**Software Testing (SE 301a)**

**LAB FILE**

**(2023-2024)**

**Submitted By**

**Sanoj (2K21/SE/159)**

**Submitted to**

**Prof. Ruchika Malhotra**

**(Head of Department, Software Engineering)**



# DELHI TECHNOLOGICAL UNIVERSITY

# (Formerly Delhi College of Engineering)

**Bawana Road, Delhi-110042**

**Index**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Experiment Name** | **Date** | **Teacher Sign** | **Remarks** |
| 1. | Write a program to find the largest among three numbers and generate test cases using boundary value analysis. | 18 Aug 2023 |  |  |
| 2. | Write a program to find the largest among three numbers and generate test cases using Robust Test Case Approach. | 25 Aug 2023 |  |  |
| 3. | Write a program to find the largest among three numbers and generate test cases using Worst Test Case Approach. | 1 Sep 2023 |  |  |
| 4. | Write a program to find the largest among three numbers and generate test cases using Worst Robust Test Case Approach. | 15 Sep 2023 |  |  |
| 5. | Write a program to find the Triangle types on the basis of the input sides given and generate test cases using Equivalence Class Testing. | 22 Sep 2023 |  |  |
| 6. | Write a program to find the Triangle types on the basis of the input sides given and generate test cases using Decision Table Testing. | 6 Oct 2023 |  |  |
| 7. | Write a program to find the Cyclomatic Complexity of a program. | 13 Oct 2023 |  |  |
| 8. | Write a program to perform Mutation Testing | 20 Oct 2023 |  |  |
| 9. | Write a program to perform DD Path testing | 27 Oct 2023 |  |  |
| 10. |  |  |  |  |

**Experiment - 1**

**Aim:** Write a program to find the largest among three numbers and generate test cases using boundary value analysis.

**Code:**

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n, a[3], x, c, maxLimits[3], min[3], e[3][4];

n = 3;

cout << "Enter the minimum and maximum limits of the variables:\n";

for (int y = 0; y < n; y++)

{

cin >> min[y] >> maxLimits[y];

}

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

{

a[x] = (min[x] + maxLimits[x]) / 2;

}

for (int a1 = 0; a1 < n; a1++)

{

e[a1][0] = min[a1];

e[a1][1] = min[a1] + 1;

e[a1][2] = maxLimits[a1] - 1;

e[a1][3] = maxLimits[a1];

}

cout << " " <<"V1"<< " " <<"V2"<< " " << "V3" <<" " <<"Expected Value"<<endl;

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

{

for (int j = 0; j < 4; j++)

{

int ma = 0;

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

{

if (l != 0)

{

cout << "\t" << a[l];

ma = max(ma, a[l]);

}

}

cout << "\t" << e[i][j];

ma = max(ma,e[i][j]);

for (int k = 0; k < n - (i + 1); k++)

{

cout << "\t" << a[k + i + 1];

ma = max(ma, a[k + i + 1]);

}

cout << "\t" << ma;

cout << "\n";

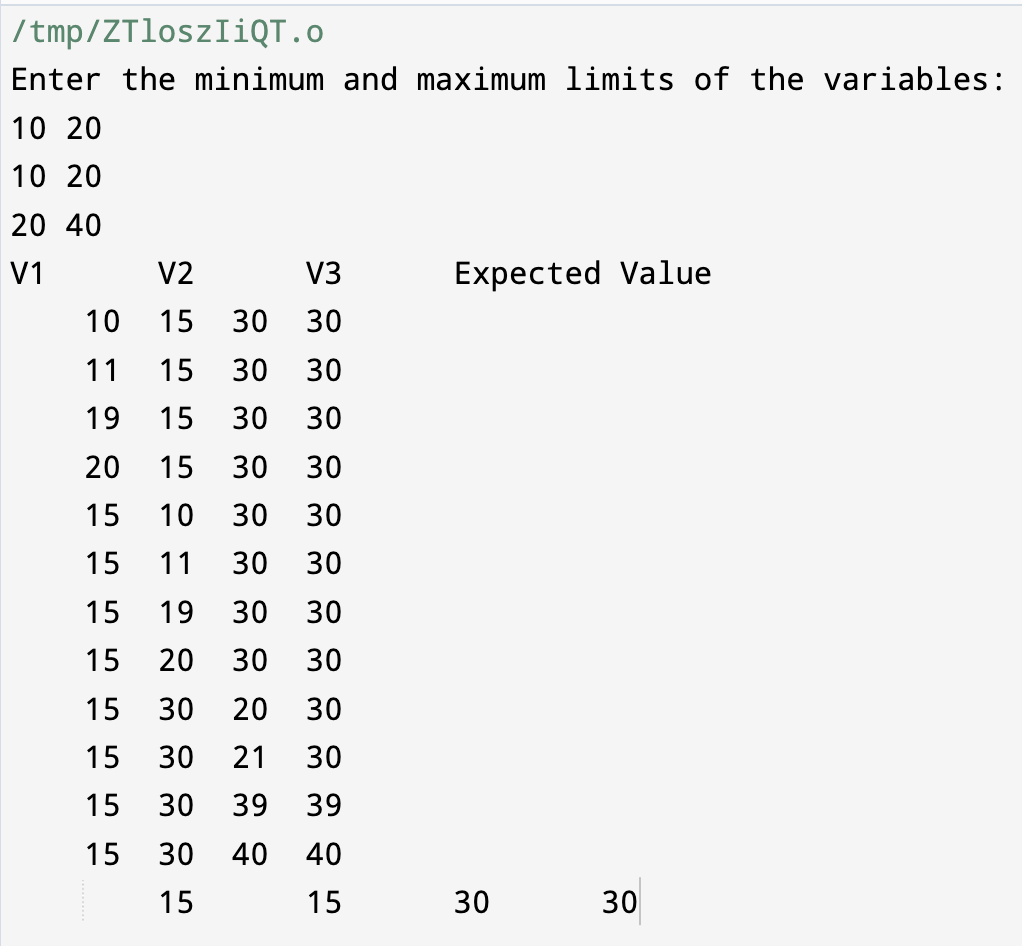
}

}

cout << " " << a[0] << " " << a[1] << " " << a[2] <<" " << a[2];

}

**Output:**

****

**Experiment - 2**

**Aim:** Write a program to find the largest among three numbers and generate test cases using Robust Test Case Approach.

**Code:**

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n, a[3], x, c, maxLimits[3], min[3], e[3][6];

n = 3;

cout << "Enter the minimum and maximum limits of the variables:\n";

for (int y = 0; y < n; y++)

{

cin >> min[y] >> maxLimits[y];

}

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

{

a[x] = (min[x] + maxLimits[x]) / 2;

}

for (int a1 = 0; a1 < n; a1++)

{

e[a1][0] = min[a1];

e[a1][1] = min[a1] + 1;

e[a1][2] = min[a1] - 1;

e[a1][3] = maxLimits[a1] - 1;

e[a1][4] = maxLimits[a1];

e[a1][5] = maxLimits[a1]+1;

}

cout << " " <<"V1"<< " " <<"V2"<< " " << "V3" <<" " <<"Expected Value"<<endl;

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

{

for (int j = 0; j < 6; j++)

{

int ma = 0;

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

{

if (l != 0)

{

cout << "\t" << a[l];

ma = max(ma, a[l]);

}

}

cout << "\t" << e[i][j];

ma = max(ma,e[i][j]);

for (int k = 0; k < n - (i + 1); k++)

{

cout << "\t" << a[k + i + 1];

ma = max(ma, a[k + i + 1]);

}

cout << "\t" << ma;

cout << "\n";

