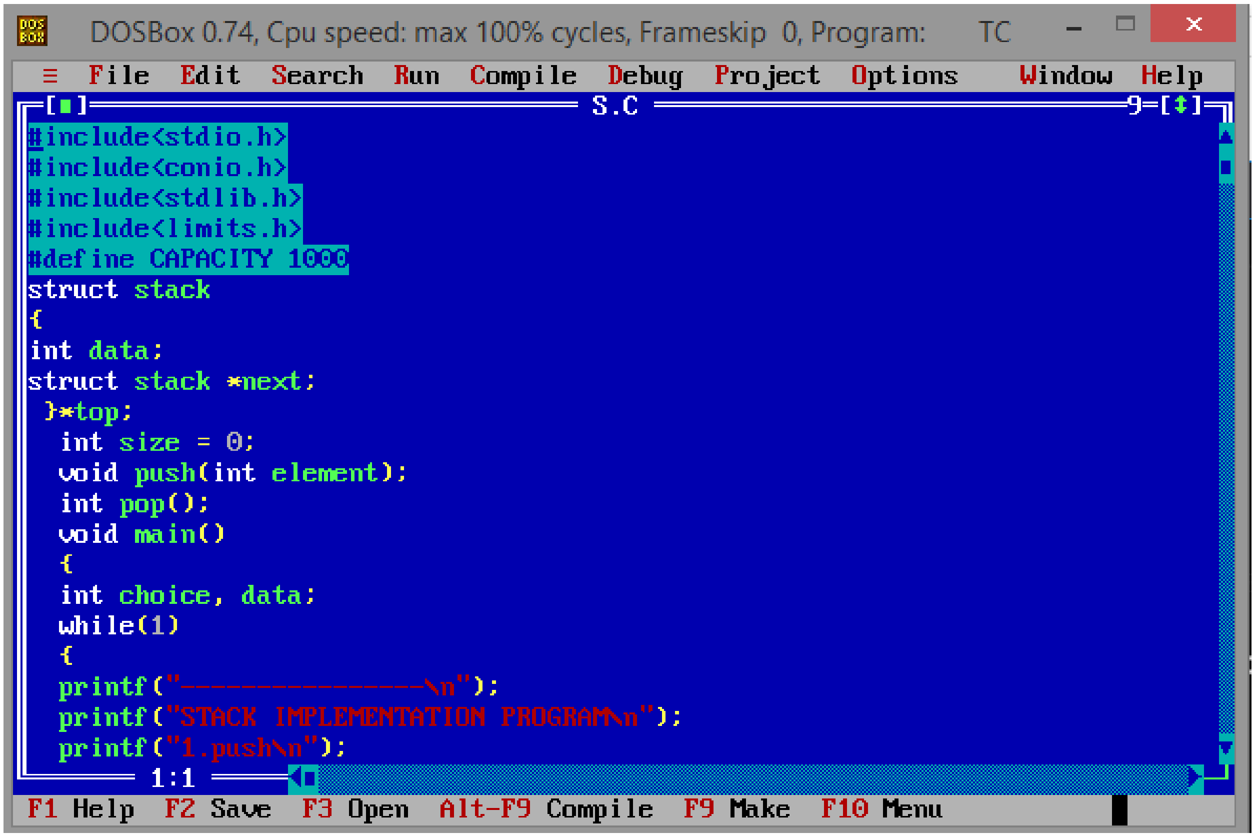
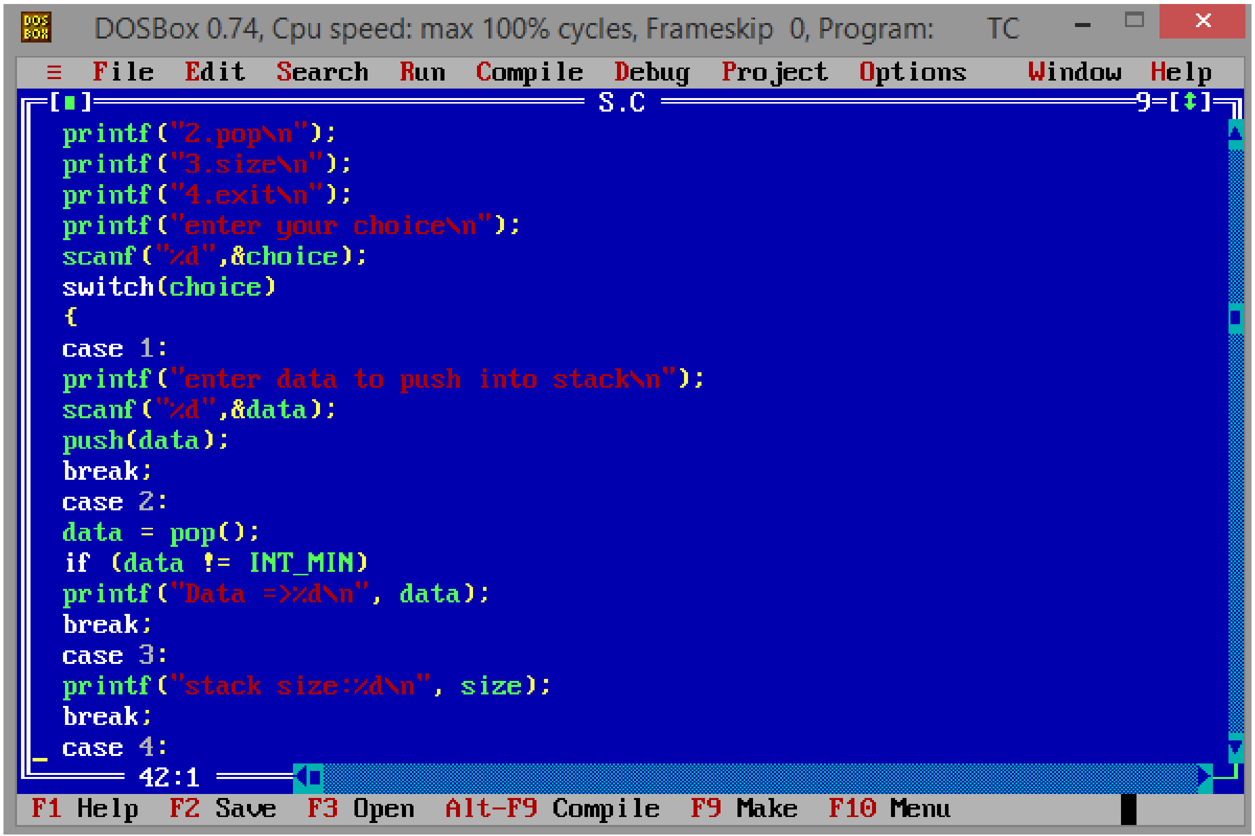
1)**Program to implement liked stack**

