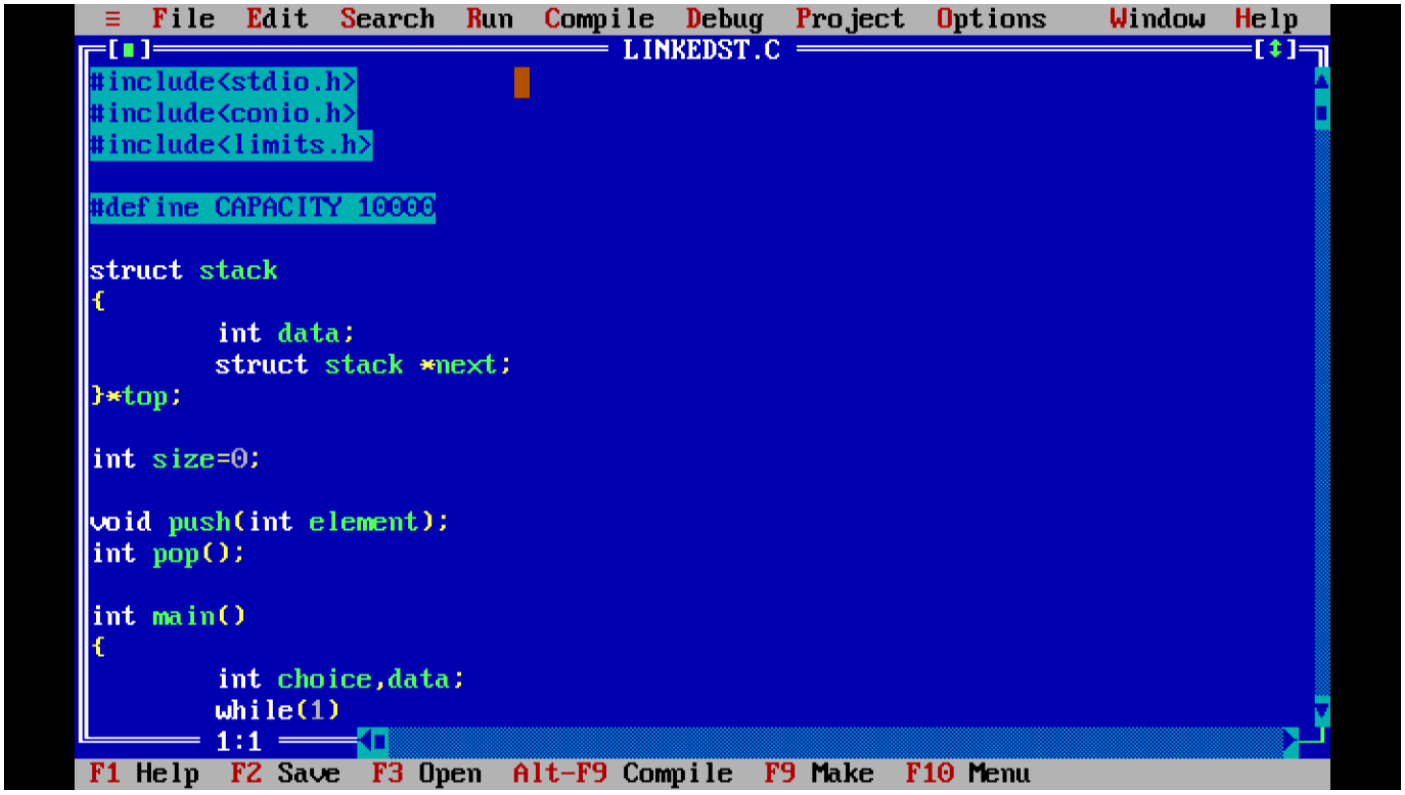
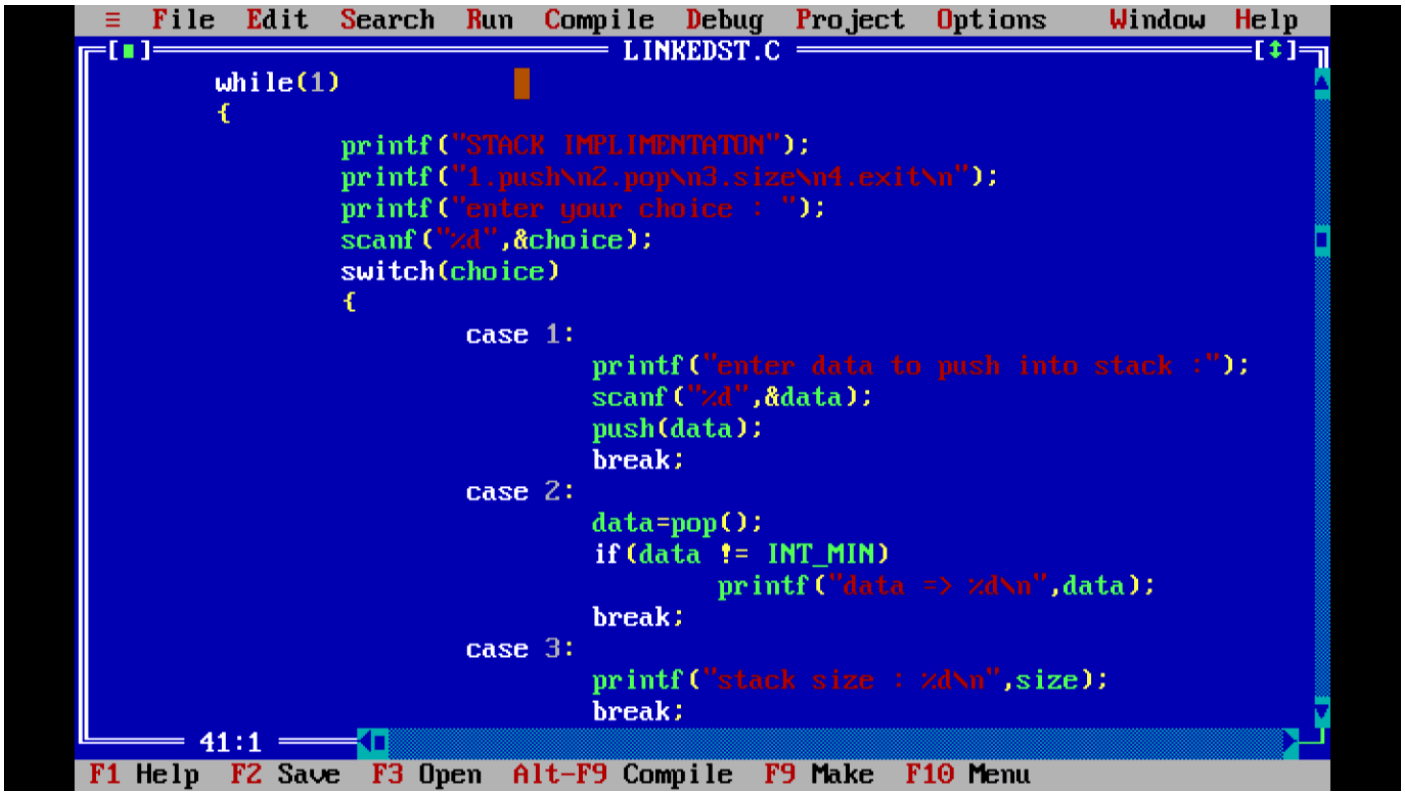
1. **Implement linked stack**

