Discrete Structure

We have coded two different codes. First one tells number of possible relations that a set can hold through formulas.

In second code, considering the Cartesian product a relation it checks whether the relation holds a particular relation or not. It Is menu driven.

Code 1 (Formulas)

# Input:

#include<iostream>

#include<math.h>

using namespace std;

bool checkRepeat(int set[], int index);

int main()

{

int\* Set1, \* Set2;

int size1, size2, relationSize = 0;

do

{

cout << "Enter Size of Set Set1 (MIN 3) : ";

cin >> size1;

if (size1 < 3)

cout << "Minimum size should be 3 \n";

} while (size1 < 3);

Set1 = new int[size1];

cout << "Enter Elements of Set Set1 : \n";

for (int i = 0; i < size1;)

{

cin >> Set1[i];

if (checkRepeat(Set1, i))

i++;

else

cout << "Numbers can't be repeated in Sets\n";

}

do

{

cout << "Enter Size of Set Set2 (MIN 3) : ";

cin >> size2;

} while (size1 < 3);

Set2 = new int[size2];

cout << "Enter Elements of Set Set2 : \n";

for (int i = 0; i < size2; )

{

cin >> Set2[i];

if (checkRepeat(Set2, i))

i++;

else

cout << "Numbers can't be repeated in Sets ReEnter\n";

}

//Cartesian Product

cout << "Cartesian Product = ";

cout << "{";

for (int i = 0; i < size1; i++)

{

for (int j = 0; j < size2; j++)

{

cout << "(" << Set1[i] << "," << Set2[j] << ")";

}

}

cout << "}\n";

relationSize = size1 \* size2;

//No of Relations

cout << "Number of Relations : " << pow(2, relationSize) << endl;

//No of Reflexive Relations 2^n(n-1).

cout << "\nNumber of Reflexive Relations : " << pow(2, (relationSize \* (relationSize - 1))) << endl;

cout << "Number of Reflexive Relations on set 1 : " << pow(2, (size1 \* (size1 - 1))) << endl;

cout << "Number of Reflexive Relations on set 2 : " << pow(2, (size2 \* (size2 - 1))) << endl;

//No of Symmetric Relations 2^(n(n + 1)/2)

cout << "\n\nNumber of Symmetric Relations : " << pow(2, (relationSize \* (relationSize + 1) / 2)) << endl;

cout << "Number of Symmetric Relations on set 1 : " << pow(2, (size1 \* (size1 + 1))) << endl;

cout << "Number of Symmetric Relations on set 2 : " << pow(2, (size2 \* (size2 + 1))) << endl;

//No of Anti-Symmetric Relations 2^n 3^n(n-1)/2.

cout << "\n\nNumber of Anti-Symmetric Relations " << pow(2, relationSize) \* pow(3, (relationSize \* (relationSize - 1) / 2)) << endl;

cout << "Number of Anti-Symmetric Relations on set1 : " << pow(2, size1) \* pow(3, (size1 \* (size1 - 1) / 2)) << endl;

cout << "Number of Anti-Symmetric Relations on set2 : " << pow(2, size2) \* pow(3, (size2 \* (size2 - 1) / 2)) << endl;

//No of Symmetric and Anti-Symmetric Relations 2n

cout << "\n\nNumber of symmetric and anti symmetric relations : " << 2 \* relationSize << endl;

cout << "Number of symmetric and anti symmetric relations on set 1 : " << 2 \* size1 << endl;

cout << "Number of symmetric and anti symmetric relations on set 2 : " << 2 \* size2 << endl;

//No of Reflexive and Symmetric Relations 2^n(n-1)/2.

cout << "\n\nNumber of that are both Reflexive and Symmetric " << pow(2, (relationSize \* (relationSize - 1) / 2)) << endl;

cout << "Number of that are both Reflexive and Symmetric on set1 : " << pow(2, (size1 \* (size1 - 1) / 2)) << endl;

cout << "Number of that are both Reflexive and Symmetric on set2 : " << pow(2, (size2 \* (size2- 1) / 2)) << endl;

system("pause");

return 0;

}

bool checkRepeat(int set[], int index)

{

for (int i = 0; i < index; i++)

{

if (set[i] == set[index])