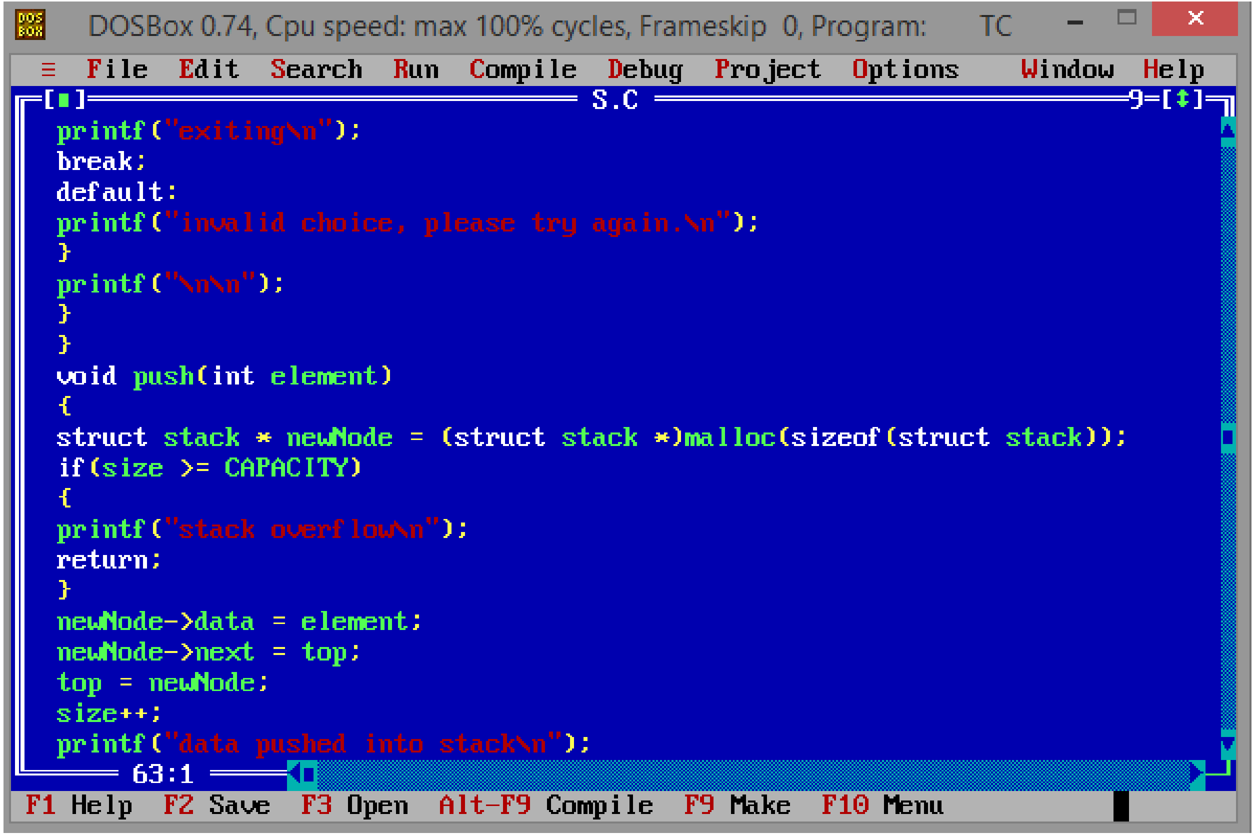
Program code


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
S.C
9
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

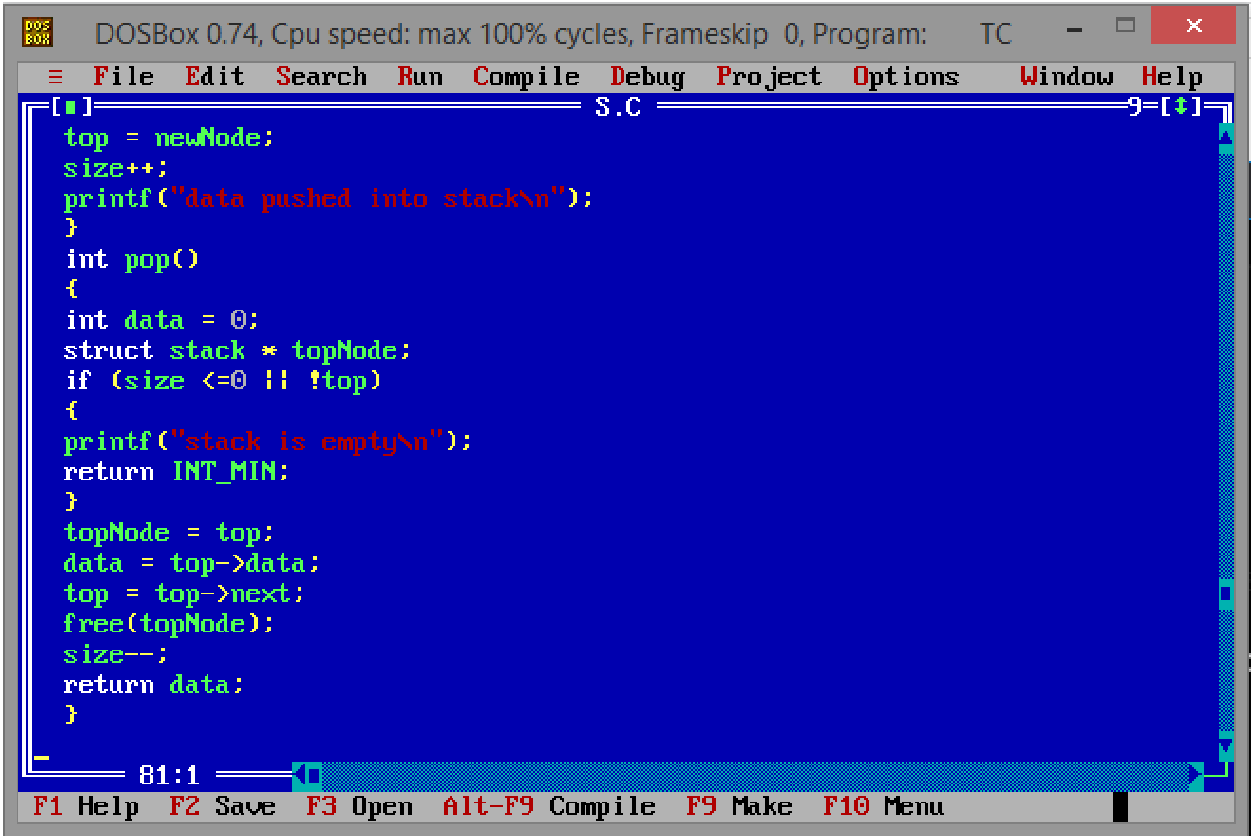
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<limits.h>
#define CAPACITY 1000
struct stack
{
int data;
struct stack *next;
 }*top;
  int size = 0;
  void push(int element);
  int pop();
  void main()
  {
  int choice, data;
  while(1)
  {
  printf("----------------\n");
  printf("STACK IMPLEMENTATION PROGRAM\n");
  printf("1.push\n");
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
[■] S.C 9=[↕]
42:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
printf("2.pop\n");
printf("3.size\n");
printf("4.exit\n");
printf("enter your choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:
printf("enter data to push into stack\n");
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;
case 4:
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
[■] S.C 9=[↕]
63:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

