/\*Write a program that takes two or more sets as input and produces set operations like union,

intersection, difference and symmetric difference as its output.\*/

#include<stdio.h>

#include<stdlib.h>

int set\_union(int setA[], int m, int setB[], int n, int UNION[])

{

int i,j,k=0;

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

{

UNION[k]=setA[i];

k++;

}

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

{

int flag=1;

for(j=0;j<m;j++)

{

if(setA[j]== setB[i])

{

flag = 0;

break;

}

}

if(flag ==1)

{

UNION[k] = setB[i];

k++;

}

}

return (k);

}

int Intersection(int setA[],int m,int setB[],int n,int INTER[])

{

int i,j,k=0,l;

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

{

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

{

if(setA[i]==setB[j])

{

INTER[k]=setA[i];

k++;

}

}

}

return k;

}

int Difference(int setA[],int m, int setB[],int n,int DIFF[])

{

int i,j,k=0;

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

{

int flag=0;

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

{

if(setA[i]==setB[j])

{

flag =1;

break;

}

}

if (flag == 0)

{

DIFF[k++]=setA[i];

}

}

return k;

}

int Symmetric(int setA[],int m,int setB[],int n,int SYMM[])

{

int k = set\_union(setA,m,setB,n,SYMM);

}

int element(int set[],int size)

{

int i;

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

{

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

}

}

void display(int set[],int size)

{

int i,j,k;

printf("{");

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

{

if(i<size-1)

{

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

}

else if (i==size-1)

{

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

}

}

printf("}\n\n");

}

void bubblesort(int set[], int size)

{

int i, j,temp;

for(i=0;i<size-1;i++)

{

for(j=0;j<size-i-1;j++)

{

if ( set[j]>set[j+1])

{

temp = set[j+1];

set[j+1] = set[j];

set[j] = temp;

}

}

}

}

int main(){

int m,n,i,j,UNION[40],INTERSECTION[40], DIFFA[40],DIFFB[40], SYM[40], INTER[40],SYMM[40];

int union\_num,inter\_num,diff\_numA,diff\_numB,symm\_num;

printf("Enter size of set A: ");

scanf("%d",&m);

printf("Enter size of set B: ");

scanf("%d",&n);

int setA[m], setB[n];

printf("Enter the elements of setA: \n");

element(setA, m);

bubblesort(setA,m);

printf("Enter the elements of setB: \n");

element(setB, n);

bubblesort(setB,n);

//displaying the elemensts of set A and set B

printf("The Elements of the setA: \n");

display(setA,m);

printf("The Elements of the setB: \n");

display(setB,n);

//union set operation

printf("The union of setA and setB: \n");

union\_num = set\_union(setA,m,setB,n,UNION);

bubblesort(UNION,union\_num);

display(UNION,union\_num);

//intersection set operation

printf("The intersection of setA and setB: \n");

inter\_num = Intersection(setA,m,setB,n,INTER);

bubblesort(INTER,inter\_num);

display(INTER,inter\_num);

//difference set operation

printf("The difference of setA and setB (SetA - SetB): \n");

diff\_numA = Difference(setA,m,setB,n,DIFFA);

bubblesort(DIFFA,diff\_numA);

display(DIFFA, diff\_numA);

//difference set operation

printf("Difference of setA and setB (SetB - SetA): \n");

diff\_numB = Difference(setB,m,setA,n,DIFFB);

bubblesort(DIFFB,diff\_numB);

display(DIFFB,diff\_numB);

//symmetric difference set opration

printf("Symmetric difference of setA and setB ((SetA - SetB) U (SetB - SetA)) : \n");

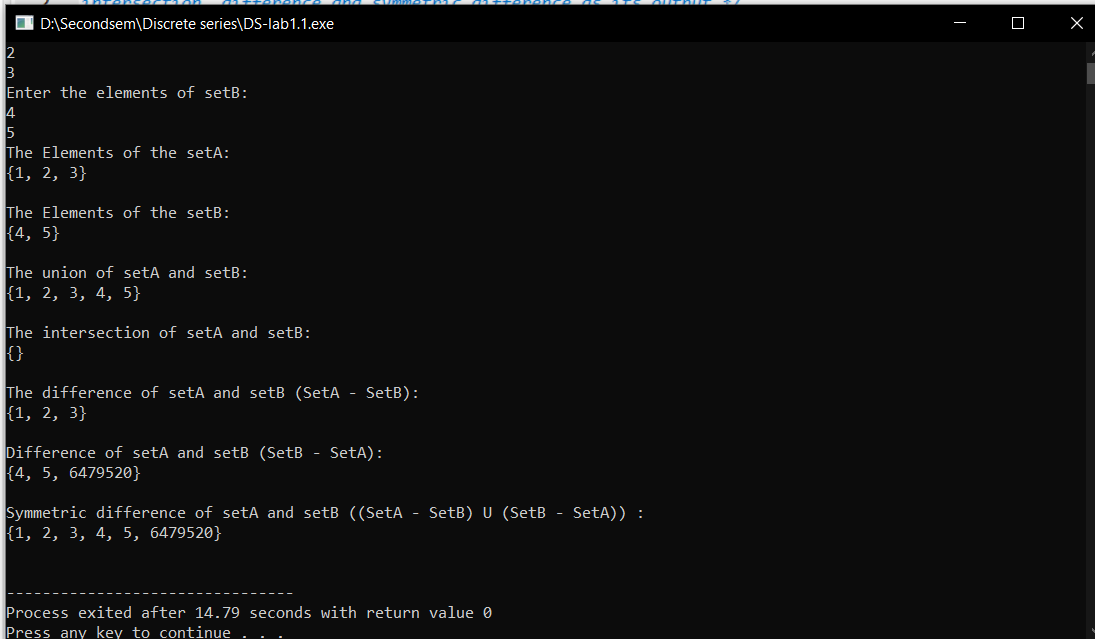
symm\_num = Symmetric(DIFFA,diff\_numA,DIFFB,diff\_numB, SYMM);