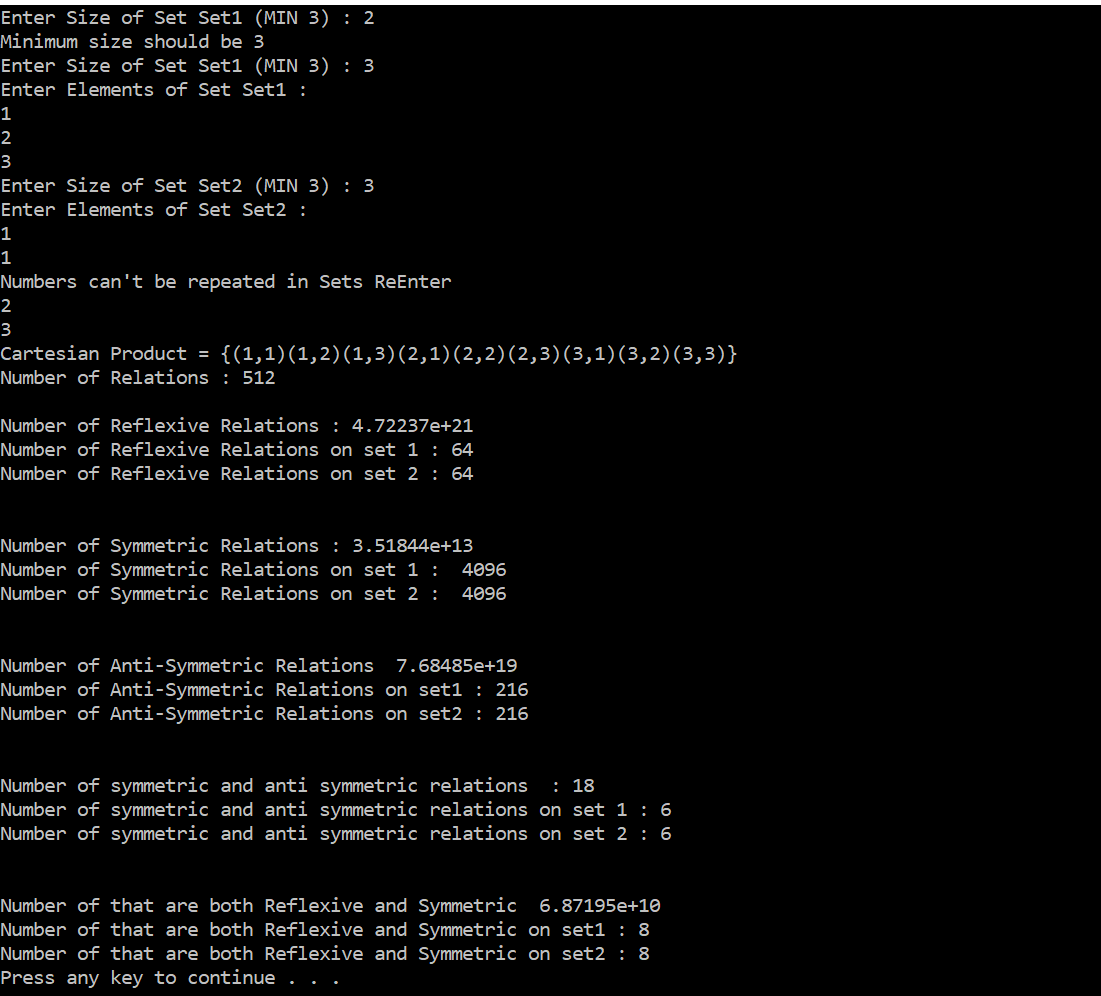
return false;

}

return true;

}

# Output:



Code 2(Logics)

# Input:

#include<iostream>

#include<Windows.h>

#include<math.h>

using namespace std;

void menu();

void DisplayRelation(int set1[], int set2[], int size1, int size2);

bool checkRepeat(int set[], int index);

bool reflexive(int set1[], int set2[], int size1, int size2);

bool symmetric(int set1[], int set2[], int size1, int size2);

bool antisymmetric(int set1[], int set2[], int size1, int size2);

int main()

{

int\* Set1, \* Set2, ch;

int size1, size2, count = 0, count1 = 0;

bool anti = false;

do

{

cout << "Enter Size of Set Set1 (MIN 3) : ";

cin >> size1;

if (size1 < 3)

cout << "Minimum size should be 3 \n";

} while (size1 < 3);

Set1 = new int[size1];

cout << "Enter Elements of Set Set1 : \n";

for (int i = 0; i < size1;)

{

cin >> Set1[i];

if (checkRepeat(Set1,i))

i++;

else

cout << "Numbers can't be repeated in Sets\n";

}

do

{

cout << "Enter Size of Set Set2 (MIN 3) : ";

cin >> size2;

if (size2 < 3)

cout << "Minimum size should be 3 \n";

} while (size2 < 3);

Set2 = new int[size2];

cout << "Enter Elements of Set Set2 : \n";

for (int i = 0; i < size2; )

{

cin >> Set2[i];

if (checkRepeat(Set2, i))

i++;

else

cout << "Numbers can't be repeated in Sets\n";

}

cout << endl << endl;

while (true)