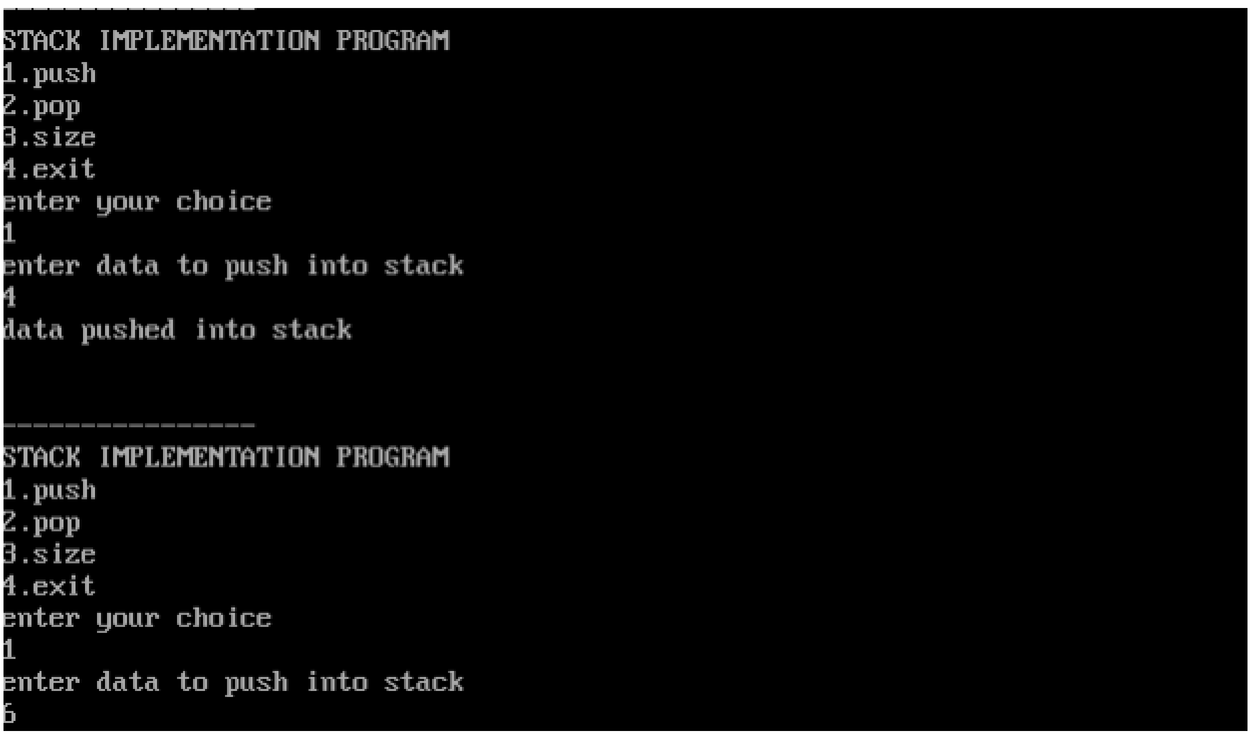
```
printf("exiting\n");
break;
default:
printf("invalid choice, please try again.\n");
}
printf("\n\n");
}
}
void push(int element)
{
struct stack * newNode = (struct stack *)malloc(sizeof(struct stack));
if(size >= CAPACITY)
{
printf("stack overflow\n");
return;
}
newNode->data = element;
newNode->next = top;
top = newNode;
size++;
printf("data pushed into stack\n");
```



