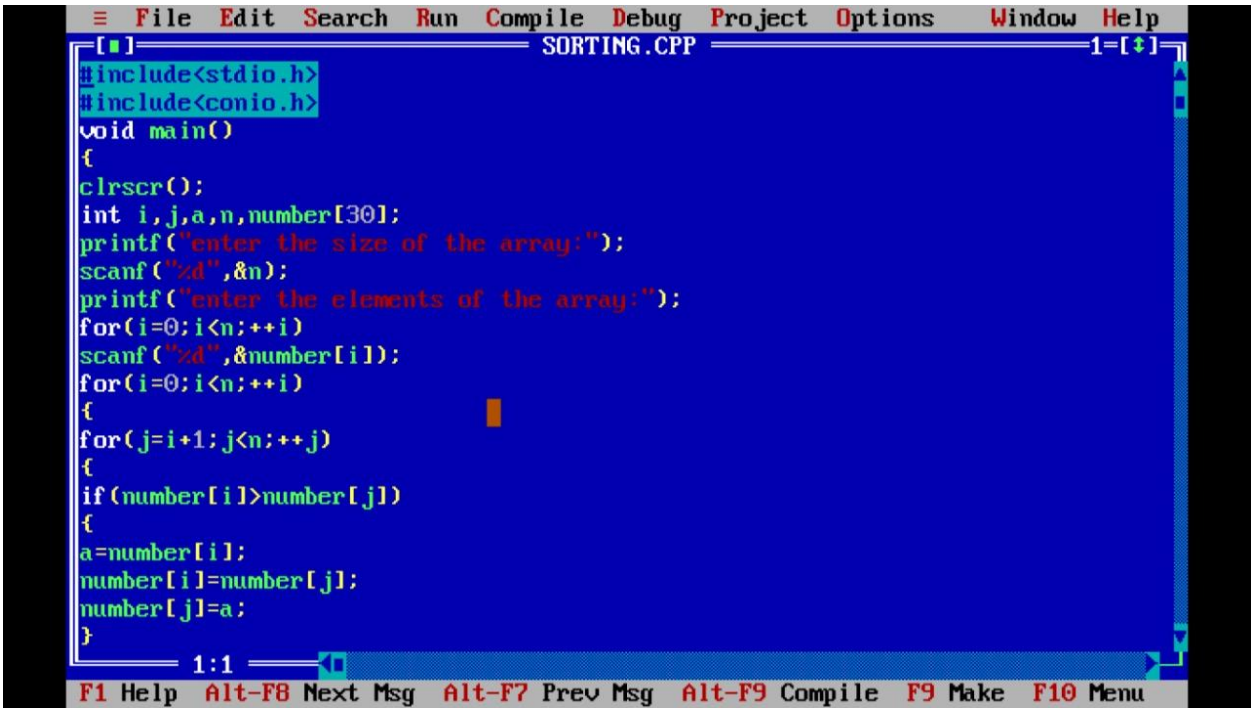
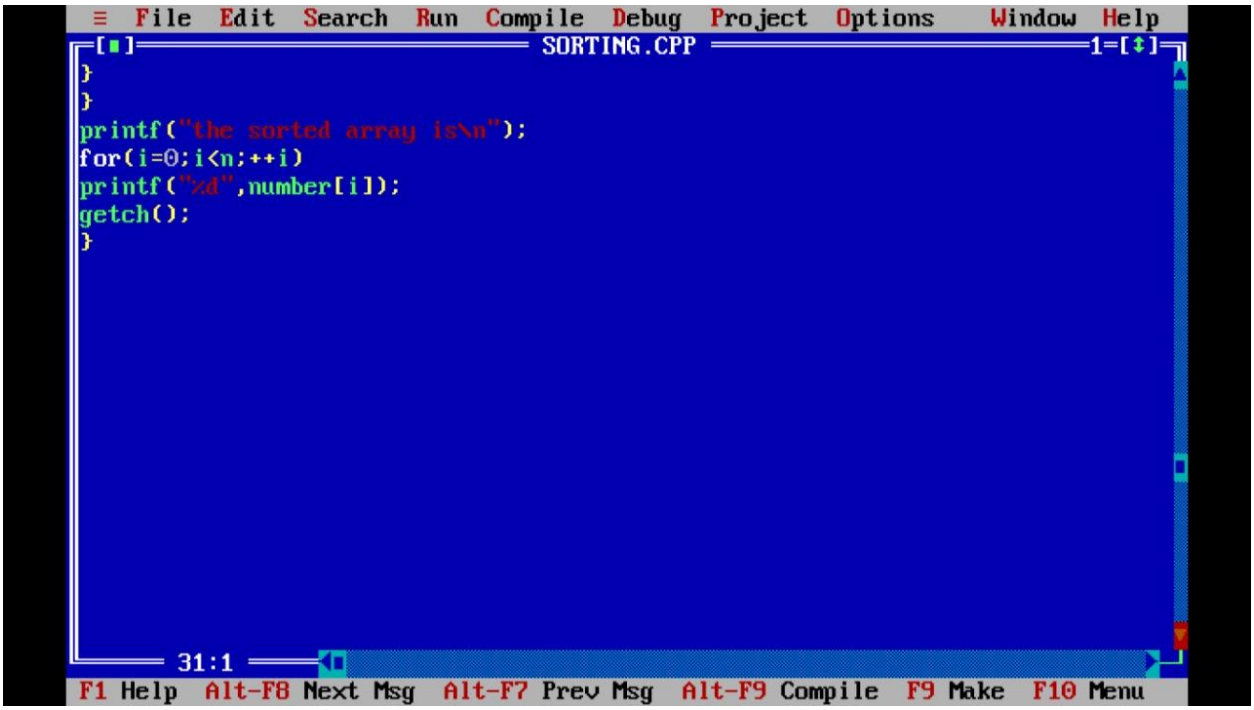