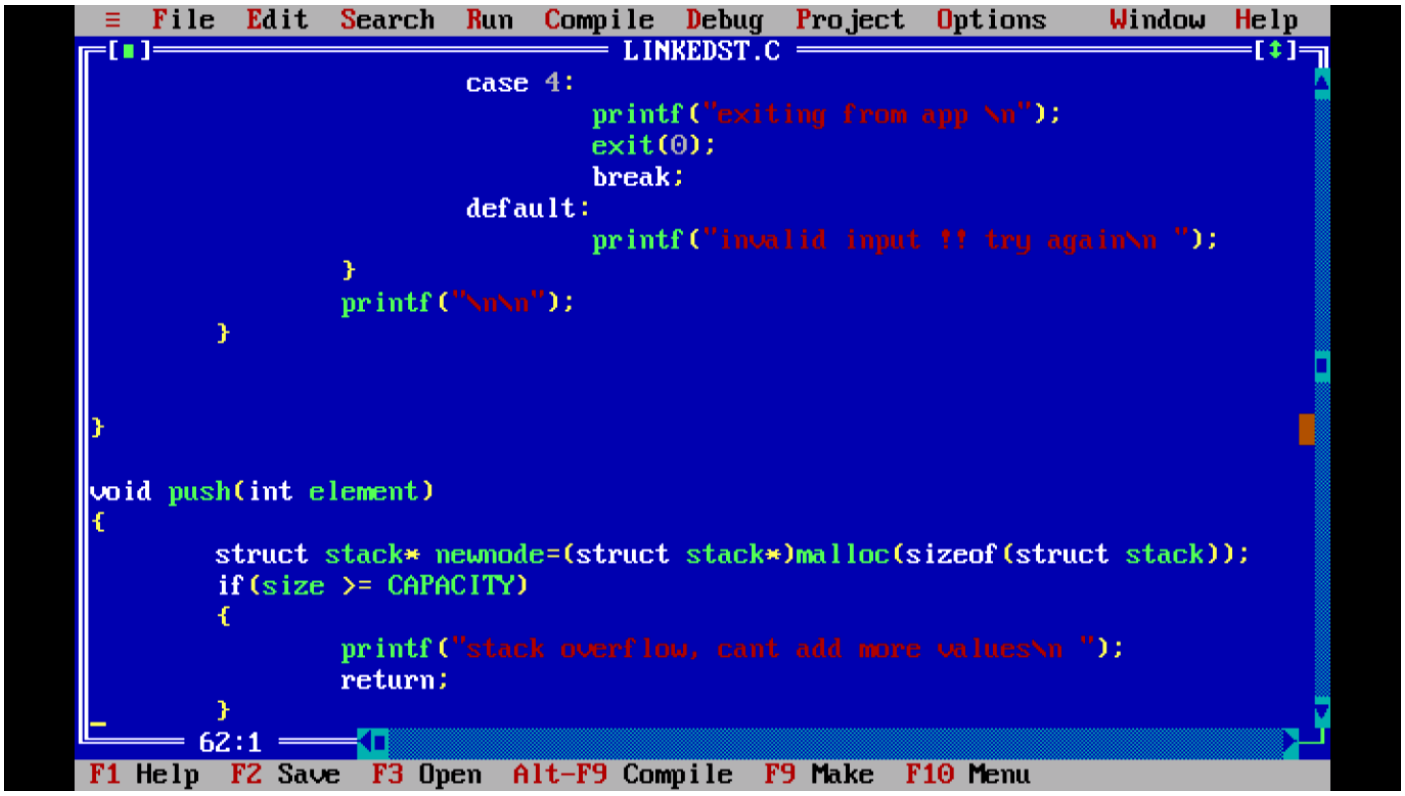
PROGRAM

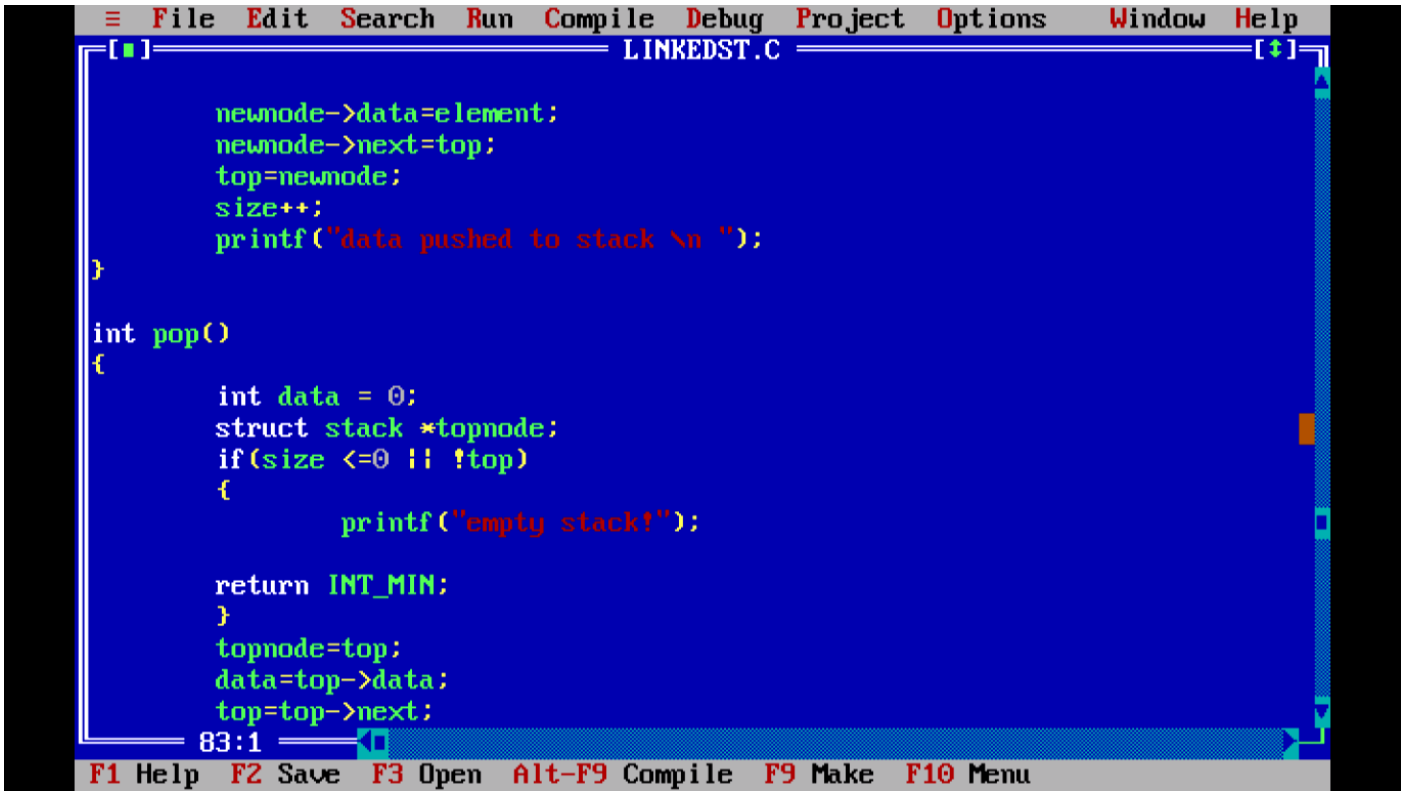


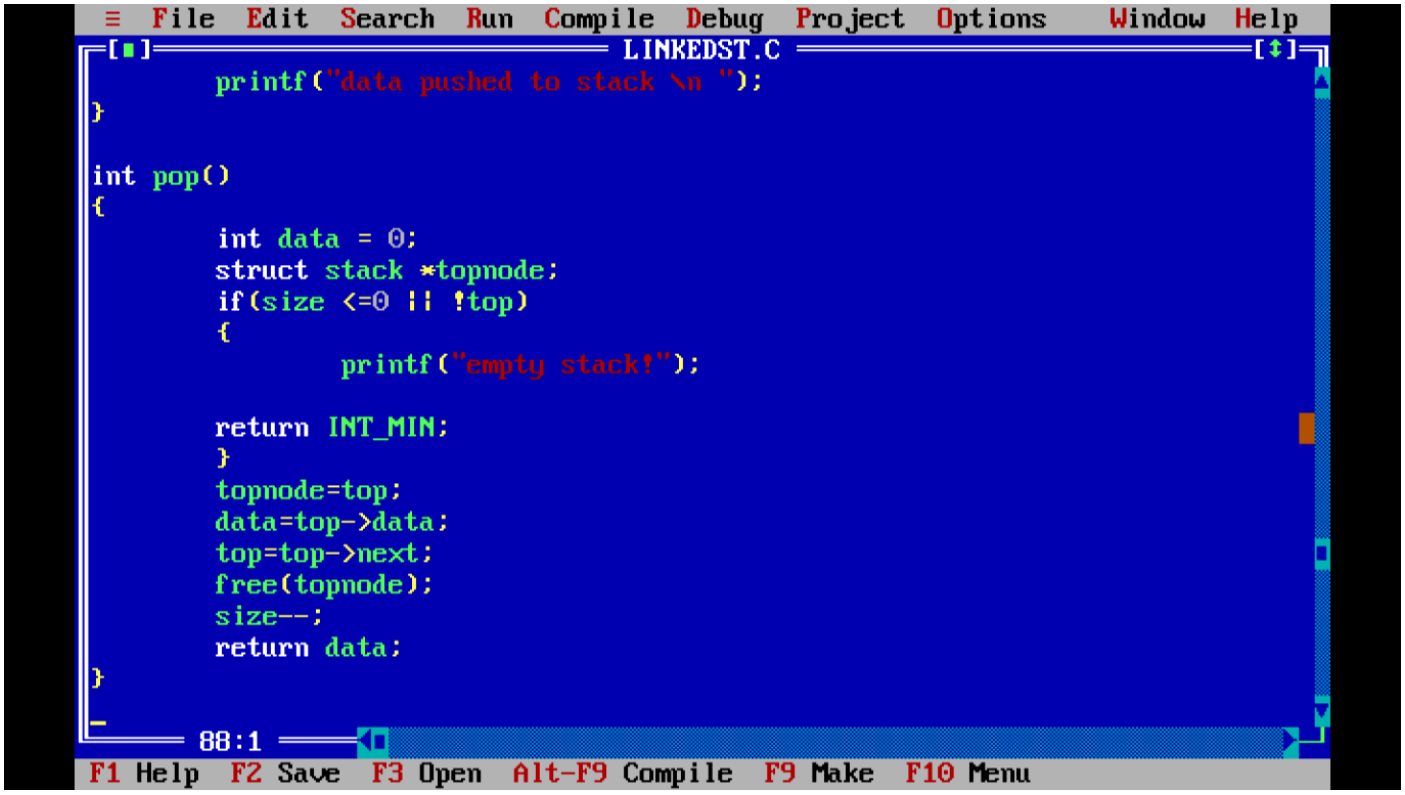
```
File  Edit  Search  Run  Compile  Debug  Project  Options  Window  Help
                        LINKEDST.C
#include<stdio.h>
#include<conio.h>
#include<limits.h>

#define CAPACITY 10000

struct stack
{
        int data;
        struct stack *next;
}*top;

int size=0;

void push(int element);
int pop();

int main()
{
        int choice,data;
        while(1)
1:1
F1 Help  F2 Save  F3 Open  Alt-F9 Compile  F9 Make  F10 Menu
```



```
File  Edit  Search  Run  Compile  Debug  Project  Options  Window  Help
                        LINKEDST.C
        while(1)
        {
                printf("STACK IMPLIMENTATON");
                printf("1.push\n2.pop\n3.size\n4.exit\n");
                printf("enter your choice : ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:
                                printf("enter data to push into stack :");
                                scanf("%d",&data);
                                push(data);
                                break;
                        case 2:
                                data=pop();
                                if(data != INT_MIN)
                                        printf("data => %d\n",data);
                                break;
                        case 3:
                                printf("stack size : %d\n",size);
                                break;
41:1
F1 Help  F2 Save  F3 Open  Alt-F9 Compile  F9 Make  F10 Menu
```


File Edit Search Run Compile Debug Project Options Window Help
LINKEDST.C
case 4:
printf("exiting from app \n");
exit(0);
break;
default:
printf("invalid input !! try again\n ");
}
printf("\n\n");
}
}
void push(int element)
{
struct stack* newmode=(struct stack*)malloc(sizeof(struct stack));
if(size >= CAPACITY)
{
printf("stack overflow, cant add more values\n ");
return;
}
62:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
LINKEDST.C
newmode->data=element;
newmode->next=top;
top=newmode;
size++;
printf("data pushed to stack \n ");
}
int pop()
{
int data = 0;
struct stack *topnode;
if(size <=0 || !top)
{
printf("empty stack!");
return INT_MIN;
}
topnode=top;
data=top->data;
top=top->next;
83:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu