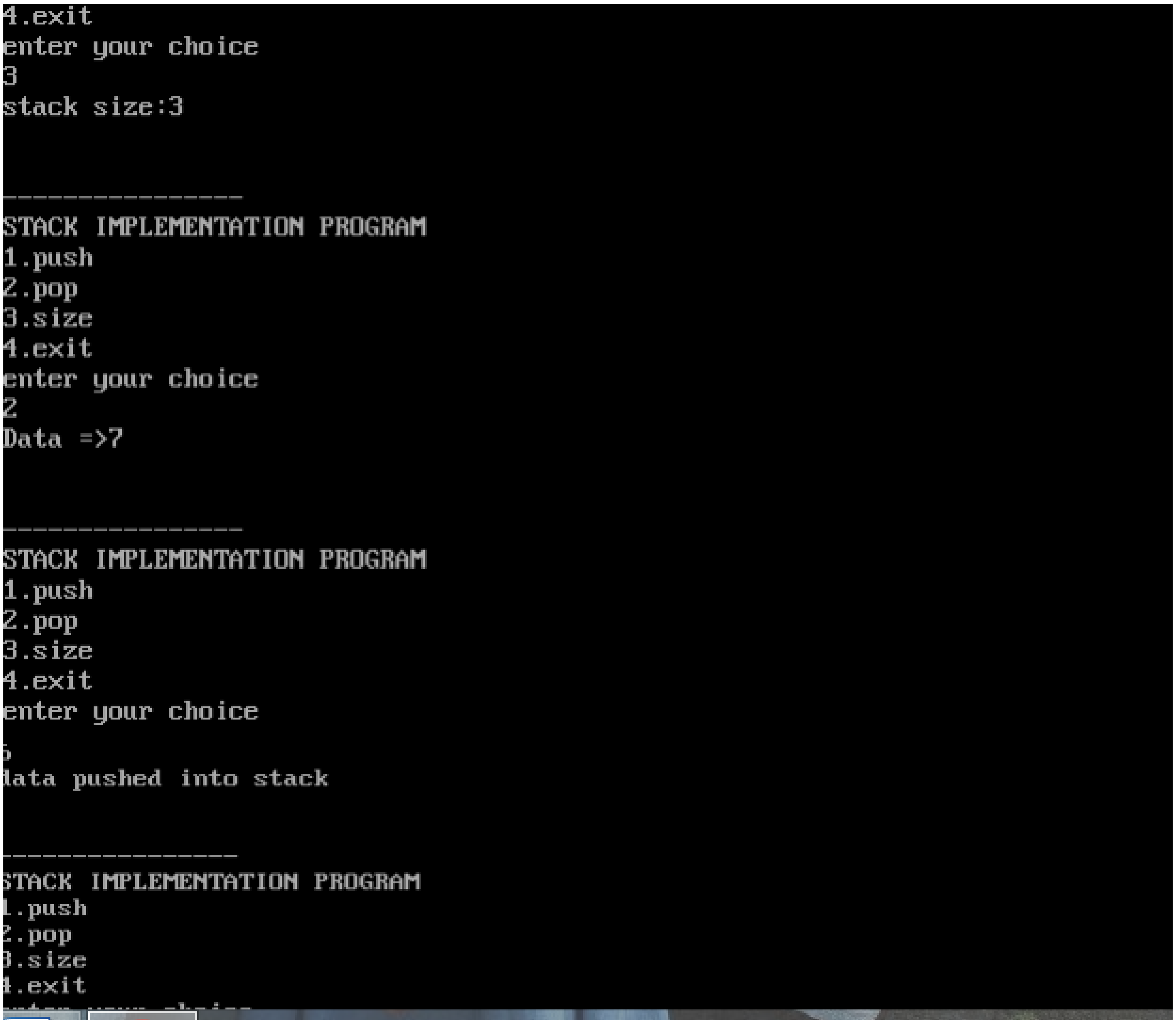
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
S.C
81:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
top = newNode;
size++;
printf("data pushed into stack\n");
}
int pop()
{
int data = 0;
struct stack * topNode;
if (size <=0 || !top)
{
printf("stack is empty\n");
return INT_MIN;
}
topNode = top;
data = top->data;
top = top->next;
free(topNode);
size--;
return data;
}
```

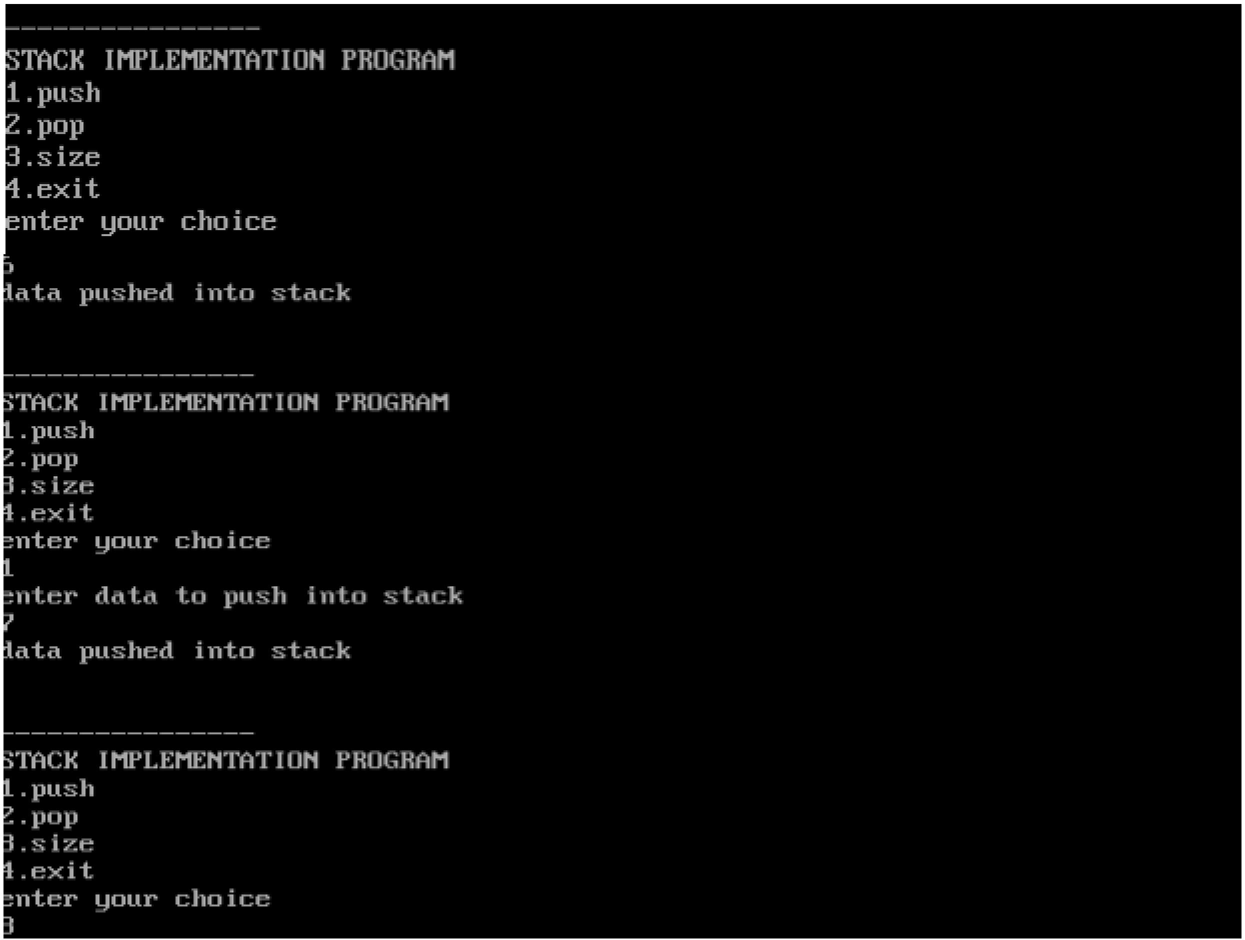
Output



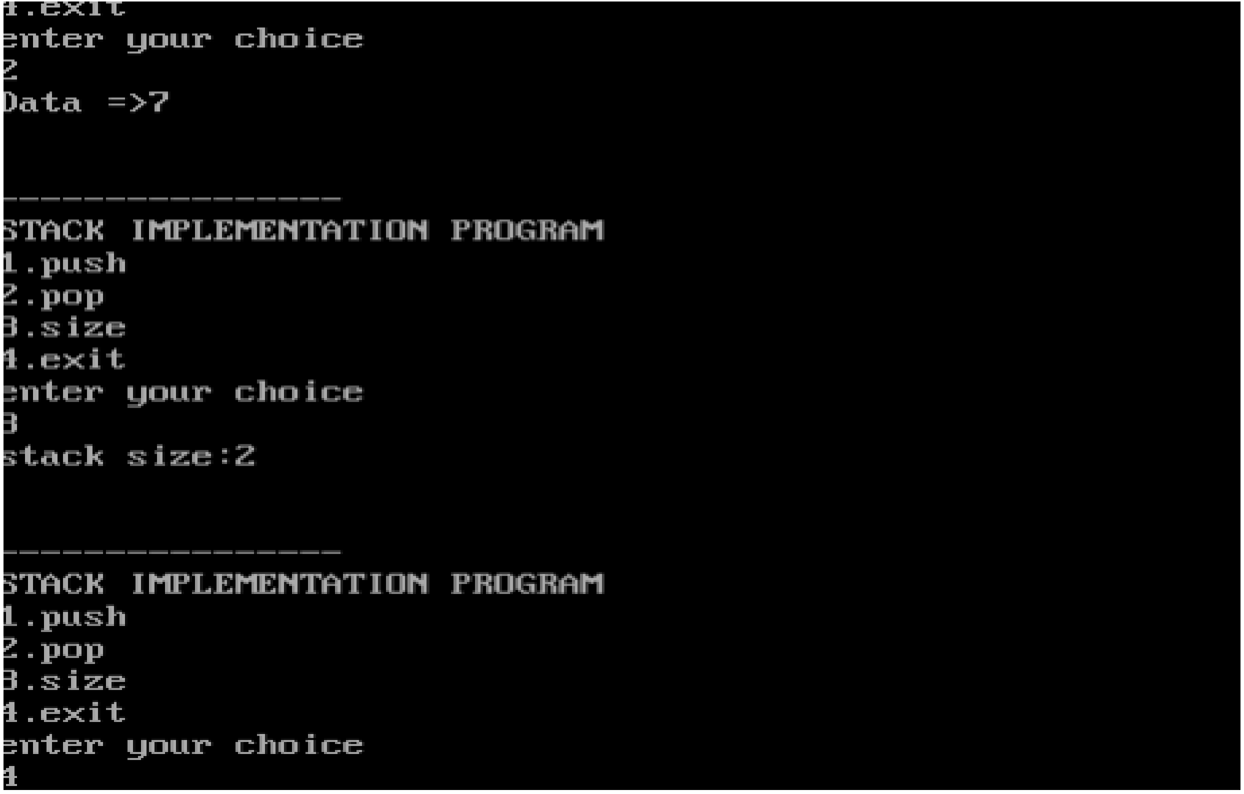
```
----------------
STACK IMPLEMENTATION PROGRAM
1.push
2.pop
3.size
4.exit
enter your choice
1
enter data to push into stack
4
data pushed into stack


----------------
STACK IMPLEMENTATION PROGRAM
1.push
2.pop
3.size
4.exit
enter your choice
1
enter data to push into stack
6
```



```
4.exit
enter your choice
3
stack size:3


----------------
STACK IMPLEMENTATION PROGRAM
1.push
2.pop
3.size
4.exit
enter your choice
2
Data =>7


----------------
STACK IMPLEMENTATION PROGRAM
1.push
2.pop
3.size
4.exit
enter your choice
5
data pushed into stack