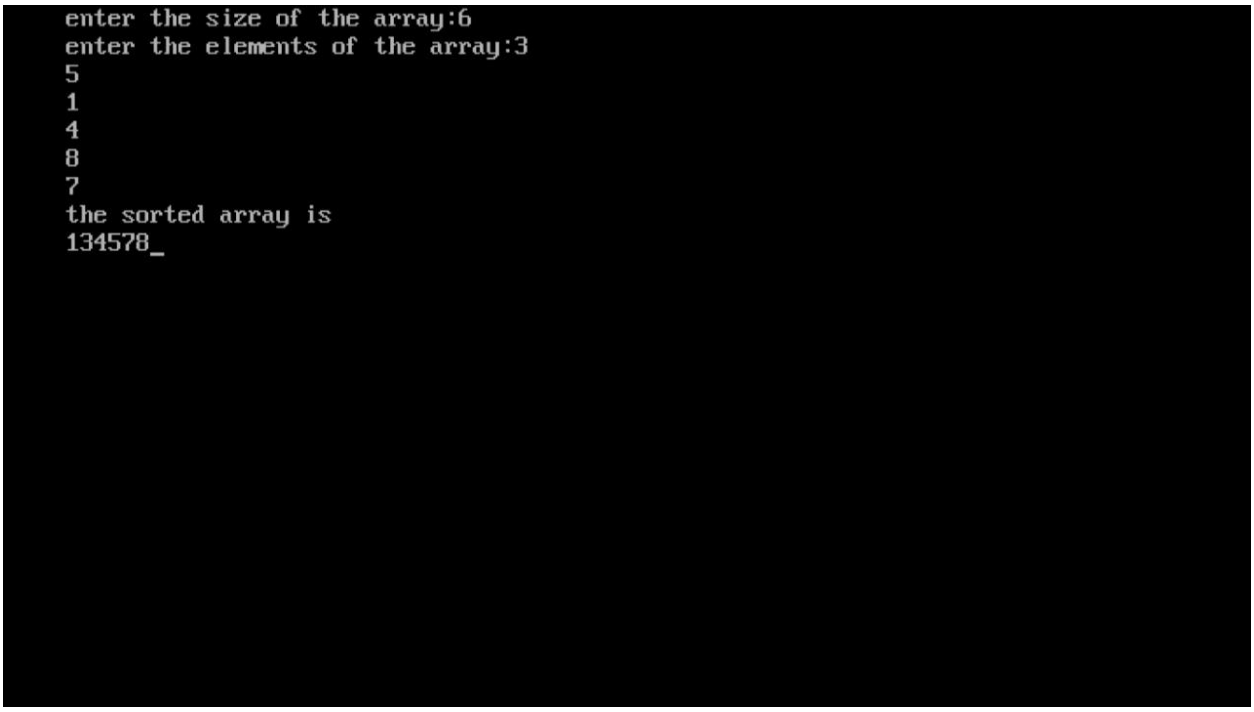
## 1. Sorting of array

