**LAB CYCLE 2**

**Submitted by**

**SIJI JOSE**

**MCA 135**

**PROGRAM 1 : BIT STRING**

#define MAX 30

#include<stdio.h>

#include<stdlib.h>

void create(int set[]);

void print(int set[]);

void Union(int set1[],int set2[],int set3[]);

void intersection(int set1[],int set2[],int set3[]);

void difference(int set1[],int set2[],int set3[]);

void symmdiff(int set1[],int set2[],int set3[]);

int member(int set[],int x);

void main()

{ int set1[MAX],set2[MAX],set3[MAX];

int x,op;

set1[0]=set2[0]=set3[0]=0;

do{

printf("\n1)Create 2)Print 3)Union 4)Intersection 5)Difference 6)Symmetric Difference 7)Quit");

printf("\nEnter Your Choice:");

scanf("%d",&op);

switch(op){

case 1: printf("---Creating First Set---");

create(set1);

printf("---Creating Second Set---");

create(set2);

break;

case 2: printf("First Set : ");

print(set1);

printf("Second Set : ");

print(set2);

break;

case 3: Union(set1,set2,set3);print(set3);break;

case 4: intersection(set1,set2,set3);print(set3);break;

case 5: difference(set1,set2,set3);print(set3);break;

case 6: symmdiff(set1,set2,set3);print(set3);break;}

printf("\nPRESS ANY KEY");

}while(op!=7);}

void create(int set[])

{ int n,i,x;

set[0]=0;/\*make it a null set\*/

printf("\n No. of elements in the set:");

scanf("%d",&n);

printf("enter set elements :");

for(i=1;i<=n;i++)

scanf("%d",&set[i]);

set[0]=n; }

void print(int set[])

{ int i,n;

n=set[0];/\* number of elements in the set \*/

printf("\n Members of the set :-->");

for(i=1;i<=n;i++)

printf("%d ",set[i]);

}

/\* union of set1[] and set2[] is stored in set3[]\*/

void Union(int set1[],int set2[],int set3[])

{ int i,n;

/\* copy set1[] to set3[]\*/

set3[0]=0;/\*make set3[] a null set \*/

n=set1[0];/\* number of elements in the set\*/

//Union of set1,set2= set1 + (set2-set1)

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

set3[i]=set1[i];

n=set2[0];

for(i=1;i<=n;i++)

if(!member(set3,set2[i]))

set3[++set3[0]]=set2[i]; // insert and increment no. of elements

}

/\*function returns 1 or 0 depending on whether x belongs

to set[] or not \*/

int member(int set[],int x)

{ int i,n;

n=set[0]; /\* number of elements in the set\*/

for(i=1;i<=n;i++)

if(x==set[i])

return(1);

return(0);

}

void intersection(int set1[],int set2[],int set3[])

{

int i,n;

set3[0]=0; /\* make a NULL set\*/

n=set1[0];/\* number of elements in the set\*/

for(i=1;i<=n;i++)

if(member(set2,set1[i])) /\* all common elements are inserted in set3[]\*/

set3[++set3[0]]=set1[i]; // insert and increment no. of elements

}

/\*difference of set1[] and set2[] is stored in set3[]\*/

void difference(int set1[],int set2[],int set3[])

{ int i,n;

n=set1[0];/\* number of elements in the set\*/

set3[0]=0;/\*make it a null set\*/

for(i=1;i<=n;i++)

if(!member(set2,set1[i]))

set3[++set3[0]]=set1[i]; // insert and increment no. of elements

}

void symmdiff(int set1[],int set2[],int set3[])

{ int i,n;

n=set1[0];/\* number of elements in the set\*/

set3[0]=0;/\*make it a null set\*/

//Calculate set1-set2

for(i=1;i<=n;i++)

if(!member(set2,set1[i]))

set3[++set3[0]]=set1[i]; // insert and increment no. of elements

//Calculate set2-set1

n=set2[0];

for(i=1;i<=n;i++)

if(!member(set1,set2[i]))

set3[++set3[0]]=set2[i]; // insert and increment no. of elements

}

**PROGRAM 2 : DISJOINT SETS**

#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) };