bubblesort(SYMM,symm\_num);

display(SYMM, symm\_num);

}

Output:



//Write a program that takes two or more sets as input and produces their Cartesian product as output.

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

void CartProduct(int arr1[],int arr2[],int n1, int n2){

int i,j;

printf("{");

for(i=0; i < n1; i++){

for(j=0; j < n2; j++){

if (j<n2-1){

printf("(%d,%d), ",arr1[i],arr2[j]);

}

else if (j== n2-1){

printf("(%d,%d)",arr1[i],arr2[j]);

}

}

}

printf("}\n");

}

void display(int set[],int size){

int i,j,k;

printf("{");

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

{

if(i<size-1)

{

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

}

else if (i==size-1)

{

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

}

}

printf("}\n\n");

}

int main(){

int n1,n2,i,j;

printf("Enter the size of Set A : ");

scanf("%d",&n1);

printf("Enter the size of Set B: ");

scanf("%d",&n2);

int setA[n1],setB[n2];

printf("Enter the elements of Set A:\n");

for(i=0; i<n1; i++)

{

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

}

printf("Enter the elements of Set B :\n");

for(j=0;j<n2;j++)

{

scanf("%d",&setB[j]);

}

printf("Elements of setA: \n");

display(setA,n1);

printf("Elements of setB: \n");

display(setB,n2);

printf("Cartesain Product of Set A and Set B is: \n ");

CartProduct(setA,setB,n1,n2);

return 0;

}

Output:



//Program to take a real number and produce ceiling and floor integers as output.

#include<stdio.h>

int floorValue(float num){

if (num == (int)(num)){

return num;

}

else if (num>=0){

return (int)(num/1);

}

else{

return (int)(num-1);

}

}

int CeilValue(float num){

if(num == (int)(num)){

return num;

}

else if(num>=0){

return (int)((num/1)+1);

}

else{

return (int)(num);

}

}

int main(){

float x;

int ceil,floor;

printf("Enter the number: ");

scanf("%f",&x);

ceil = CeilValue(x);

floor = floorValue(x);

printf("The Ceiling value of %.2f: %d\n",x,ceil);

printf("The Floor value of %.2f: %d\n", x ,floor);

if ( (int)(x) != x)

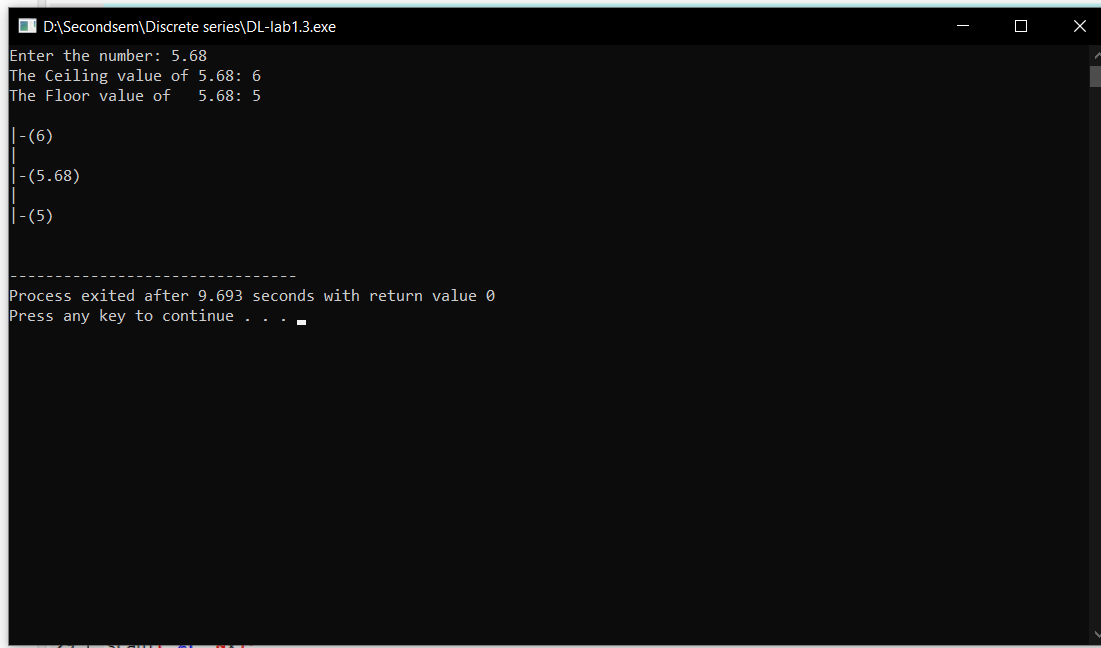
{

printf("\n|-(%d)\n|\n|-(%.2f)\n|\n|-(%d)\n\n",ceil,x,floor);

