**DISJOINT SET OPERATIONS**

#include<stdio.h>

struct disjointset

{

int parent[10];

int rank[10];

int n;

} dis;

void makeset()

{

int i;

for(i=0;i<dis.n;i++)

{

dis.parent[i]=i;

dis.rank[i]=0;

}

}

void displyset()

{

int i;

printf("Parent Array:\n");

for(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.rank[i]);

}

}

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 xset1=find(x);

int yset2=find(y);

if(xset1==yset2)

return;

if(dis.rank[xset1]<dis.rank[yset2])

{

dis.parent[xset1]=yset2;

dis.rank[xset1]= -1;

}

else if(dis.rank[xset1]>dis.rank[yset2])

{

dis.parent[yset2]=xset1;

dis.rank[xset1]= -1;

}

else

{

dis.parent[yset2]=xset1;

dis.rank[xset1]=dis.rank[xset1]+1;

dis.rank[yset2]=-1;

}

}

int main()

{

int x,y,n;

printf("Number of elements :\n");

scanf("%d",&dis.n);

makeset();

int ch;

do

{

printf("\n---menu---\n");

printf("\n1.union\n2.find\n3.display\n4.Exit\n");

printf("enter choice:\n");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("enter elements to perform union:");

scanf("%d %d",&x,&y);

Union(x,y);

break;

case 2: printf("enter elements to check if it is connected:");

scanf("%d %d",&x,&y);

if(find(x)==find(y))

printf("connected component");

else

printf("Not Connected component");

break;

case 3: displyset();

break;

}

}while(ch!=4);

return 0;

}

**OUTPUT**