----------------
STACK IMPLEMENTATION PROGRAM
1.push
2.pop
3.size
4.exit
enter your choice
```



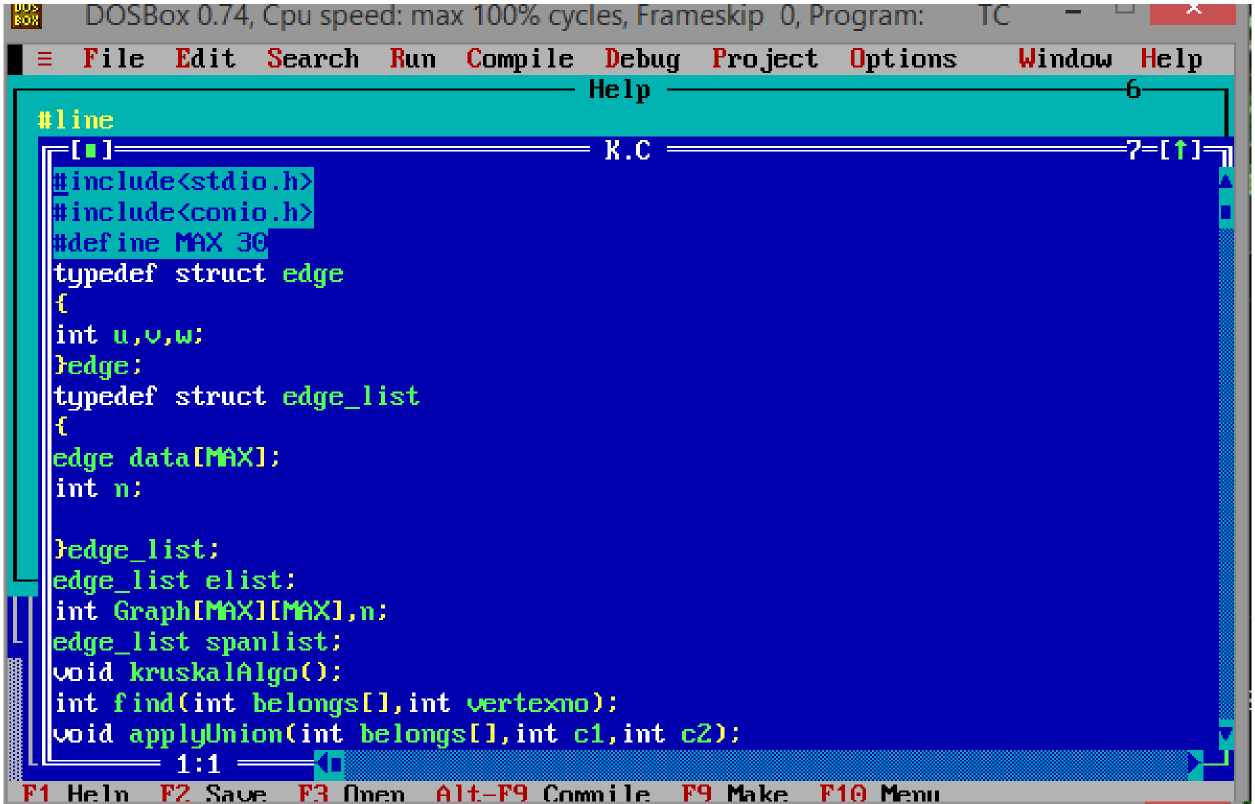
```
----------------
STACK IMPLEMENTATION PROGRAM
1.push
2.pop
3.size
4.exit
enter your choice
5
data pushed into stack


----------------
STACK IMPLEMENTATION PROGRAM
1.push
2.pop
3.size
4.exit
enter your choice
1
enter data to push into stack
7
data pushed into stack


----------------
STACK IMPLEMENTATION PROGRAM
1.push
2.pop
3.size
4.exit
enter your choice
3
```



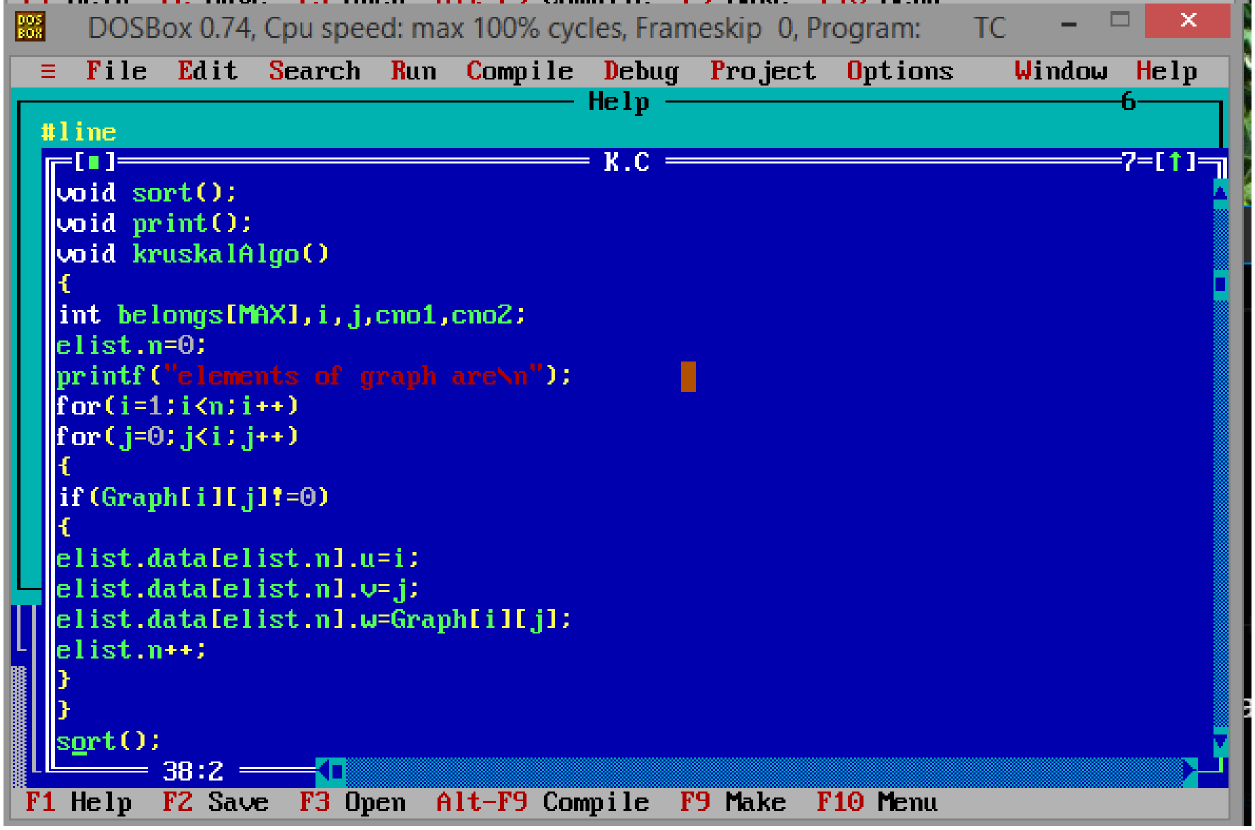
```
4.exit
enter your choice
2
Data =>7


----------------
STACK IMPLEMENTATION PROGRAM
1.push
2.pop
3.size
4.exit
enter your choice
3
stack size:2