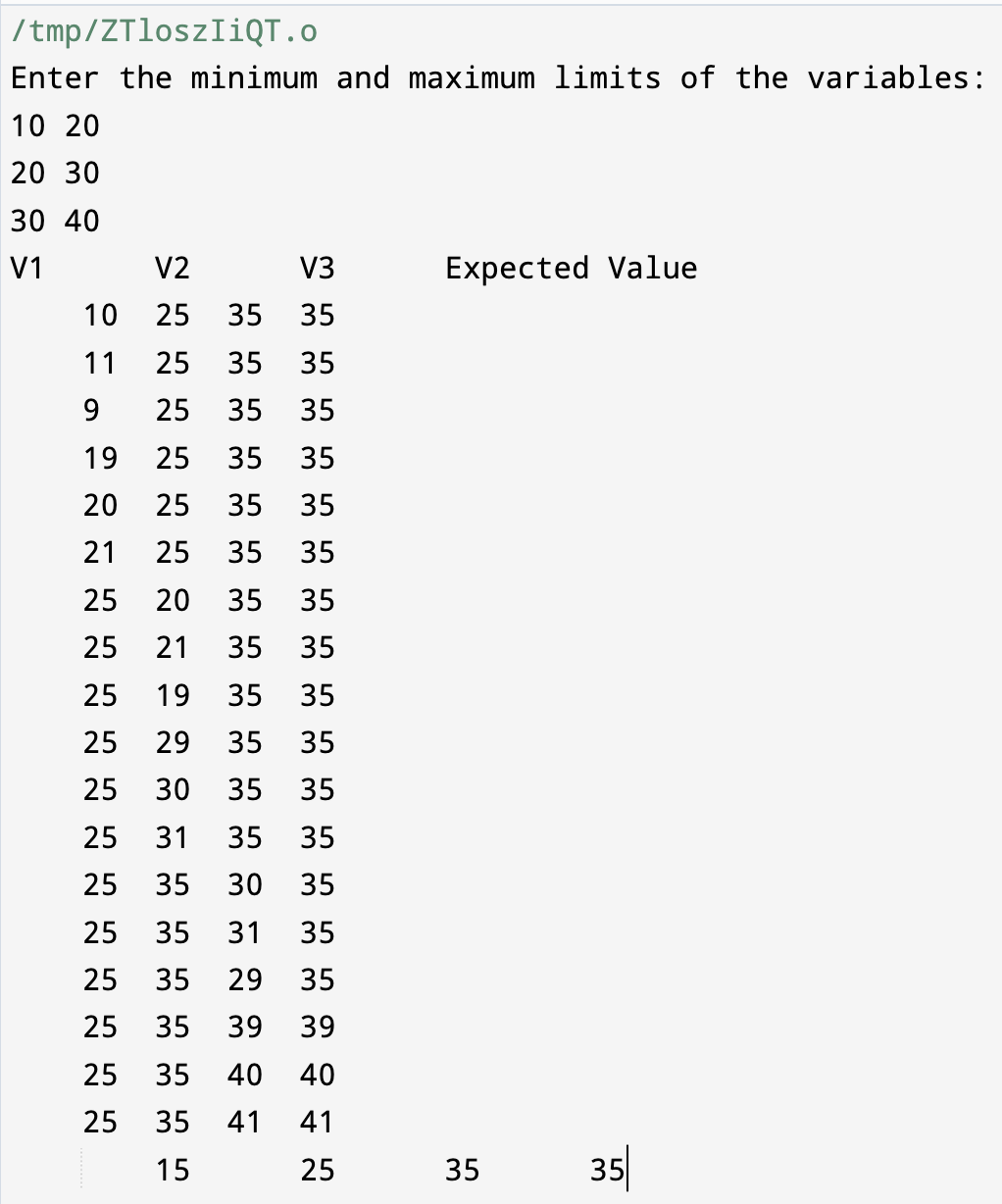
}

}

cout << " " << a[0] << " " << a[1] << " " << a[2] <<" " << a[2];

}

**Output:**



**Experiment - 3**

**Aim:** Write a program to find the largest among three numbers and generate test cases using Worst Test Case Approach.

**Code:**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int minVal, maxVal;

cout << "Enter the minimum value: ";

cin >> minVal;

cout << "Enter the maximum value: ";

cin >> maxVal;

int testValues[5] = { minVal, minVal + 1, (minVal + maxVal) / 2, maxVal - 1, maxVal };

cout << "-------------------------------------------------------------" << endl;

cout << "Test Case | Variable 1 | Variable 2 | Variable 3 | Expected Max" << endl;

cout << "-------------------------------------------------------------" << endl;

int testCase = 1;

for (int a : testValues) {

for (int b : testValues) {

for (int c : testValues) {

int maxNum = max(a, max(b, c));

cout << setw(10) << testCase << setw(13) << a << setw(13) << b << setw(13) << c << setw(15) << maxNum << endl;

testCase++;