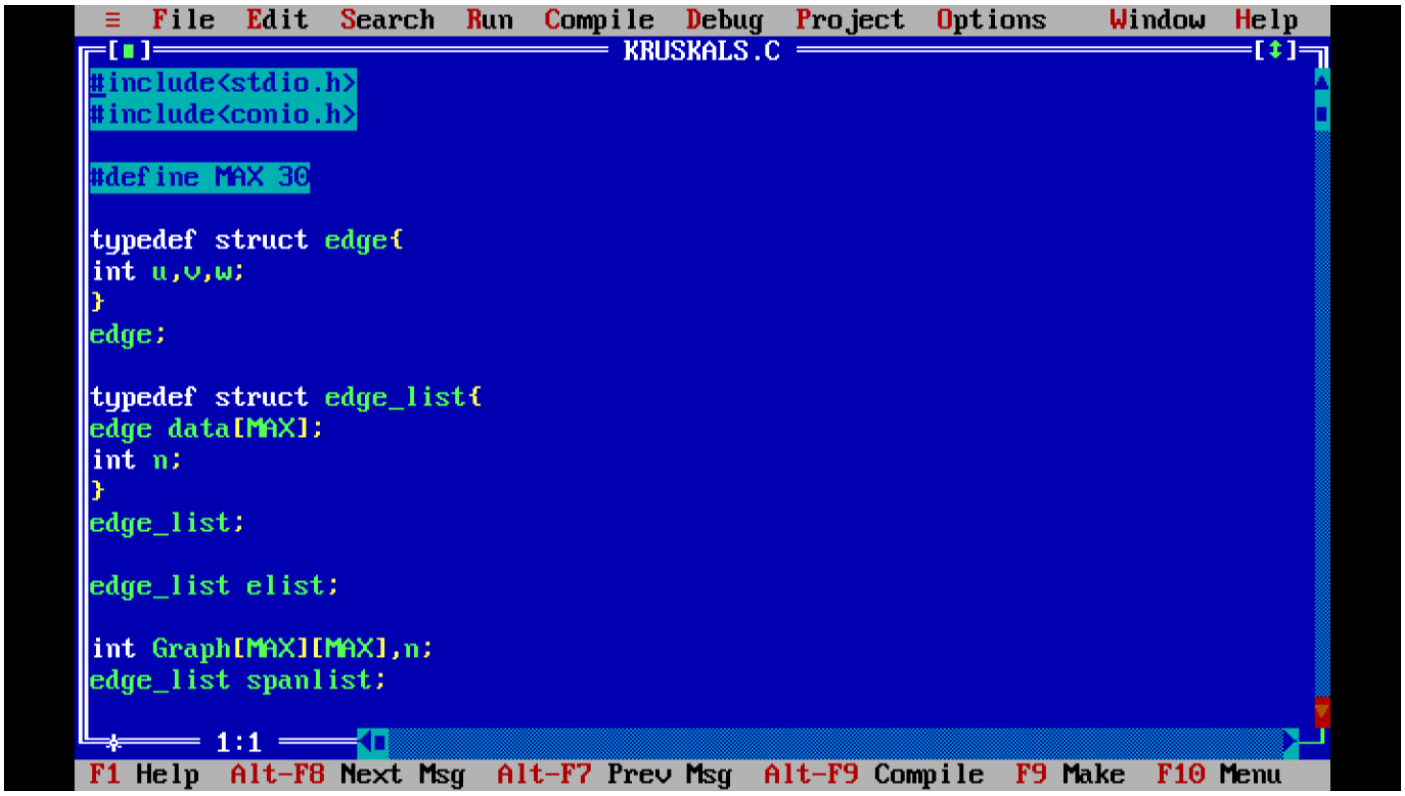
OUTPUT

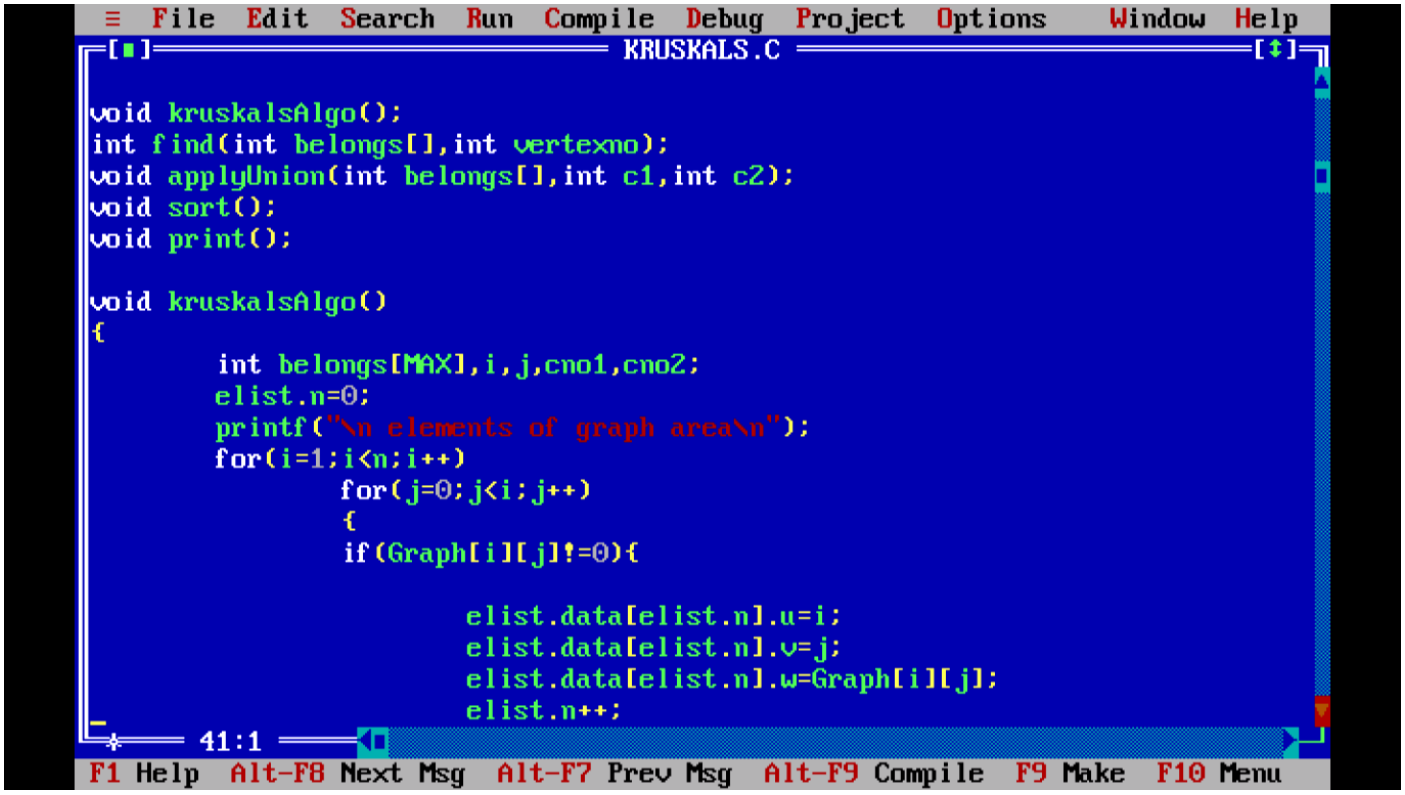
```
C:\TURBOC3\BIN>TC
STACK IMPLIMENTATON1.push
2.pop
3.size
4.exit
enter your choice : 1
enter data to push into stack :10
data pushed to stack


STACK IMPLIMENTATON1.push
2.pop
3.size
4.exit
enter your choice : 1
enter data to push into stack :20
data pushed to stack


STACK IMPLIMENTATON1.push