----------------
STACK IMPLEMENTATION PROGRAM
1.push
2.pop
3.size
4.exit
enter your choice
4
```

2)Program to implement kruskal”s algorithm

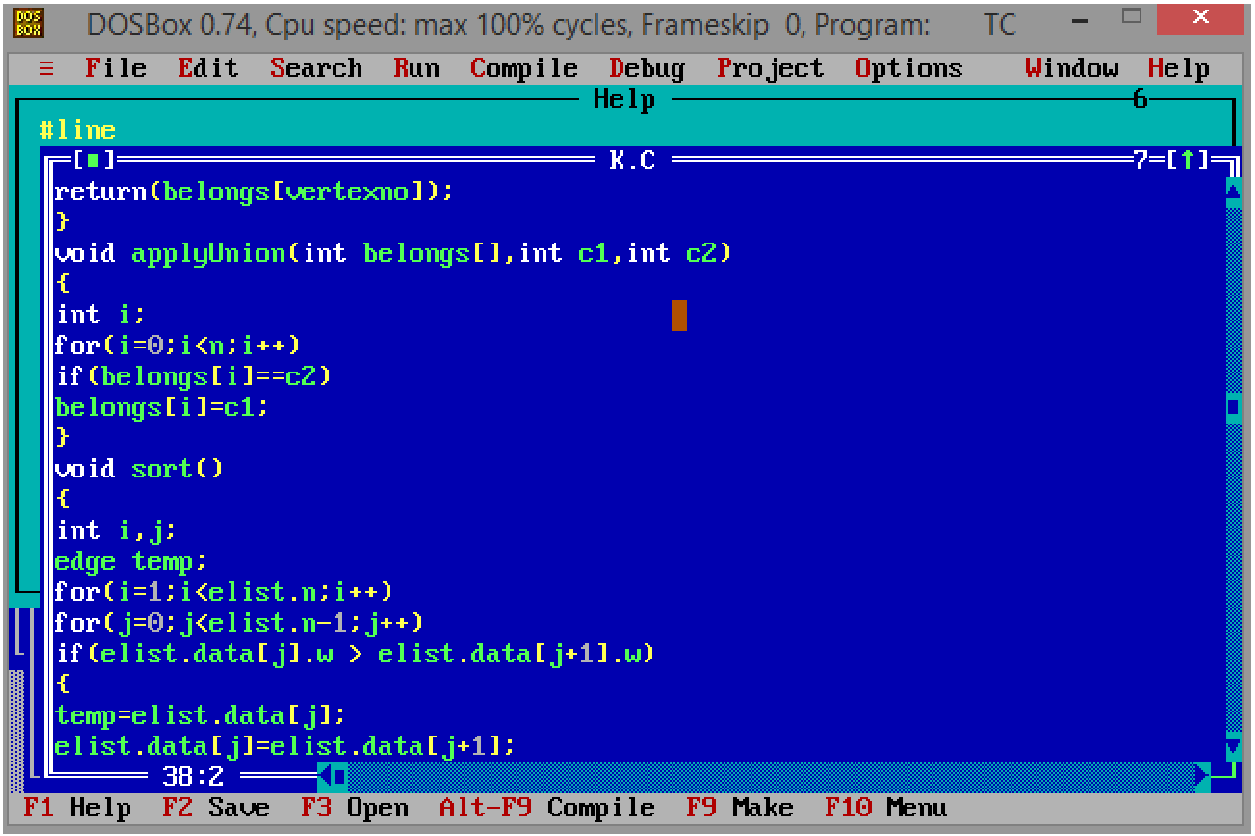
Program code


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
Help
#line
K.C
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

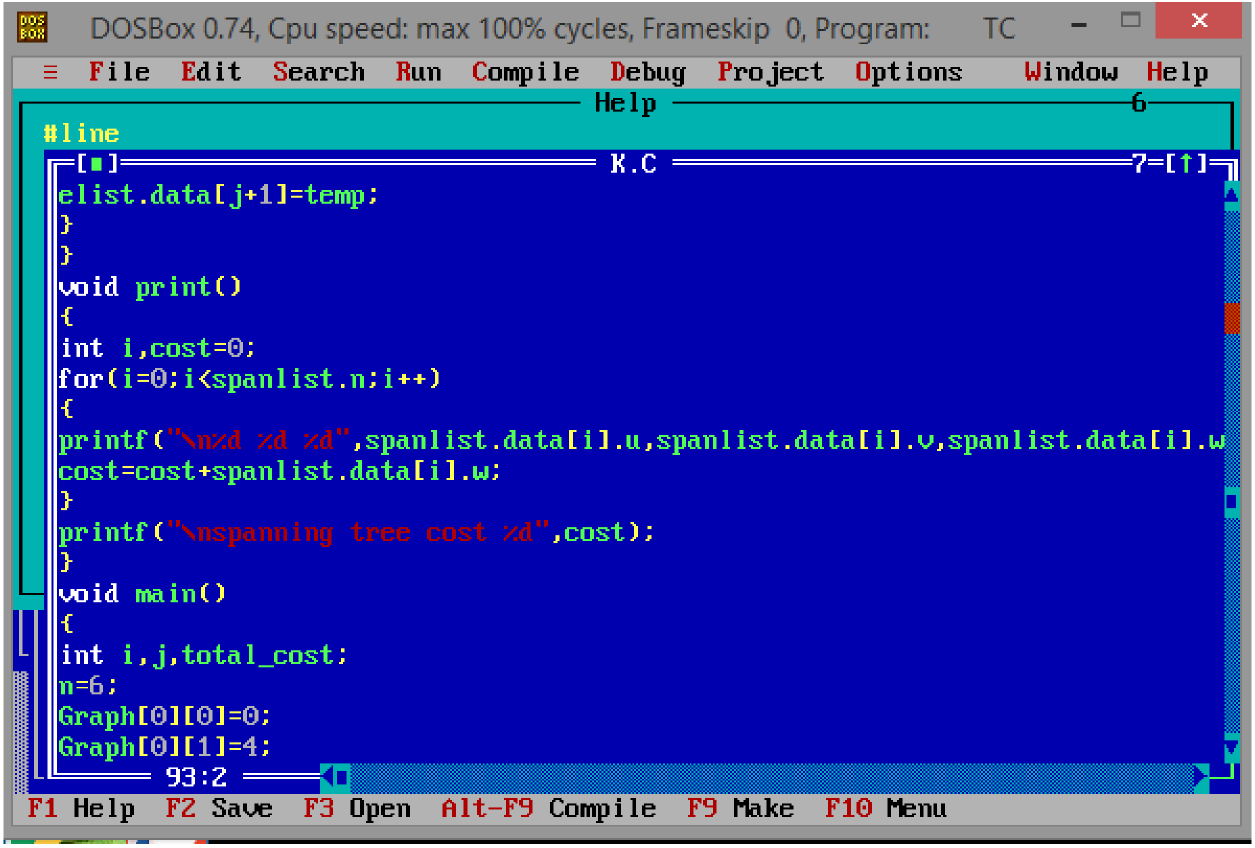
```c
#include<stdio.h>
#include<conio.h>
#define MAX 30
typedef struct edge
{
int u,v,w;
}edge;
typedef struct edge_list
{
edge data[MAX];
int n;

}edge_list;
edge_list elist;
int Graph[MAX][MAX],n;
edge_list spanlist;
void kruskalAlgo();
int find(int belongs[],int vertexno);
void applyUnion(int belongs[],int c1,int c2);
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
Help
#line
K.C
38:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