}

}

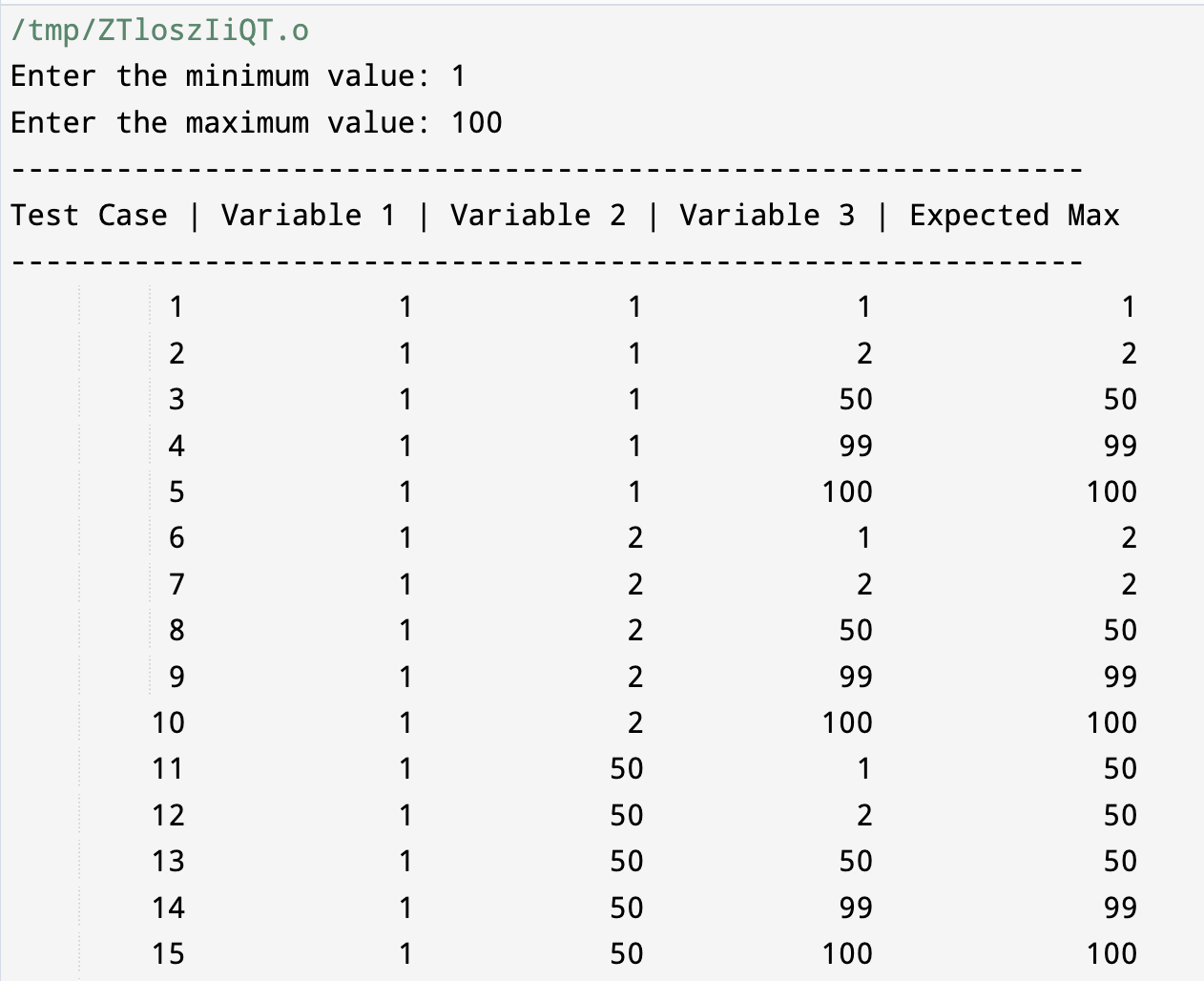
}

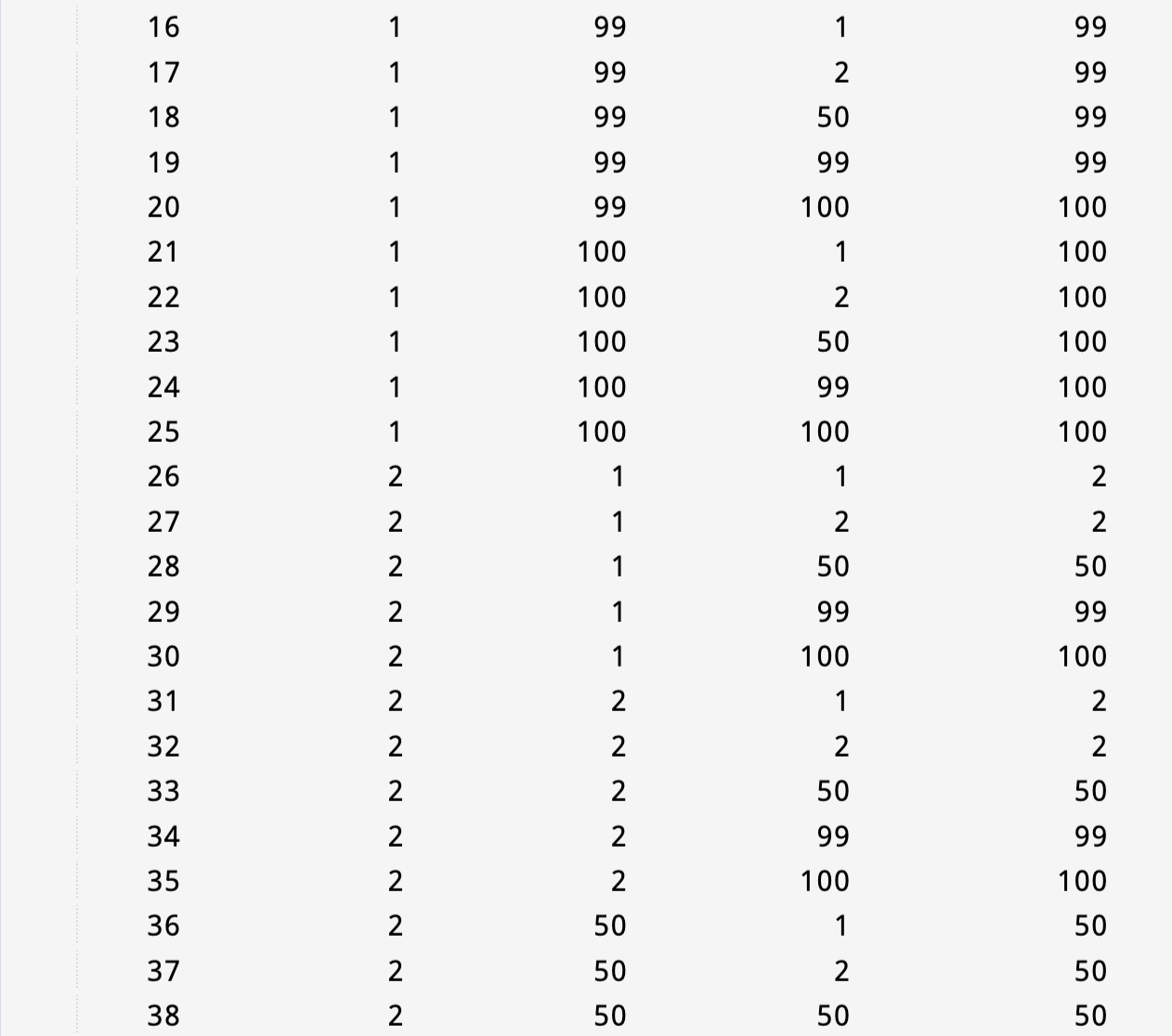
cout << "-------------------------------------------------------------" << endl;