2.pop
3.size
4.exit
enter your choice : _
```

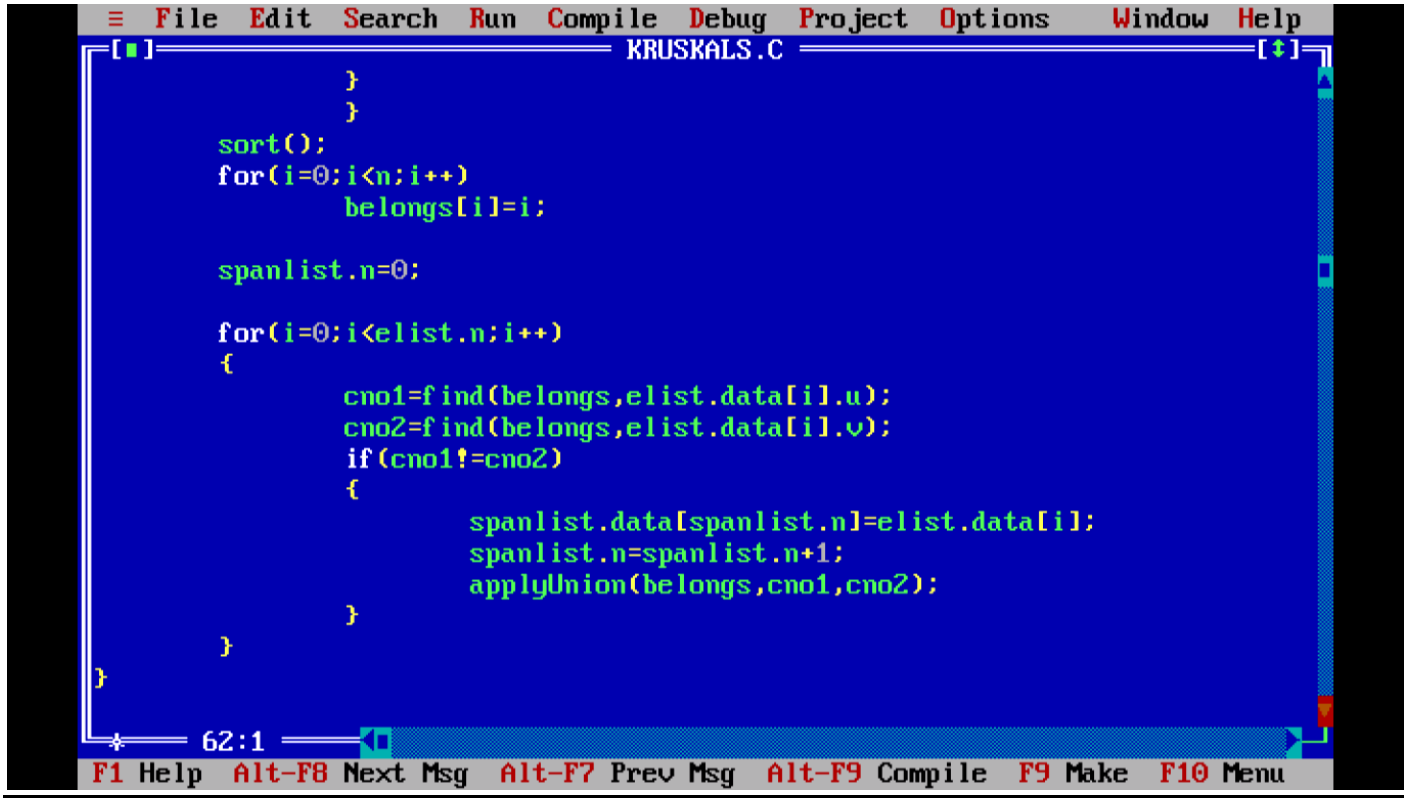
```
enter data to push into stack :20
data pushed to stack


STACK IMPLIMENTATON1.push
2.pop
3.size
4.exit
enter your choice : 3
stack size : 2


STACK IMPLIMENTATON1.push
2.pop
3.size
4.exit
enter your choice : 2
data => 20


STACK IMPLIMENTATON1.push
2.pop
3.size
4.exit
enter your choice :
```