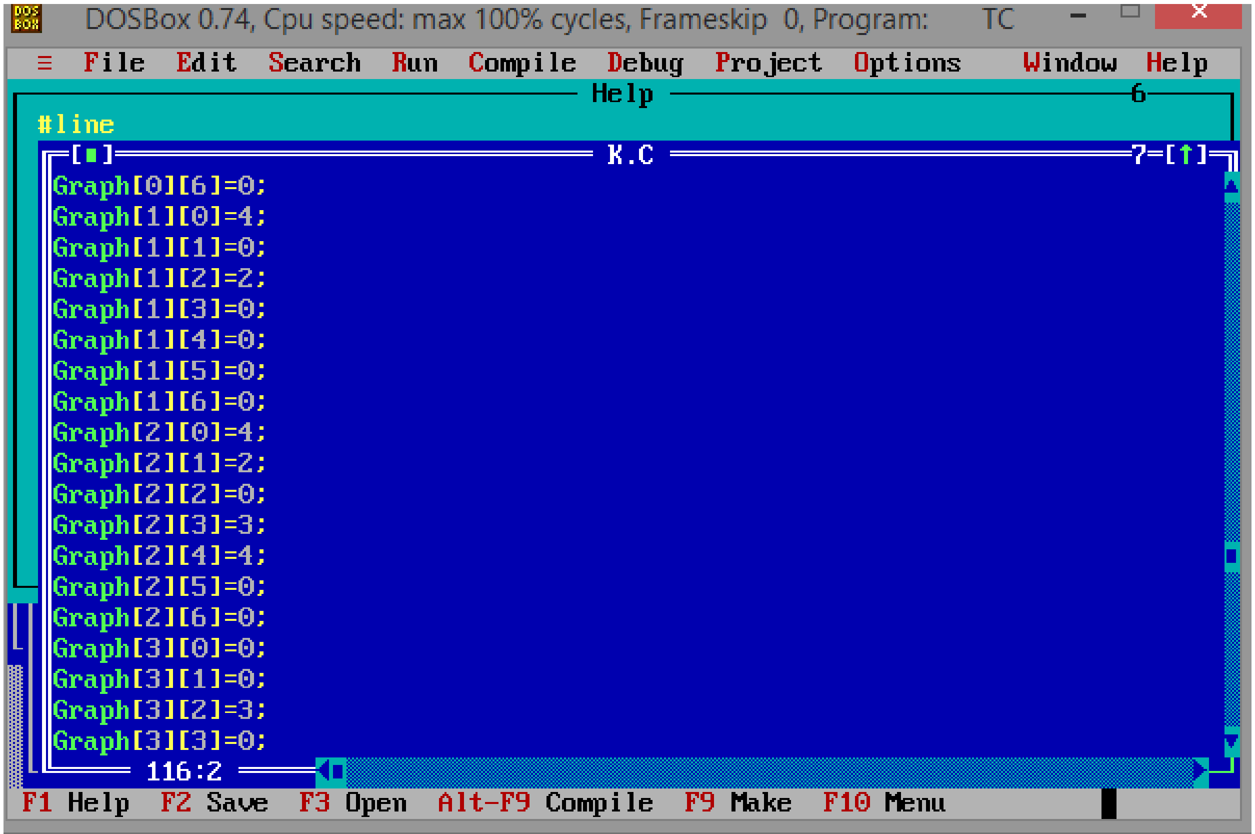
```c
void sort();
void print();
void kruskalAlgo()
{
int belongs[MAX],i,j,cno1,cno2;
elist.n=0;
printf("elements of graph are\n");
for(i=1;i<n;i++)
for(j=0;j<i;j++)
{
if(Graph[i][j]!=0)
{
elist.data[elist.n].u=i;
elist.data[elist.n].v=j;
elist.data[elist.n].w=Graph[i][j];
elist.n++;
}
}
sort();
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
Help
6
#line
[■]
K.C
7=[↑]
38:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

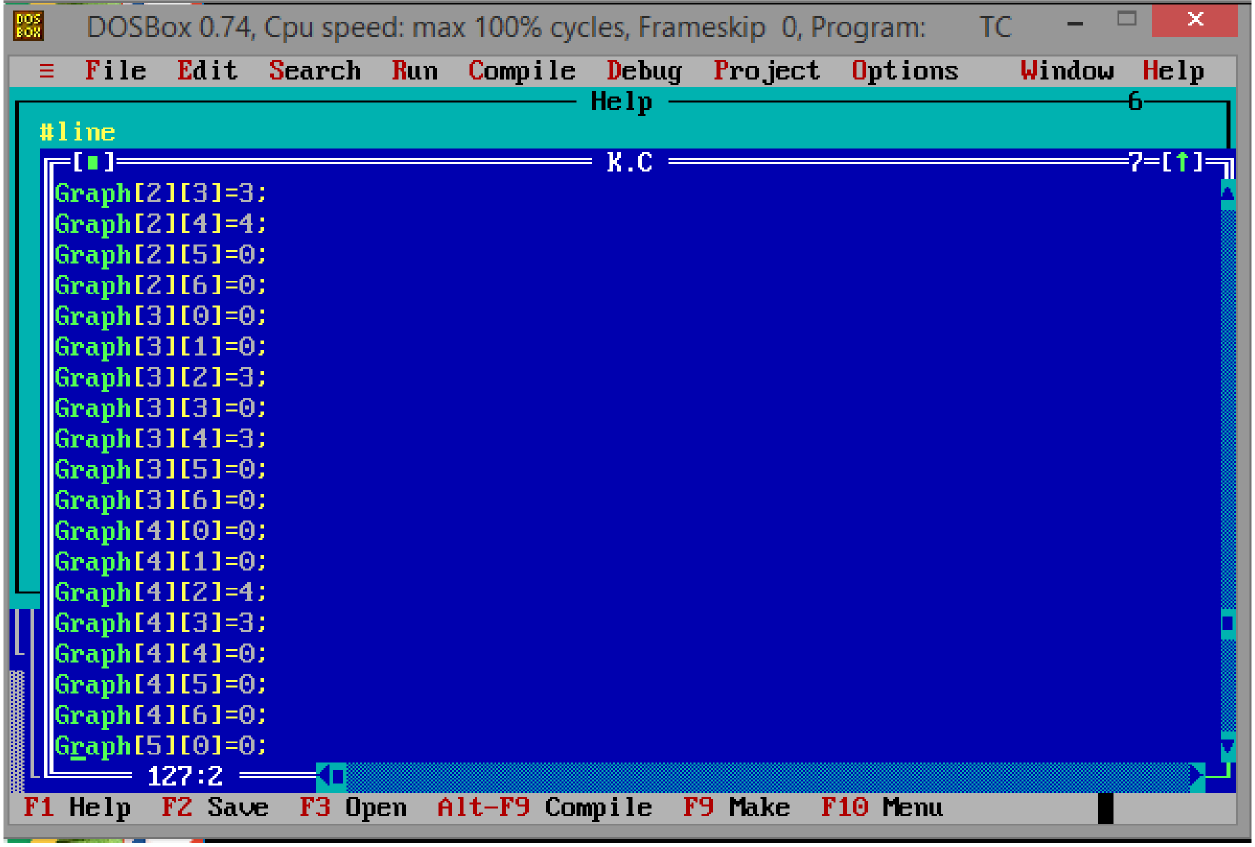
```
return(belongs[vertexno]);
}
void applyUnion(int belongs[],int c1,int c2)
{
int i;
for(i=0;i<n;i++)
if(belongs[i]==c2)
belongs[i]=c1;
}
void sort()
{
int i,j;
edge temp;
for(i=1;i<elist.n;i++)
for(j=0;j<elist.n-1;j++)
if(elist.data[j].w > elist.data[j+1].w)
{
temp=elist.data[j];
elist.data[j]=elist.data[j+1];
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
Help
6
#line
[■]
K.C
7=[↑]
93:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