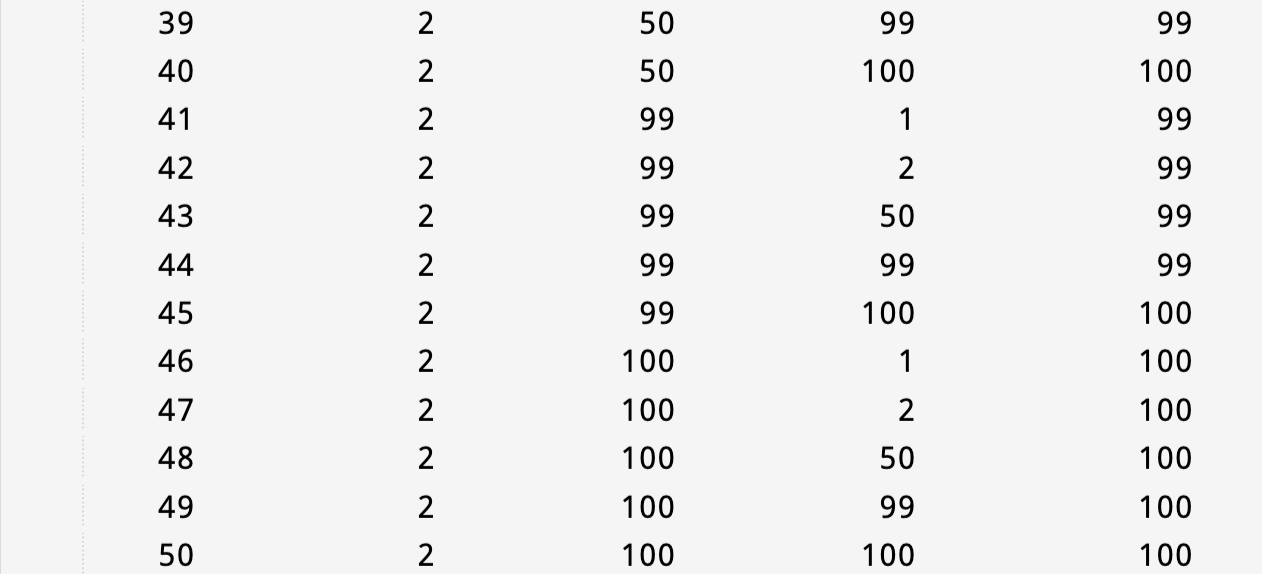
return 0;