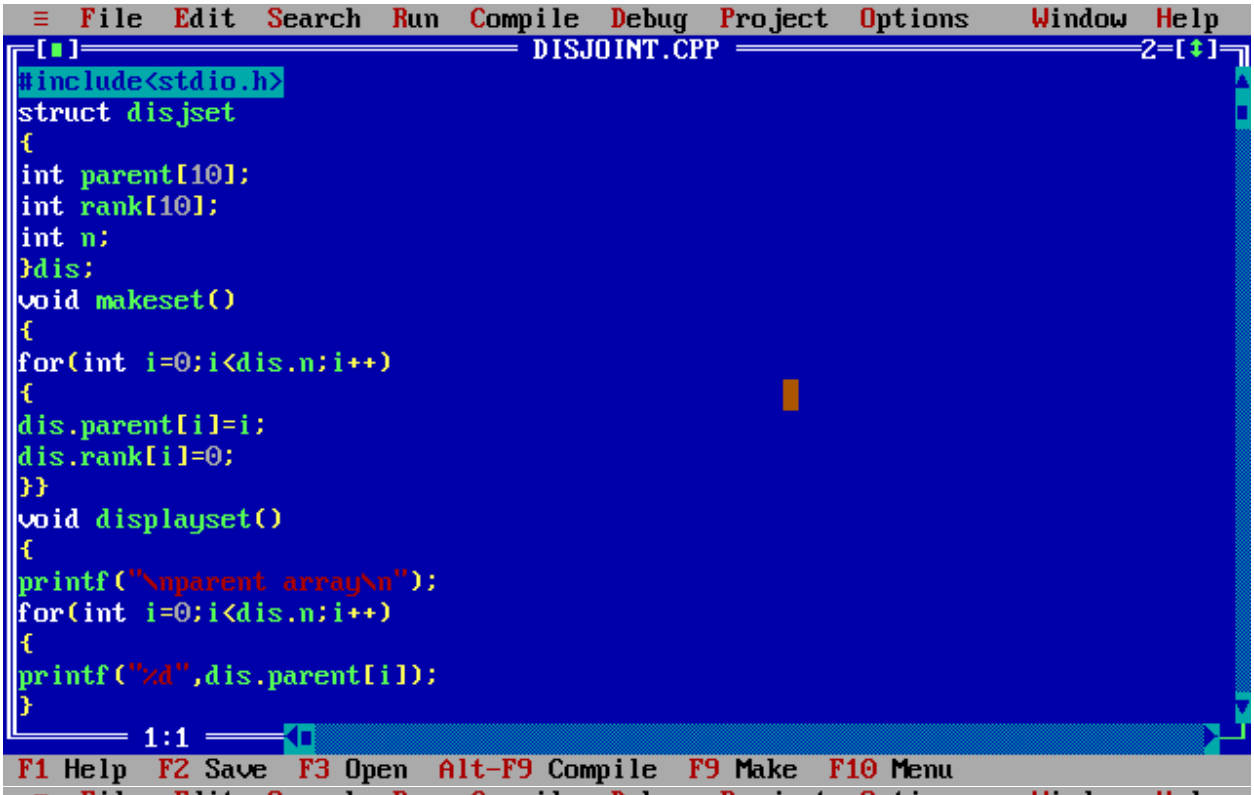


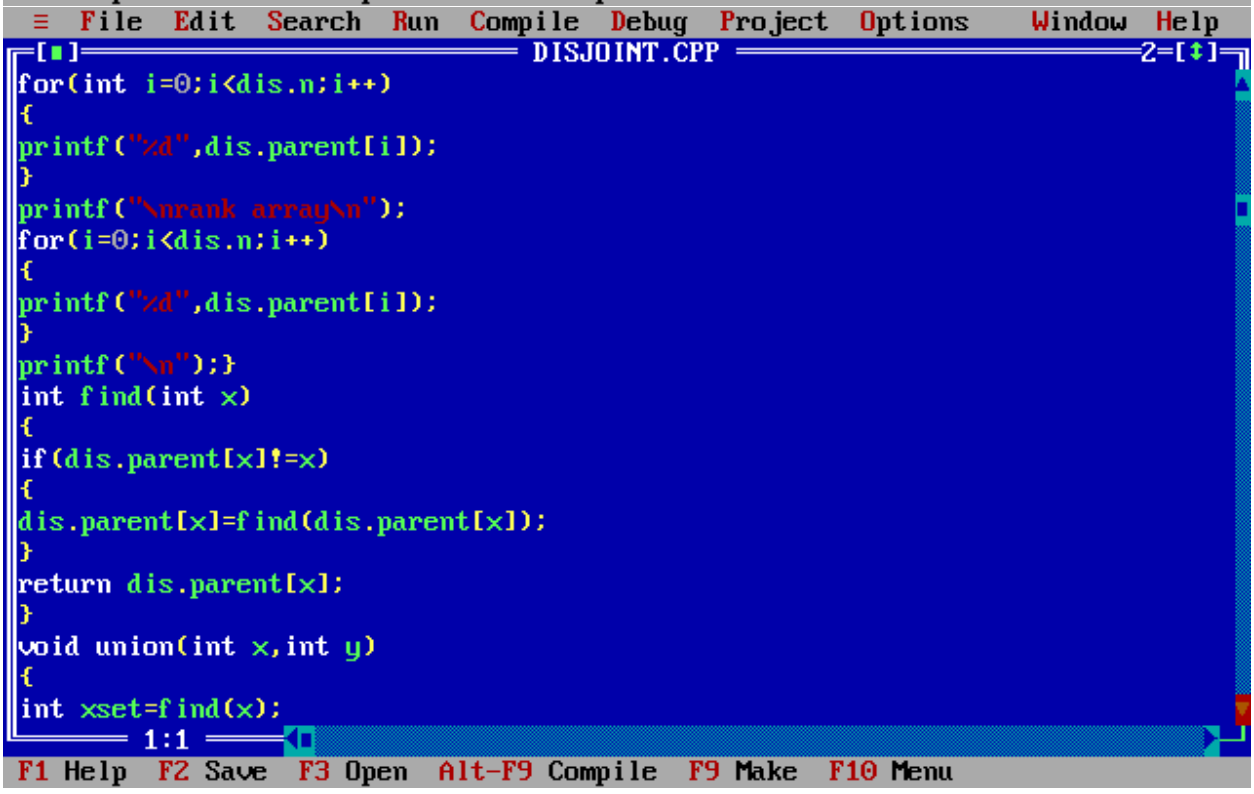



# Output




2.Disjoint set operations


File Edit Search Run Compile Debug Project Options Window Help
DISJOINT.CPP
#include<stdio.h>
struct disjset
{
int parent[10];
int rank[10];
int n;
}dis;
void makeset()
{
for(int i=0;i<dis.n;i++)
{
dis.parent[i]=i;
dis.rank[i]=0;
}}
void displayset()
{
printf("\nparent array\n");
for(int i=0;i<dis.n;i++)
{
printf("%d",dis.parent[i]);
}
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