{

DisplayRelation(Set1, Set2, size1, size2);

menu();

cin >> ch;

switch (ch)

{

case 1:

cout << "Number of Relations = " << pow(2, (size1 \* size2)) << endl;

break;

case 2:

if (size1 < size2)

reflexive(Set2, Set1, size2, size1);

else

reflexive(Set1, Set2, size1, size2);

break;

case 3:

symmetric(Set1, Set2, size1, size2);

break;

case 4:

antisymmetric(Set1, Set2, size1, size2);

break;

case 5:

if (antisymmetric(Set1, Set2, size1, size2))

{

system("cls");

DisplayRelation(Set1, Set2, size1, size2);

menu();

cout << "Yes it has both Symmetric and Anti Symmetric Relation\n";

}

else

{

cout << "It does not have both relations\n";

}

break;

case 6:

if (reflexive(Set1, Set2, size1, size2) && symmetric(Set1, Set2, size1, size2))

{

system("cls");

DisplayRelation(Set1, Set2, size1, size2);

menu();

cout << "Yes it has both Symmetric and Reflexive Relation\n";

}

break;

case 7:

return 0;

default:

cout << "Enter a valid Input\n";

break;

}

}

system("pause");

return 0;

}

/////////////////// Functions ////////////////////////

void menu()

{

cout << "\n1.) No of Relations\n2.) Check Reflexive Relation\n3.) Check Symmetric Relation\n";

cout << "4.) Check Anti\_Symmetric Relation\n5.) check Anti Symmetric and Symmetric\n";

cout<<"6.) Check Reflexive and Symmetric Relations\n7.) Exit\nEnter : ";

}

void DisplayRelation(int set1[], int set2[], int size1, int size2)

{

cout << "\nSet1 : { ";

for (int i = 0; i < size1; i++)

{

cout << set1[i] << " ";

}

cout << "}\nSet2 : { ";

for (int i = 0; i < size2; i++)

{

cout << set2[i] << " ";

}

cout << "}\n\nCartesian Product : {";

for (int i = 0; i < size1; i++)

{

for (int j = 0; j < size2; j++)

{

cout << "(" << set1[i] << "," << set2[j] << ")";

}

}

}

bool checkRepeat(int set[],int index)

{

for (int i = 0; i < index; i++)

{

if (set[i] == set[index])

return false;

}

return true;

}

bool reflexive(int set1[], int set2[], int size1, int size2)

{

int count = 0;

bool reflexive;

for (int i = 0; i < size1; i++)

{

reflexive = false;

for (int j = 0; j < size2; j++)

{

if (set1[i] == set2[j])

{

reflexive = true;

count++;

break;

}

}

if (!reflexive)

{

cout << "This is not a reflexive Relation as every element is not related to itself\n";

break;

return false;

}

}

cout << "Yes it has reflexive relation\n";

return true;

}

bool symmetric(int set1[], int set2[], int size1, int size2)

{

int temp1, temp2, count = 0;

bool sym = false;

for (int i = 0; i < size1; i++)

{

for (int j = 0; j < size2; j++)

{

sym = false;

if (set1[i] == set2[j])

{

cout << "( " << set1[i] << " , " << set2[j] << " ) ";

sym = true;

}

else

{

temp1 = set1[i]; temp2 = set2[j];

for (int k = 0; k < size1; k++)

{

for (int l = 0; l < size2; l++)

{

if (l == j)

continue;

if (temp2 == set1[k])

{

sym = true;

cout << "( " << temp1 << " , " << temp2 << " ) ";

}

}

}

}

if (!sym)

{

cout << set1[i] << "It is not symmetric Relation\n";

return false;

}

}

}

cout << "\nIt is a symmetric Relation\n";

return true;

}

bool antisymmetric(int set1[], int set2[], int size1, int size2)

{

int temp1, temp2, count = 0;

bool anti1, anti2;

for (int i = 0; i < size1; i++)

{

for (int j = 0; j < size2; j++)

{

anti1 = false;

anti2 = false;

if (set1[i] == set2[j])

{

cout << "( " << set1[i] << " , " << set2[j] << " ) ";

anti1 = true;

}

else

{

temp1 = set1[i]; temp2 = set2[j];

for (int k = 0; k < size1; k++)

{

for (int l = 0; l < size2; l++)

{

if (l == j)

continue;

if (temp2 == set1[k])

{

anti2 = true;

cout << "\n\nVoilation : ( " << temp1 << " , " << temp2 << " )\n";

}

}

}

}

if (anti2)

{

cout << "\nIt is not Anti Symmetric Relation\n";

return false;

}

}

}

cout << "\nIt is Anti Symmetric Relation\n";

return true;

}

# Output:

