LAB CYCLE 2

SUBMITTED TO :

FOUSIA MISS

SUBMITTED BY:

NANDANA ANIL

ROLL NO : MCA107

S1 MCA

1.BIT STRING

```
#include<stdio.h>

void Union(int A1[],int B1[]);
void Intersection(int A1[], int B1[] );
void BitString(int s[]);
void Difference(int A1[], int B1[]);
int A1[5],B1[5],S[5] ,D[5],k=0;
int U[5]={1,2,3,4,5};

int main()
{

  int i=0,j=0;
  int A[]={2,4,5},B[]={1,3,5};

  printf("U=");
  for(i=0; i<5; i++)
  {
    printf("%d\t",U[i]);
  }
  printf("\n");

  printf("A=");
  for(i=0; i<3; i++)
  {
    printf("%d\t",A[i]);
  }
  printf("\n");

  printf("B=");
  for(i=0; i<3; i++)
  {
    printf("%d\t",B[i]);
  }
  printf("\n");

  BitString(A);
   printf("Bitstring of A=");
  for(i=0; i<5; i++)
  {
    printf("%d\t",S[i]);
    A1[i]=S[i];
  }
  printf("\n");
```

```
 BitString(B);
 printf("Bitstring of B=");
 for(i=0; i<5; i++)
 {
  printf("%d\t",S[i]);
  B1[i]=S[i];
 }
 printf("\n");

 Union(A1,B1);
 Intersection(A1,B1);

 printf("A difference B=");
 Difference(A1,B1);

 printf("A difference B=");
 for(i=0; i<5;i++)
  {
    if(D[i]==1)
    {
      printf("%d\t",U[i]);
     }
 }
 printf("\n");


 printf("B difference A=");
 Difference(B1,A1);

 printf("B difference A=");
 for(i=0; i<5;i++)
  {
    if(D[i]==1)
    {
      printf("%d\t",U[i]);
     }
 }
 printf("\n");


}

void BitString(int S1[])
{
   int j=0,i=0;
```

```
    while(i<5)
     {
       if(U[i]!=S1[j])
       {
        S[i]=0;
        i++;

       }

       else
       {
          S[i]=1;
          i++;
          if(j<3)

           j++;
       }
     }

 }

void Union(int A1[], int B1[])
{
  int i,C[5];

  printf("A union B=\t");
  for(i=0; i<5;i++)
  {
    if(A1[i]==B1[i]==1)
      {
       C[i]=1;
       printf("%d\t",C[i]);
      }

    else
     {
      C[i]=A1[i]+B1[i];
      printf("%d\t",C[i]);
     }
  }
  printf("\n");

  printf("A union B=\t ");
  for(i=0; i<5;i++)
  {
```

```
    if(C[i]==1)
    {
      printf("%d\t",U[i]);
    }
  }
  printf("\n");


}

void Intersection(int A1[], int B1[])
{
  int i,k=0,C[5];

  printf("A intersection B=");
  for(i=0; i<5;i++)
  {
    C[i]=A1[i]*B1[i];
    printf("%d\t",C[i]);
  }
  printf("\n");

  printf("A intersection B=");
  for(i=0; i<5;i++)
  {
    if(C[i]==1)
    {
      printf("%d\t",U[i]);



    }
  }
  printf("\n");

}

void Difference(int A1[],int B1[])
{
  int i=0;

  for(i=0;i<5;i++)
  {
    if(A1[i]==B1[i]==1)
    {
      D[i]=0;
```

```
    }
    else
    {
      D[i]=A1[i];
    }
  }

 for(i=0; i<5; i++)
  {
   printf("%d\t",D[i]);
  }
 printf("\n");


}
```

2.DISJOINT SET

```
#include<stdio.h>
#include<stdlib.h>
struct node{
struct node *rep;
struct node *next;
int data;
}*heads[50],*tails[50];
static int countRoot=0;
void makeSet(int x){
struct node *new=(struct node *)malloc(sizeof(struct node));
new->rep=new;
new->next=NULL;
new->data=x;
heads[countRoot]=new;
tails[countRoot++]=new;
}
struct node* find(int a){
int i;
struct node *tmp=(struct node *)malloc(sizeof(struct node));
for(i=0;i<countRoot;i++){
tmp=heads[i];
while(tmp!=NULL){
if(tmp->data==a)
return tmp->rep;
tmp=tmp->next;
}
```

```
}
return NULL;
}
void unionSets(int a,int b){
int i,pos,flag=0,j;
struct node *tail2=(struct node *)malloc(sizeof(struct node));
struct node *rep1=find(a);
struct node *rep2=find(b);
if(rep1==NULL||rep2==NULL){
printf("Element not present in the DS");
return;
}
if(rep1!=rep2){
for(j=0;j<countRoot;j++){
if(heads[j]==rep2){
pos=j;
flag=1;
countRoot-=1;
tail2=tails[j];
for(i=pos;i<countRoot;i++){
heads[i]=heads[i+1];
tails[i]=tails[i+1];
}}
if(flag==1)
break;
}
for(j=0;j<countRoot;j++){
if(heads[j]==rep1){
```

```
tails[j]->next=rep2;
tails[j]=tail2;
break;
}}
while(rep2!=NULL){
rep2->rep=rep1;
rep2=rep2->next;
}}}
int search(int x){
int i;
struct node *tmp=(struct node *)malloc(sizeof(struct node));
for(i=0;i<countRoot;i++){
tmp=heads[i];
if(heads[i]->data==x)
return 1;
while(tmp!=NULL){
if(tmp->data==x)
return 1;
tmp=tmp->next;
}}
return 0;
}
void main(){
int choice,x,i,j,y,flag=0;
do{
printf("\n.....MENU.....1.Make Set.....2.Display set representatives.....3.Union.....4.Find Set....5.Exit....");
printf("\nEnter your choice : ");
scanf("%d",&choice);
```

```
switch(choice){
case 1:
printf("Enter new element : ");
scanf("%d",&x);
if(search(x)==1)
printf("Element already present in the disjoint set DS");
else
makeSet(x);
break;
case 2:
for(i=0;i<countRoot;i++)
printf("%d ",heads[i]->data);
break;
case 3:
printf("Enter first element : ");
scanf("%d",&x);
printf("Enter second element : ");
scanf("%d",&y);
unionSets(x,y);
break;
case 4:
printf("Enter the element");
scanf("%d",&x);
struct node *rep=(struct node *)malloc(sizeof(struct node));
rep=find(x);
if(rep==NULL)
printf("\nElement not present in the DS");
else
```

```
printf("\nThe representative of %d is %d",x,rep->data);
break;
case 5:
exit(0);
default:
printf("\nWrong choice");
break;
}}
while(1);
};
```

LINK TO GITHUB REPOSITORY:

https://github.com/NandanaAnil/Data-Structures.git