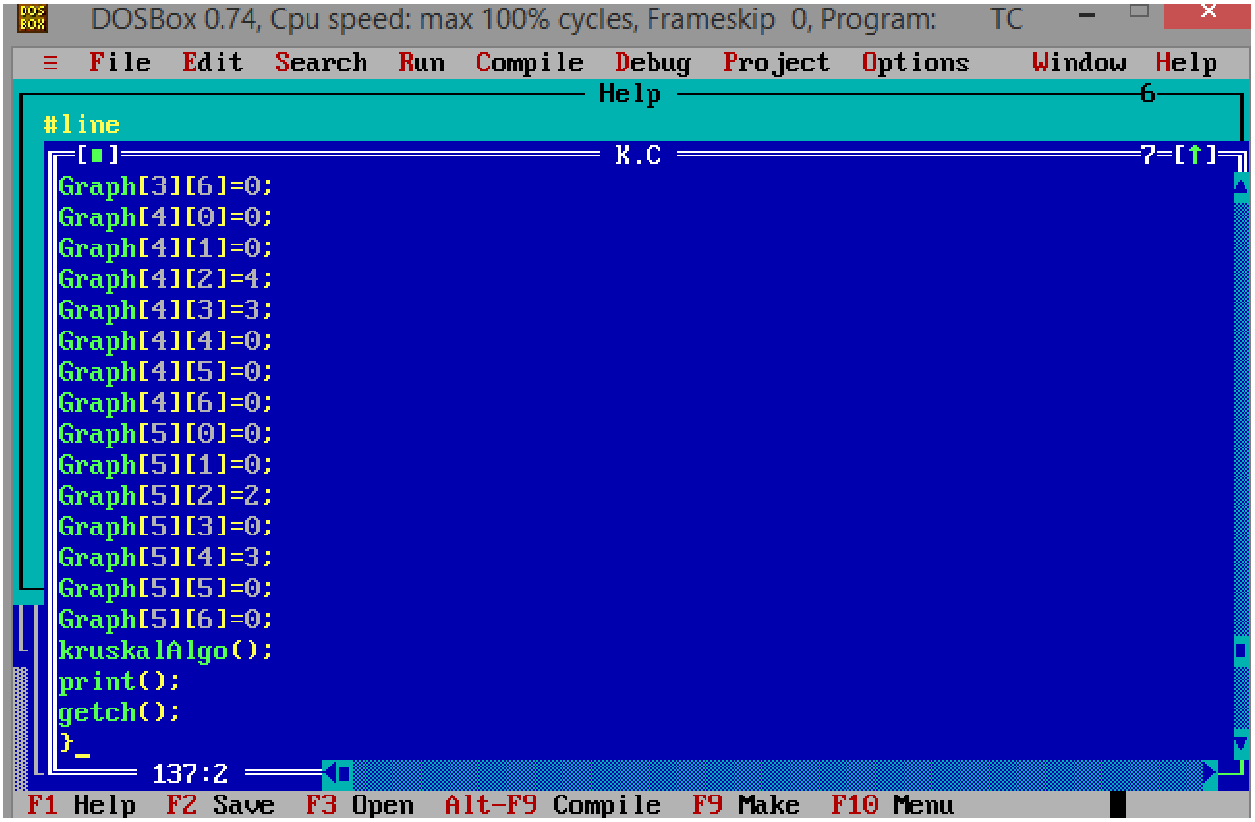
```
elist.data[j+1]=temp;
}
}
void print()
{
int i,cost=0;
for(i=0;i<spanlist.n;i++)
{
printf("\n%d %d %d",spanlist.data[i].u,spanlist.data[i].v,spanlist.data[i].w
cost=cost+spanlist.data[i].w;
}
printf("\nspanning tree cost %d",cost);
}
void main()
{
int i,j,total_cost;
n=6;
Graph[0][0]=0;
Graph[0][1]=4;
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
Help
6
#line
K.C
7
116:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
Graph[0][6]=0;
Graph[1][0]=4;
Graph[1][1]=0;
Graph[1][2]=2;
Graph[1][3]=0;
Graph[1][4]=0;
Graph[1][5]=0;
Graph[1][6]=0;
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;
```

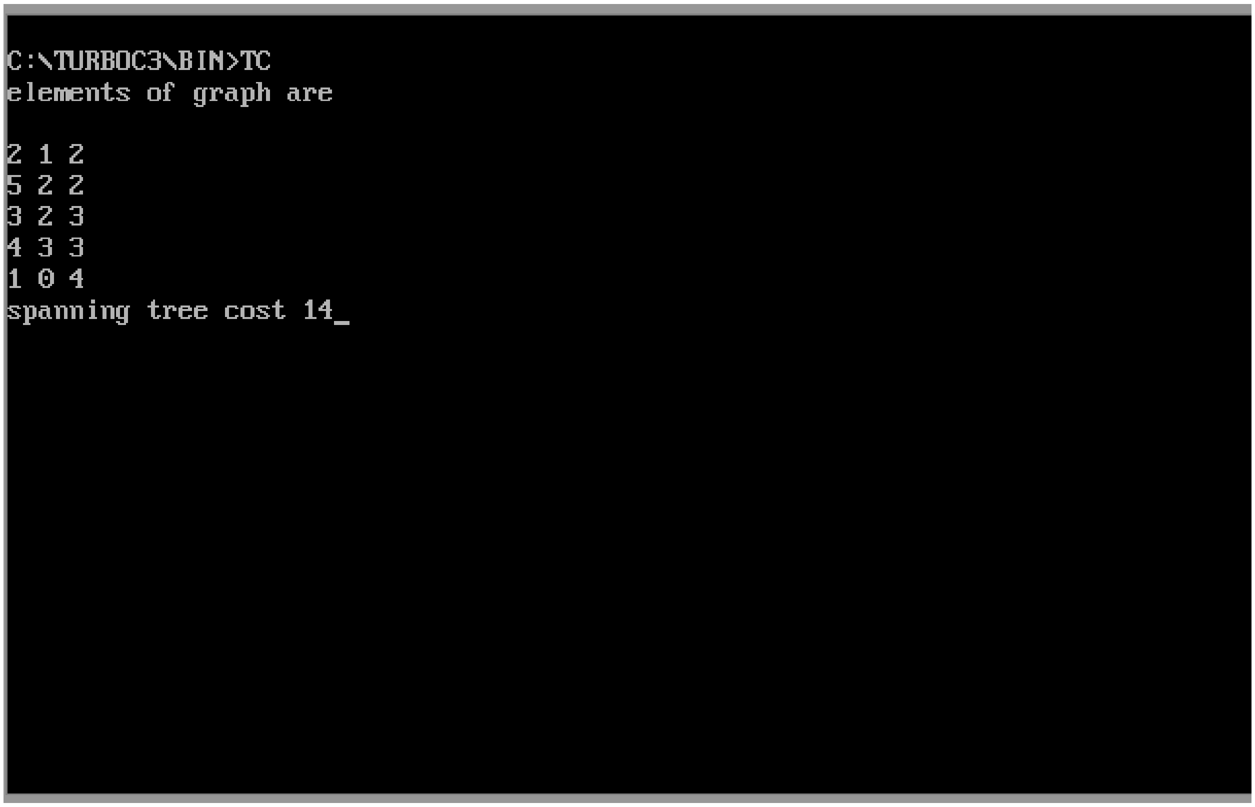

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
Help
6
#line
K.C
7
127:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
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;
Graph[4][4]=0;
Graph[4][5]=0;
Graph[4][6]=0;
Graph[5][0]=0;
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
Help
6
#line
[■]
K.C
7
137:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
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;
Graph[5][4]=3;
Graph[5][5]=0;
Graph[5][6]=0;
kruskalAlgo();
print();
getch();
}
```

Output



```
C:\TURBOC3\BIN>TC
elements of graph are

2 1 2
5 2 2
3 2 3
4 3 3
1 0 4
spanning tree cost 14_
```