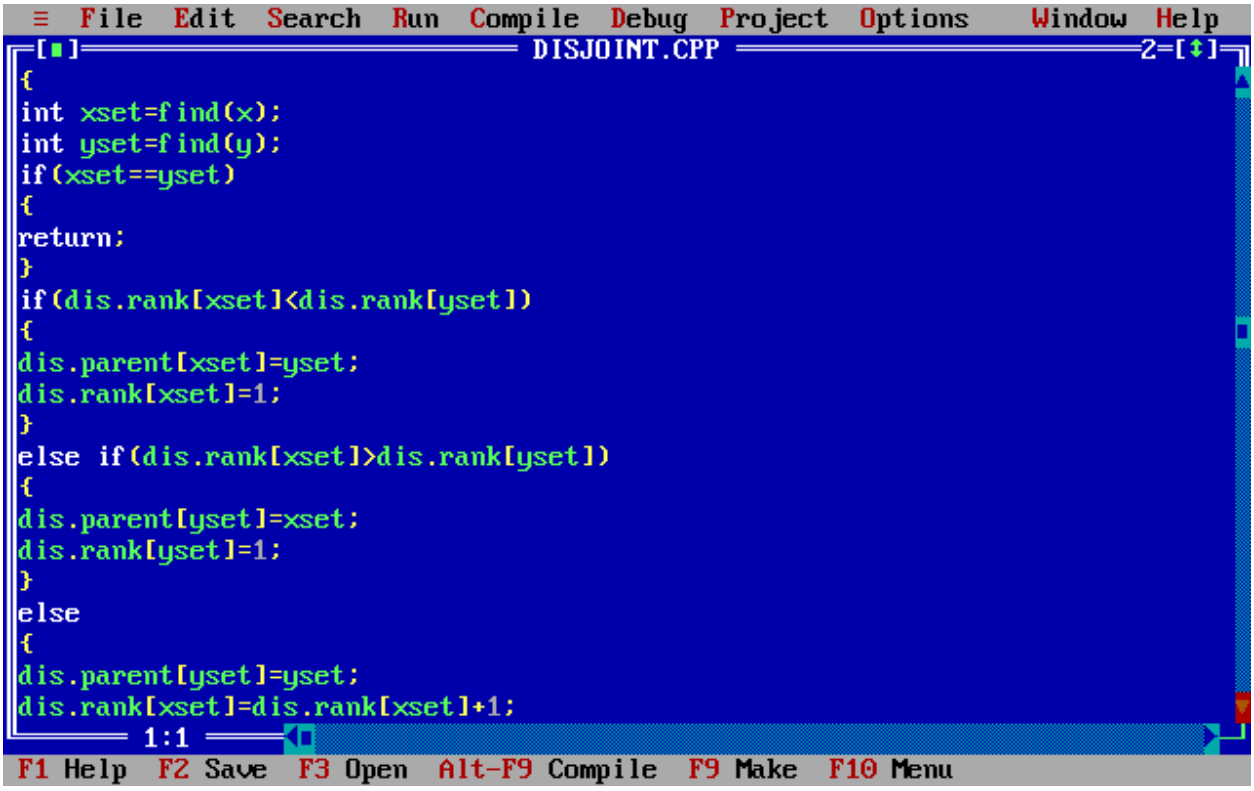

File Edit Search Run Compile Debug Project Options Window Help
DISJOINT.CPP
for(int i=0;i<dis.n;i++)
{
printf("%d",dis.parent[i]);
}
printf("\nrank array\n");
for(i=0;i<dis.n;i++)
{
printf("%d",dis.parent[i]);
}
printf("\n");}
int find(int x)
{
if(dis.parent[x]!=x)
{
dis.parent[x]=find(dis.parent[x]);
}
return dis.parent[x];
}
void union(int x,int y)
{
int xset=find(x);
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
[■]═══════════════ DISJOINT.CPP ═══════════════2=[↕]
{
int xset=find(x);
int yset=find(y);
if(xset==yset)
{
return;
}
if(dis.rank[xset]<dis.rank[yset])
{
dis.parent[xset]=yset;
dis.rank[xset]=1;
}
else if(dis.rank[xset]>dis.rank[yset])
{
dis.parent[yset]=xset;
dis.rank[yset]=1;
}
else
{
dis.parent[yset]=yset;
dis.rank[xset]=dis.rank[xset]+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
[■]═══════════════ DISJOINT.CPP ═══════════════2=[↕]
dis.rank[yset]=1;
}
}
int main()
{
int x,y,n;
printf("How many elemments");
scanf("%d",&n);
makeset();
int ch,wish;
do
{
printf("\n MENU\n1.UNION\n2.FIND\n3.DISPLAY\n");
printf("eter choice");
scanf("%d",&ch);
switch(ch)
{
cas 1:printf("enter elements to prform union");
scanf("%d%d",&x,&y);
union(x,y);
break;
═══════ 1:1 ═══
F1 Help  F2 Save  F3 Open  Alt-F9 Compile  F9 Make  F10 Menu
```

```
switch(ch)
{
cas 1:printf("enter elements to prform union");
scanf("%d%d",&x,&y);
union(x,y);
break;
case 2:printf("enter elements to check if the components ");
scanf("%d%d",&x,&y);
if(find(x)==find(y))
printf("connecet components\n");
else
printf("not connected components");
break;
case 3:
displayset();
break;
}
printf("Do you wish to continue?(1/0)\n");
scanf("%d",&wish);
}
while(wish==1);
```

```
if(find(x)==find(y))
printf("connecet components\n");
else
printf("not connected components");
break;
case 3:
displayset();
break;
}
printf("Do you wish to continue?(1/0)\n");
scanf("%d",&wish);
}
while(wish==1);
return 0;
}
```


File Edit Search Run Compile Debug Project Options Window Help
DISJOINT.CPP
{
int xset=find(x);
int yset=find(y);
if(xset==yset)
{
return;
}
if(dis.rank[xset]<dis.rank[yset])
{
dis.parent[xset]=yset;
dis.rank[xset]=1;
}
else if(dis.rank[xset]>dis.rank[yset])
{
dis.parent[yset]=xset;
dis.rank[yset]=1;
}
else
{
dis.parent[yset]=yset;
dis.rank[xset]=dis.rank[xset]+1;
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

+

```
___Menu___
1.Union
2.Find
3.Display
enter choice
2
Enter elements to check if connected components:1 4
Not connected components
Do you wish to continue?(1/0)
1

___Menu___
1.Union
2.Find
3.Display
enter choice
3

Parent Array
0122
Rank Array
001-1

Do you wish to continue?(1/0)
```

```
How many elements?4

___Menu___
1.Union
2.Find
3.Display
enter choice
1
Enter elements to perform union:2 3

Do you wish to continue?(1/0)
1

___Menu___
1.Union
2.Find
3.Display
enter choice
2
Enter elements to check if connected components:
```

+