}

}

Output:



//program to take 5 names and their age as inputs and give degree of membership as output

#include<stdio.h>

#include<stdlib.h>

float degree\_of\_memshiA(int age){

if (age <=20)

{

return 1;

}

else if(age<=30)

{

return (float)(30-age)/10;

}

else

{

return 0;

}

}

float degree\_of\_memshiB(int age){

if (age <=15)

{

return 1;

}

else if(age<=35)

{

return (float)(35-age)/20;

}

else

{

return 0;

}

}

// For Fuzzy Union

void fuzzy\_union(char Name[40][40],float MemshiA[40], float MemshiB[40]){

float union\_Set[20];

int i,j;

for(i=0;i<5;i++)

{

if(MemshiA[i]>MemshiB[i])

{

union\_Set[i]=MemshiA[i];

}

else if(MemshiA[i]< MemshiB[i])

{

union\_Set[i]= MemshiB[i];

}

else

{

union\_Set[i]=MemshiA[i];

}

}

printf("Result of the union fuzzy operation is : \n {");

for(i=0;i<5;i++){

if(i<4){

printf("%.2f/%s,",union\_Set[i],Name[i]);

}

else if(i == 4){

printf("%.2f/%s",union\_Set[i],Name[i]);

}

}

printf("}\n\n");

}

//For Fuzzy intersection

void fuzzy\_intersection(char Name[40][40],float MemshiA[40], float MemshiB[40]){

float intersection\_set[20];

int i,j;

for(i=0;i<5;i++)

{

if(MemshiA[i]>MemshiB[i])

{

intersection\_set[i]=MemshiB[i];

}

else if(MemshiA[i]< MemshiB[i])

{

intersection\_set[i]= MemshiA[i];

}

else

{

intersection\_set[i]=MemshiA[i];

}

}

printf("Result of the intersection fuzzy operation is : \n {");

for(i=0;i<5;i++){

if(i<4){

printf("%.2f/%s, ",intersection\_set[i],Name[i]);

}

else if(i==4){

printf("%.2f/%s",intersection\_set[i],Name[i]);

}

}

printf("}\n\n");

}

//Fuzzy Complement

void fuzzy\_complement(char Name[40][40],float MemshiA[40], float MemshiB[40]){

float complement\_SetA[20],complement\_SetB[20];

int i,j;

for(i=0;i<5;i++)

{

complement\_SetA[i]=1-MemshiA[i];

complement\_SetB[i]=1-MemshiB[i];

}

printf("Result of the Complement fuzzy operation of first set is : \n {");

for(i=0;i<5;i++){

if(i<4){

printf("%.2f/%s, ",complement\_SetA[i],Name[i]);

}

else if(i==4){

printf("%.2f/%s",complement\_SetA[i],Name[i]);

}

}

printf("}\n\n");

printf("Result of the Complement fuzzy operation of second set is : \n {");

for(i=0;i<5;i++){

if(i<4){

printf("%.2f/%s, ",complement\_SetB[i], Name[i]);

}

else if(i==4){

printf("%.2f/%s",complement\_SetB[i],Name[i]);

}

}

printf("}\n\n");

}

int main()

{

int age[40],i=0;

char name[40][40];

float memshiA[20],memshiB[20];

for(i=0;i<5;i++){

printf("Enter the name: ");

scanf("%s",name[i]);

printf("Enter age: ");

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

}

for(i=0;i<5;i++){

memshiA[i]= degree\_of\_memshiA(age[i]);

memshiB[i]= degree\_of\_memshiB(age[i]);

}

printf("First Set is: \n {");

for(i=0;i<5;i++)

{

if(i<4){

printf("%.2f/%s, ",memshiA[i],name[i]);

}

else if(i==4){

printf("%.2f/%s",memshiA[i],name[i]);

}

}

printf("}\n\n");

printf("Second Set is: \n {");

for(i=0;i<5;i++)

{

if(i<4){

printf("%.2f/%s, ",memshiB[i],name[i]);

}

else if(i==4){

printf("%.2f/%s",memshiB[i],name[i]);

}

}

printf("}\n\n");

fuzzy\_union(name, memshiA, memshiB);

fuzzy\_intersection(name,memshiA,memshiB);

fuzzy\_complement(name,memshiA,memshiB);

}

Output:

