5,7}},
            {{1,1},{1,2},{2,2},{3,2},{3,3},{4,3},{4,4},{5,4},{5,5},{5,6},{5,7}}};



//*Question A-3-a
int distance(int num){
int d=0;
while(tabC[num][d].x!=-1 && tabC[num][d].y!=-1 && d<11)
    d++;
return d;
}
//-----------------------------------------
int distancemin(){
int Min=0;int i;
for(i=0;i<NC;i++)
    if(distance(Min)>distance(i))
```

```
            Min=i;
return distance(Min);
}
////////////////////////////////////////////////section B//////////////////////////////////
typedef struct typechemin{
point p;
struct typechemin *suiv;
}cheminListe;
cheminListe *deb=NULL;
// on prendra le chemin de AversB celui minimale de la section A: tabC[2]
// chargement de la la liste chainée avec utilisation de A vers B
cheminListe * AversB(){
    int d=0;
    cheminListe *p,*r,*l=NULL;
    p=l;
while(tabC[2][d].x!=-1 && tabC[2][d].y!=-1 && d<11){
    r=(cheminListe*)malloc(sizeof(cheminListe));
    r->p.x=tabC[2][d].x;
    r->p.y=tabC[2][d].y;
    r->suiv=NULL;
    //p marque le dernier élément
    if(l==NULL){
        l=r;p=l;}
    else{
        p->suiv=r;
        p=r;
        }
         d++;
}
return l;

}


//1111111111111111111111111111111111111111111111111111111111111111
cheminListe * BversA(){
cheminListe *l,*p,*r,*l1=NULL;
l=AversB();
p=l;
while(p!=NULL){
    r=(cheminListe*)malloc(sizeof(cheminListe));
    r->p=p->p;
    r->suiv=NULL;
    if(l1==NULL)
        l1=r;
    else{
        r->suiv=l1;
        l1=r;
        }
        p=p->suiv;
}
return l1;
}
//22222222222222222222222222222222
void afficheChemin(cheminListe *deb){
if(deb){
printf("P(%d,%d)",deb->p.x,deb->p.y);
if(deb->suiv)
    printf(",");
afficheChemin(deb->suiv);
}
}

//3333333333333333333333333333333""
//fonction donnant un chemin quelconque de A vers B avec A et B choisis quelconques
//et sera utilisé à deux reprises dans la fonction suivante
cheminListe * AtoB(point A,point B){
  int d=0;
    cheminListe *p,*r,*l=NULL;

     p=(cheminListe*)malloc(sizeof(cheminListe));
     p->p=A;p->suiv=NULL;
     l=p;
while(p->p.x!=B.x || p->p.y!=B.y){
    r=(cheminListe*)malloc(sizeof(cheminListe));
    r->suiv=NULL;
    r->p=p->p;
    p->suiv=r;
    if(p->p.x>B.x)
    {
        r->p.x--;
    }
    else
```

```
334         if(p->p.x<B.x)
335         {
336             r->p.x++;
337         }
338         else
339         if(p->p.y>B.y){
340             r->p.y--;
341         }
342         else
343             if(p->p.y<B.y)
344         {
345             r->p.y++;
346         }
347             p=r;
348     }
349
350
351     return l;
352
353     }
354
355     cheminListe *cheminRepere(point A,point B,point R){
356     cheminListe *l,*l1,*p;
357     l=AtoB(A,R);
358     l1=AtoB(R,B);
359     p=l;
360     while(p->suiv!=NULL)
361         p=p->suiv;
362     p->suiv=l1->suiv;free(l1);
363     return l;
364     }
365
366
367     //------------------------------------------------------------------------------
368     main(){
369     //char S[100];
370     //printf("Saisir une chaine\n");
371     //gets(S);
372     //1er question
373     //printf("%s",(comment(S))?"Chaine commentaire":"chaine non commentaire");
374
375     //question A2
376     //supprim espaces(S);
377     //printf("texte sans espace : %s",S);
378
379     //printf("%s\n",analyseinstruction(S)?"c'est un identiificateur":"c'est pas un
        identificateur");
380     //derniere question de la partie R
381     cheminListe *deb;
382     point A1={3,6},B1={2,8},R1={0,1};
383     //deb=AversB();
384     deb=cheminRepere(A1,B1,R1);
385     afficheChemin(deb);
386     //printf("distance minimale :%d",distancemin());
387     getch();
388     }
389
```