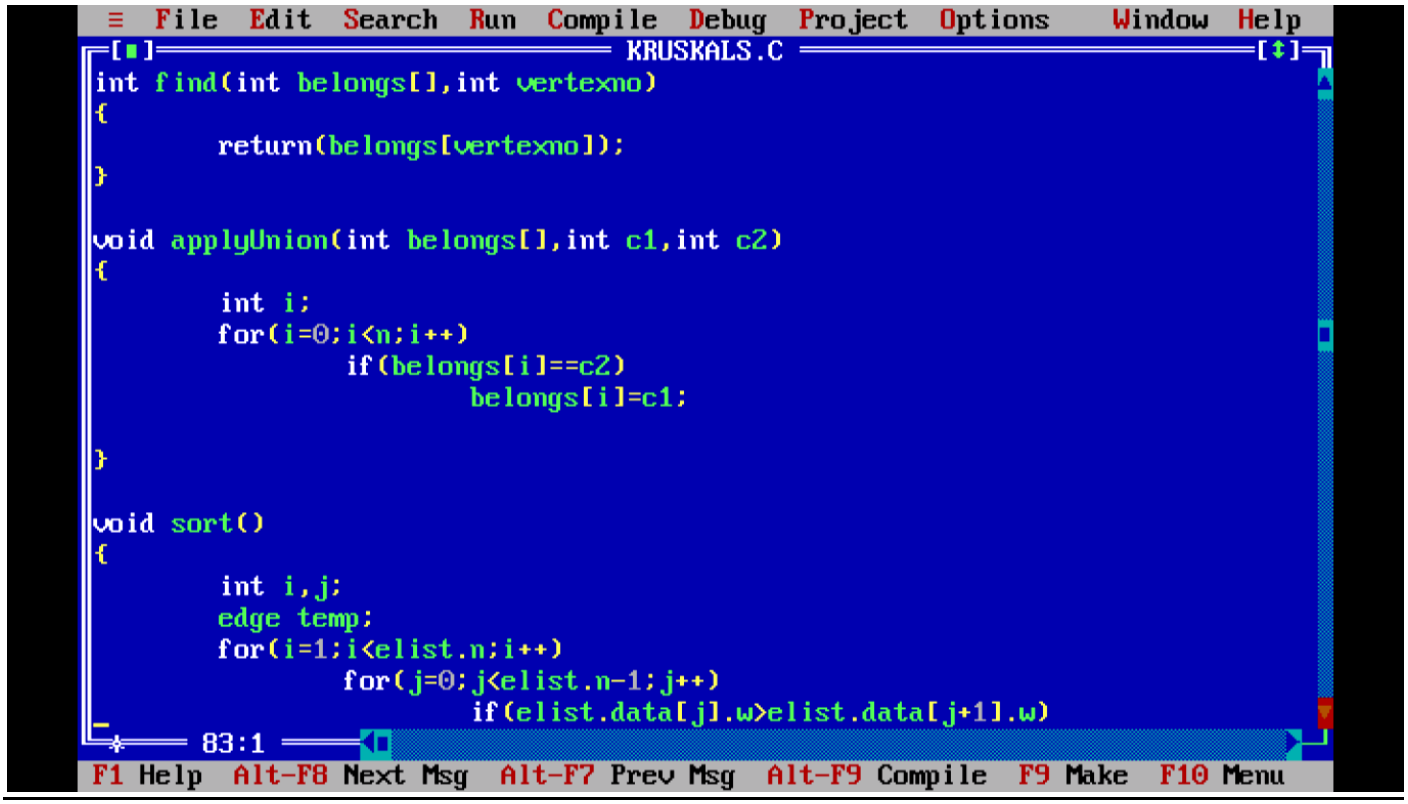
2.Implement Kruskal algorithm

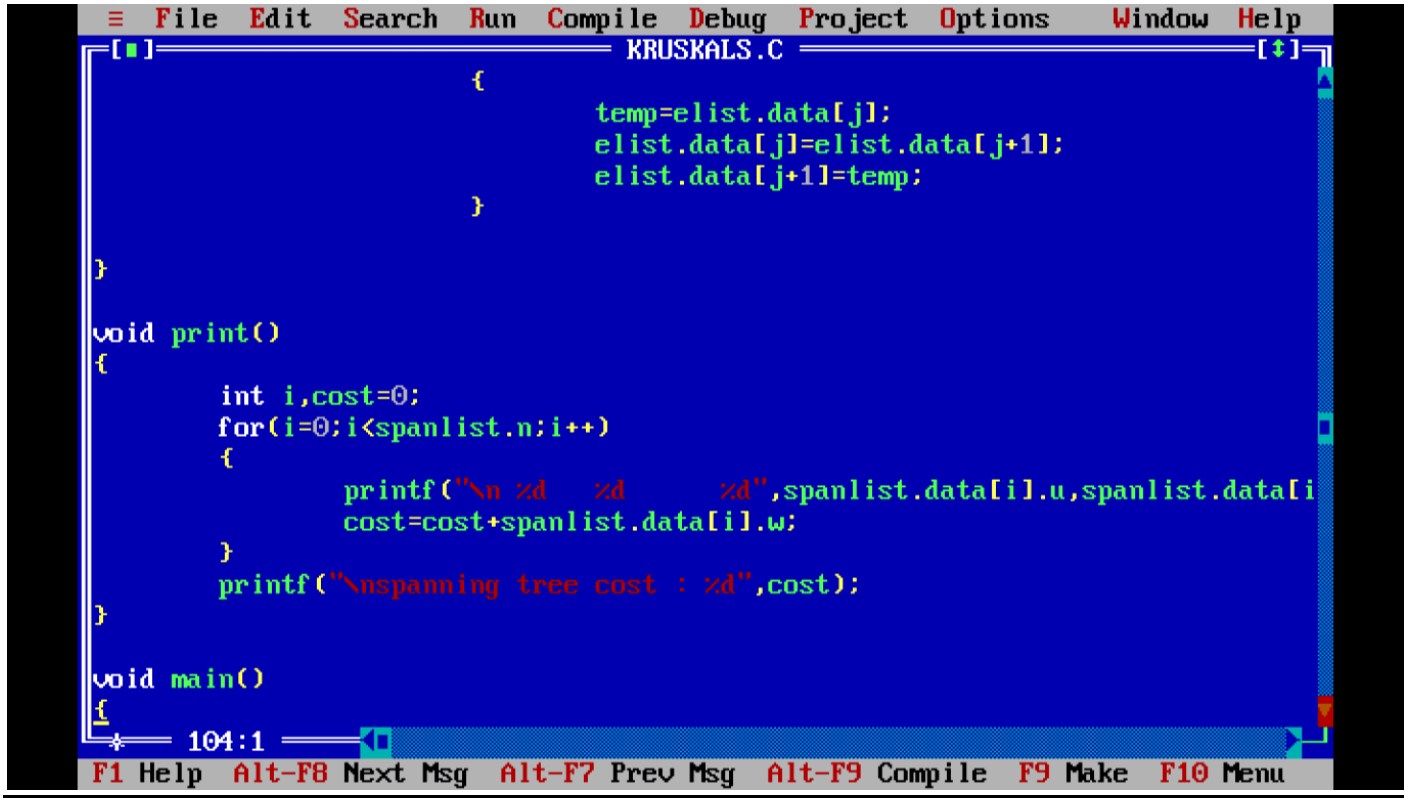
PROGRAM

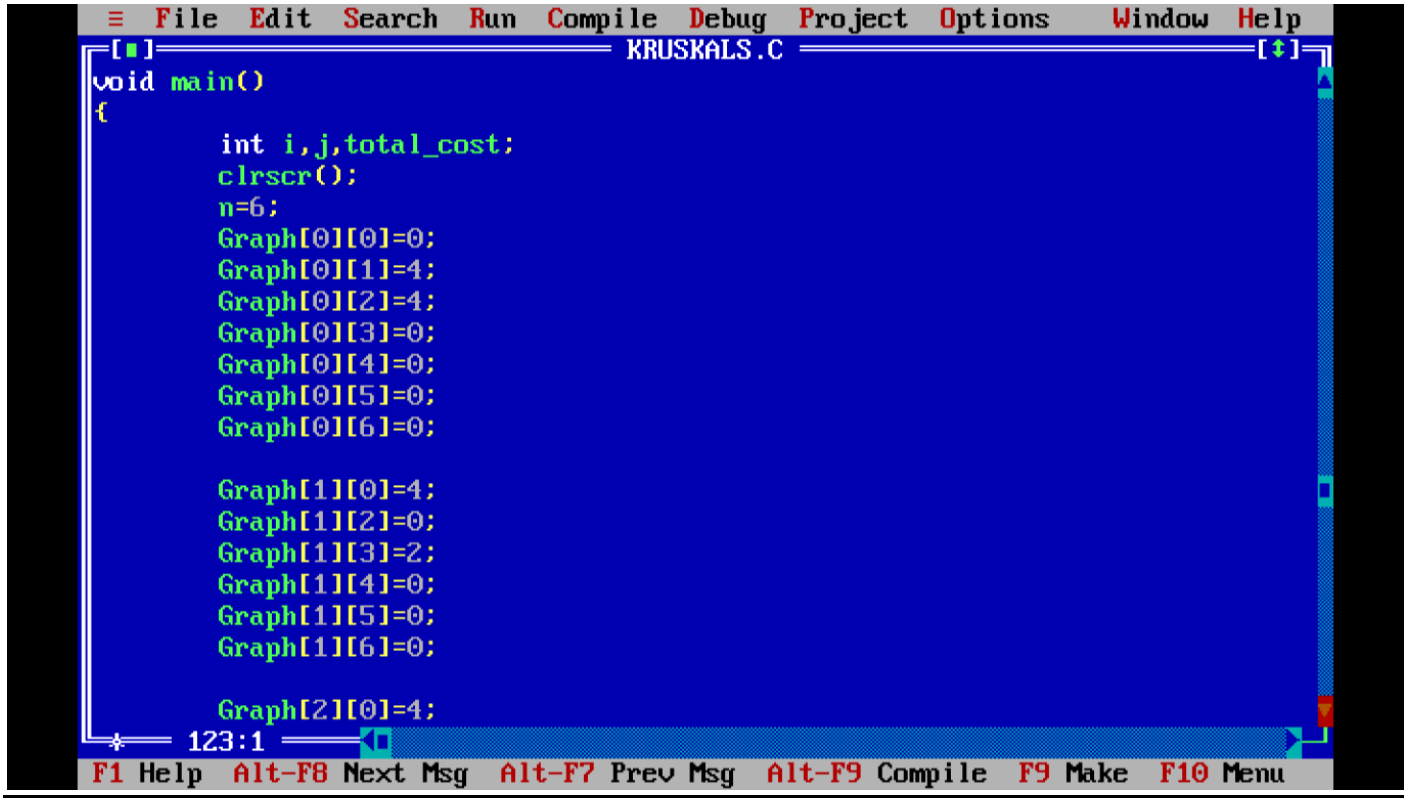


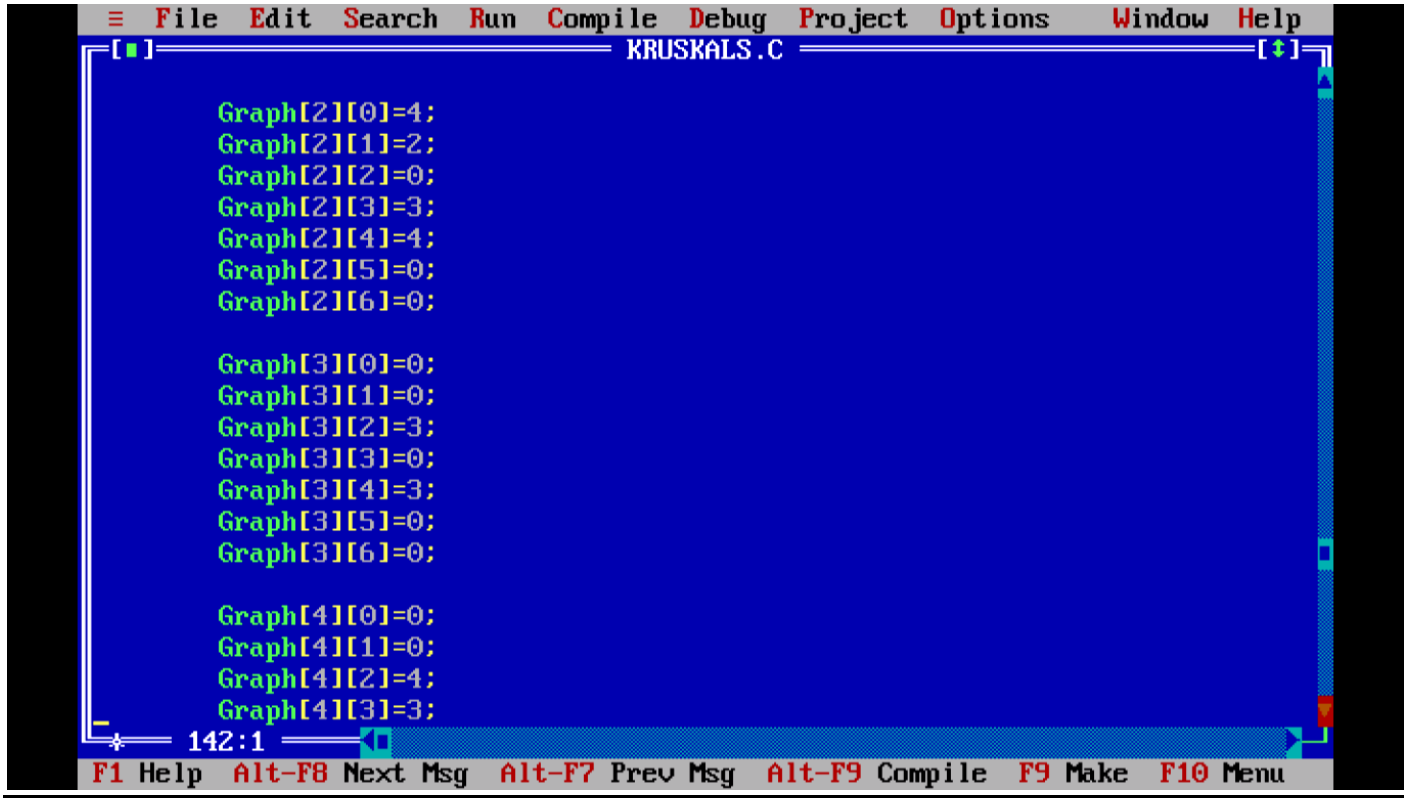


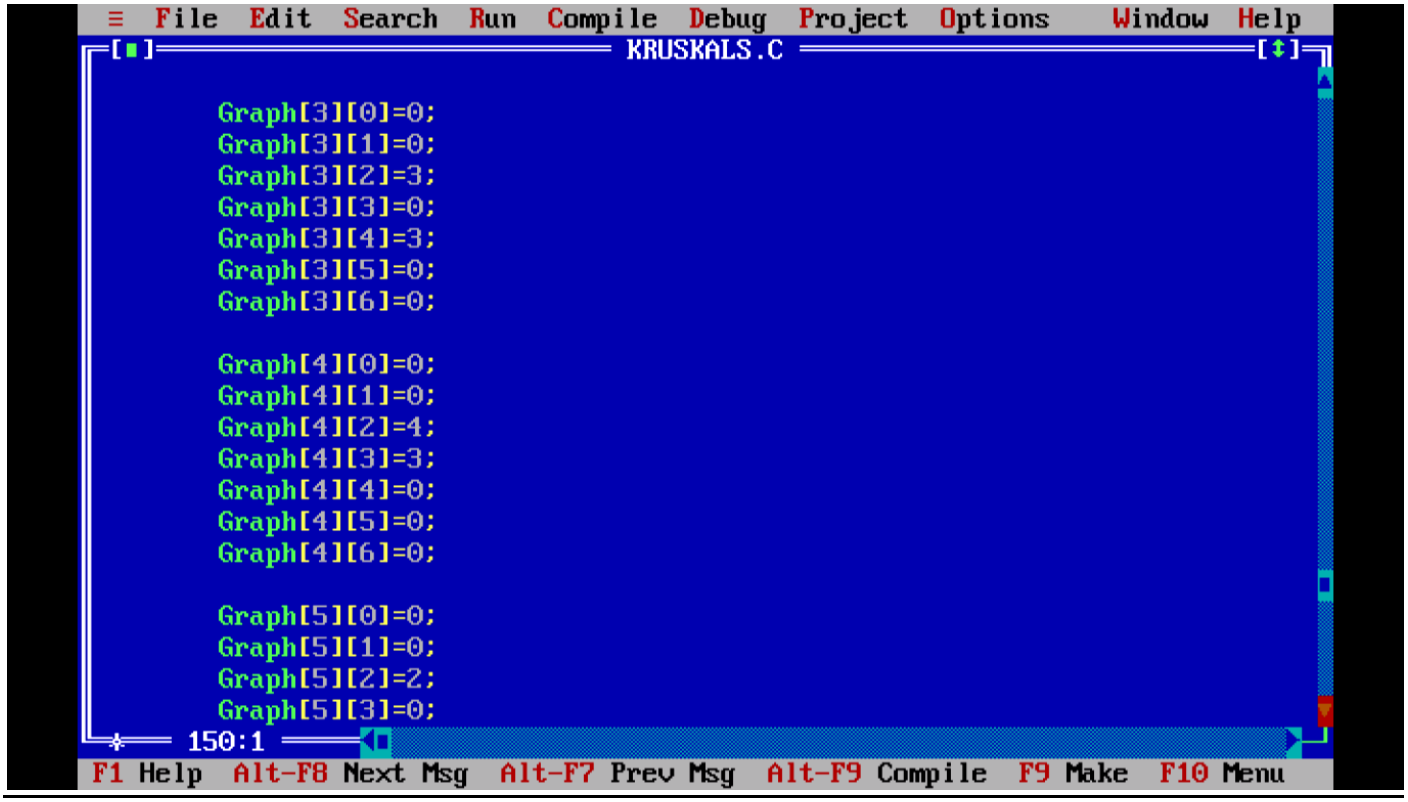

File Edit Search Run Compile Debug Project Options Window Help
KRUSKALS.C
}
}
sort();
for(i=0;i<n;i++)
belongs[i]=i;
spanlist.n=0;
for(i=0;i<elist.n;i++)
{
cno1=find(belongs,elist.data[i].u);
cno2=find(belongs,elist.data[i].v);
if(cno1!=cno2)
{
spanlist.data[spanlist.n]=elist.data[i];
spanlist.n=spanlist.n+1;
applyUnion(belongs,cno1,cno2);
}
}
}
62:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

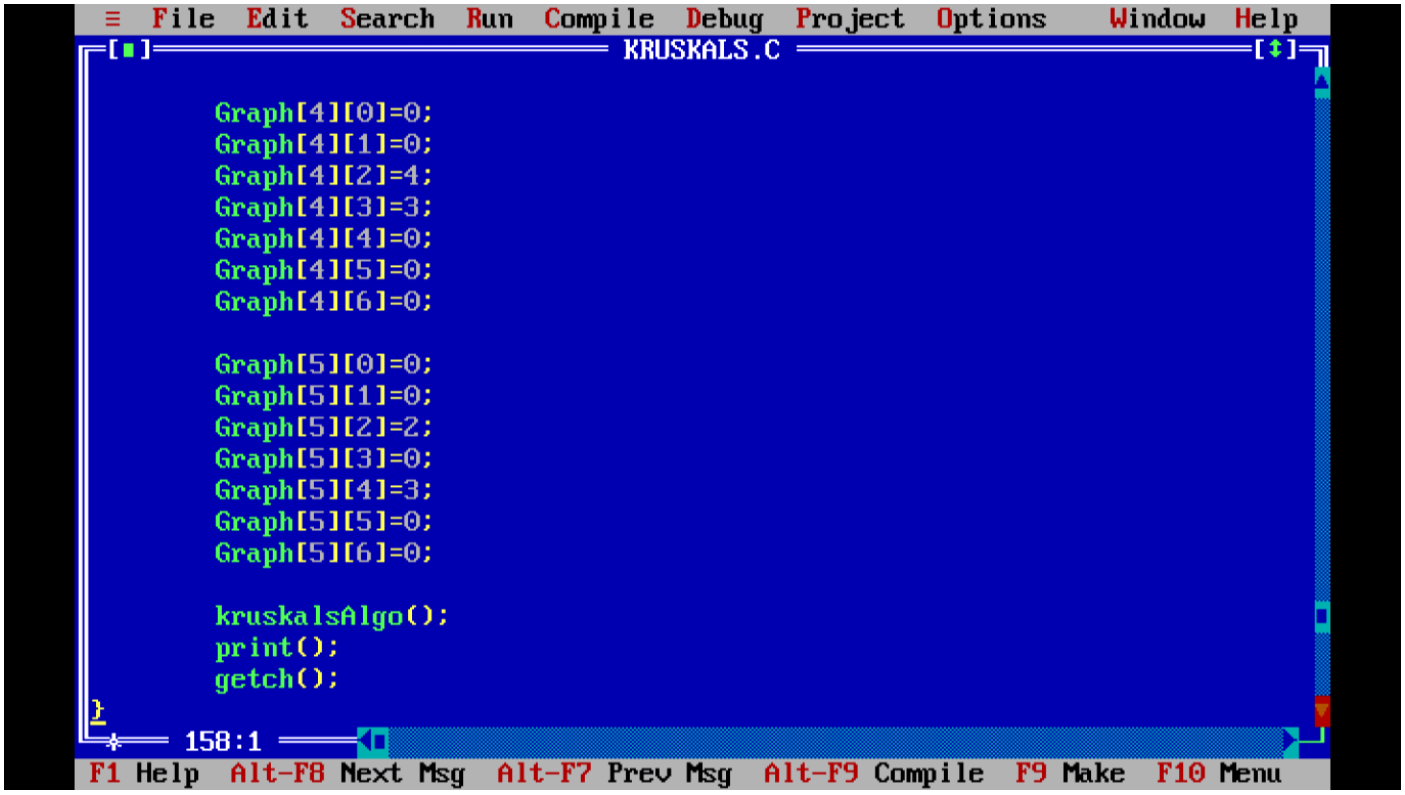

File Edit Search Run Compile Debug Project Options Window Help
KRUSKALS.C
int find(int belongs[],int vertexno)
{
return(belongs[vertexno]);
}
void applyUnion(int belongs[],int c1,int c2)
{
int i;
for(i=0;i<n;i++)
if(belongs[i]==c2)
belongs[i]=c1;
}
void sort()
{
int i,j;
edge temp;
for(i=1;i<elist.n;i++)
for(j=0;j<elist.n-1;j++)
if(elist.data[j].w>elist.data[j+1].w)
83:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
KRUSKALS.C
{
temp=elist.data[j];
elist.data[j]=elist.data[j+1];
elist.data[j+1]=temp;
}
}
void print()
{
int i,cost=0;
for(i=0;i<spanlist.n;i++)
{
printf("\n %d %d %d",spanlist.data[i].u,spanlist.data[i
cost=cost+spanlist.data[i].w;
}
printf("\nspanning tree cost : %d",cost);
}
void main()
{
104:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

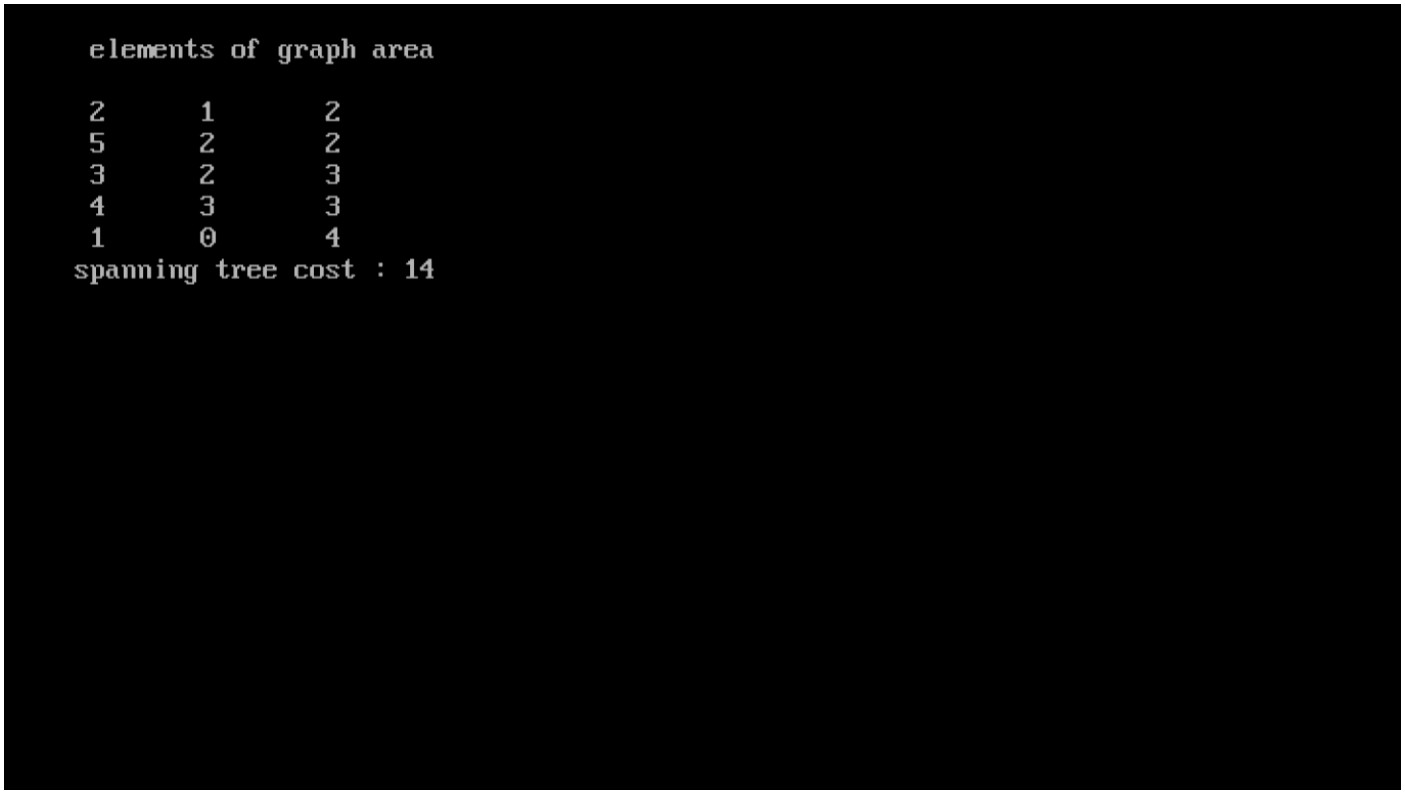

File Edit Search Run Compile Debug Project Options Window Help
KRUSKALS.C
void main()
{
int i,j,total_cost;
clrscr();
n=6;
Graph[0][0]=0;
Graph[0][1]=4;
Graph[0][2]=4;
Graph[0][3]=0;
Graph[0][4]=0;
Graph[0][5]=0;
Graph[0][6]=0;
Graph[1][0]=4;
Graph[1][2]=0;
Graph[1][3]=2;
Graph[1][4]=0;
Graph[1][5]=0;
Graph[1][6]=0;
Graph[2][0]=4;
123:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
KRUSKALS.C
Graph[2][0]=4;
Graph[2][1]=2;
Graph[2][2]=0;
Graph[2][3]=3;
Graph[2][4]=4;
Graph[2][5]=0;
Graph[2][6]=0;
Graph[3][0]=0;
Graph[3][1]=0;
Graph[3][2]=3;
Graph[3][3]=0;
Graph[3][4]=3;
Graph[3][5]=0;
Graph[3][6]=0;
Graph[4][0]=0;
Graph[4][1]=0;
Graph[4][2]=4;
Graph[4][3]=3;
142:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
KRUSKALS.C
Graph[3][0]=0;
Graph[3][1]=0;
Graph[3][2]=3;
Graph[3][3]=0;
Graph[3][4]=3;
Graph[3][5]=0;
Graph[3][6]=0;
Graph[4][0]=0;
Graph[4][1]=0;
Graph[4][2]=4;
Graph[4][3]=3;
Graph[4][4]=0;
Graph[4][5]=0;
Graph[4][6]=0;
Graph[5][0]=0;
Graph[5][1]=0;
Graph[5][2]=2;
Graph[5][3]=0;
150:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
KRUSKALS.C
Graph[4][0]=0;
Graph[4][1]=0;
Graph[4][2]=4;
Graph[4][3]=3;
Graph[4][4]=0;
Graph[4][5]=0;
Graph[4][6]=0;
Graph[5][0]=0;
Graph[5][1]=0;
Graph[5][2]=2;
Graph[5][3]=0;
Graph[5][4]=3;
Graph[5][5]=0;
Graph[5][6]=0;
kruskalsAlgo();
print();
getch();
}
158:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

OUTPUT


elements of graph area
2 1 2
5 2 2
3 2 3
4 3 3
1 0 4
spanning tree cost : 14