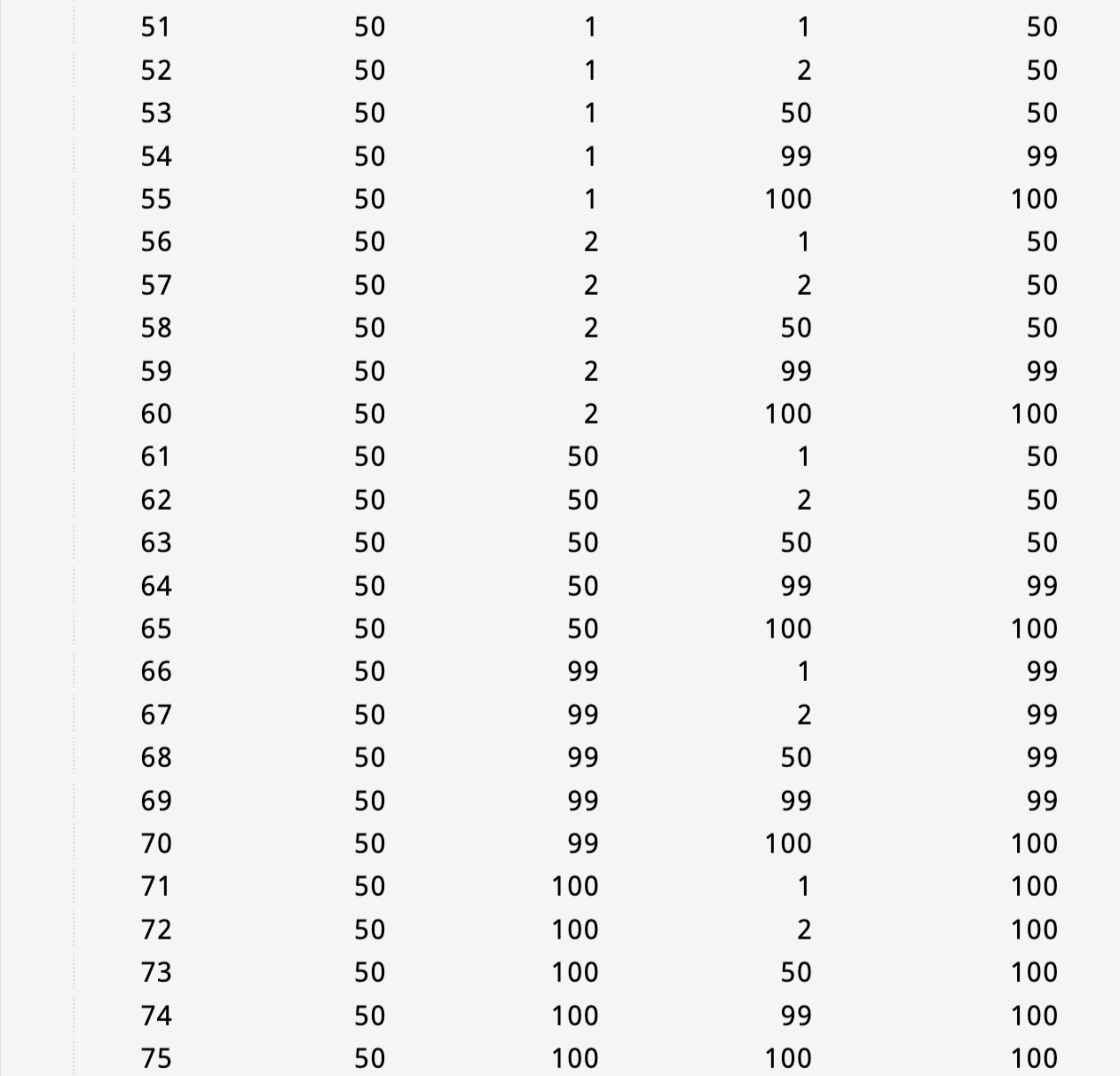
}

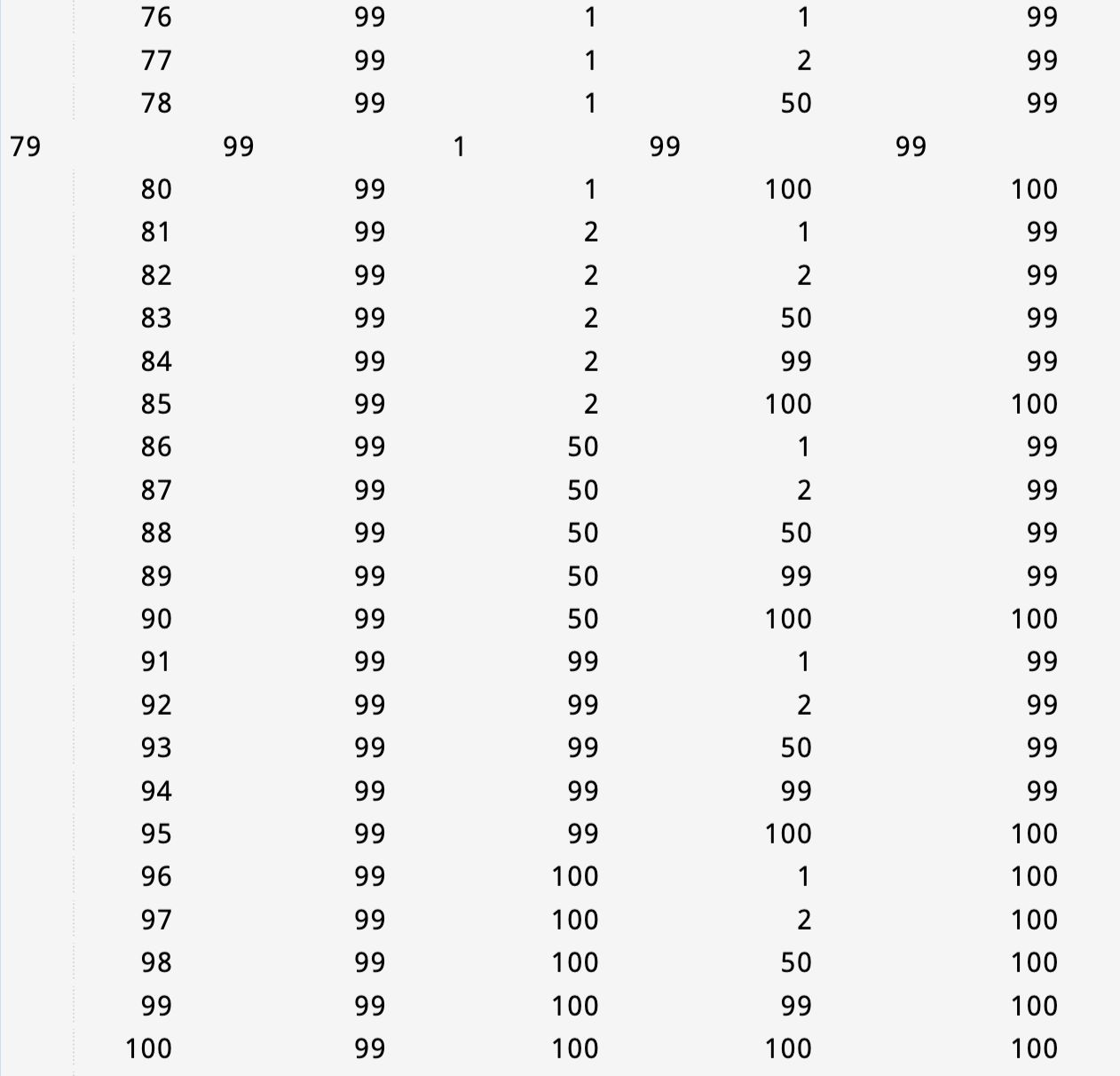
**Output:**

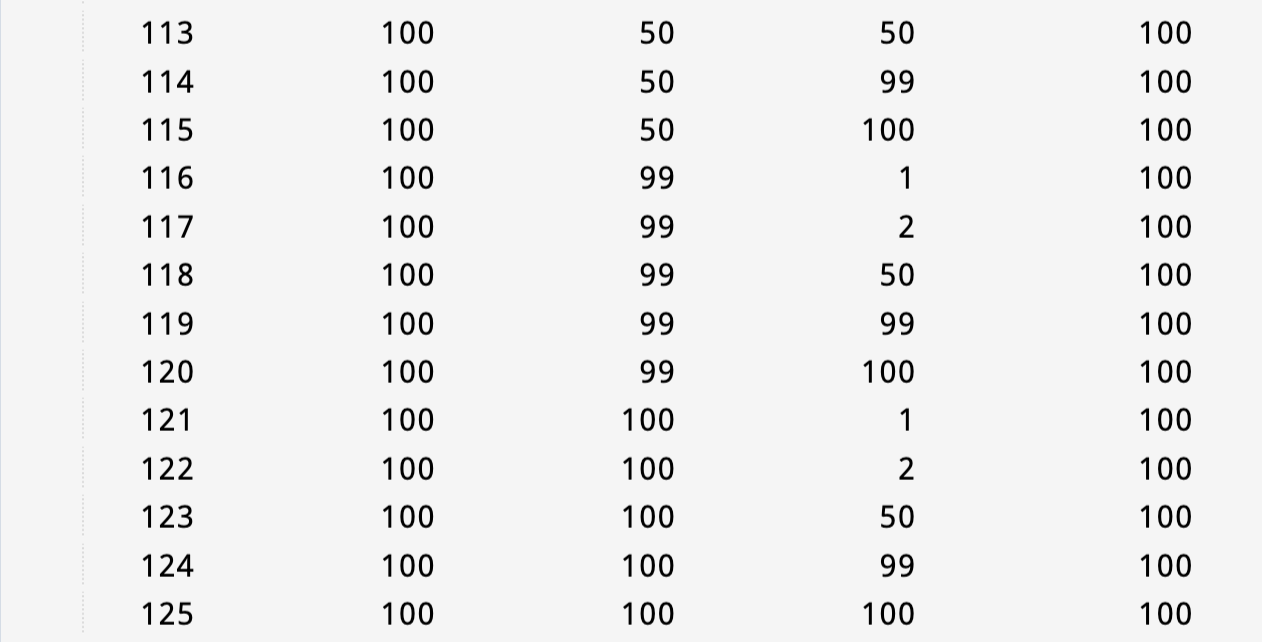












**Experiment - 4**

**Aim:** Write a program to find the largest among three numbers and generate test cases using Worst Robust Test Case Approach.

**Code:**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int minVal, maxVal;

cout << "Enter the minimum value: ";

cin >> minVal;

cout << "Enter the maximum value: ";

cin >> maxVal;

int testValues[7] = {minVal-1, minVal, minVal + 1, (minVal + maxVal) / 2, maxVal - 1, maxVal,maxVal+1 };

cout << " ---------------------------------------------------------------" << endl;

cout << " Test Case | Variable 1 | Variable 2 | Variable 3 | Expected Max" << endl;

cout << " ---------------------------------------------------------------" << endl;

int testCase = 1;

for (int a : testValues) {

for (int b : testValues) {

for (int c : testValues) {

int maxNum = max(a, max(b, c));

cout << setw(10) << testCase << setw(13) << a << setw(13) << b << setw(13) << c << setw(15) << maxNum << endl;

testCase++;

}

}

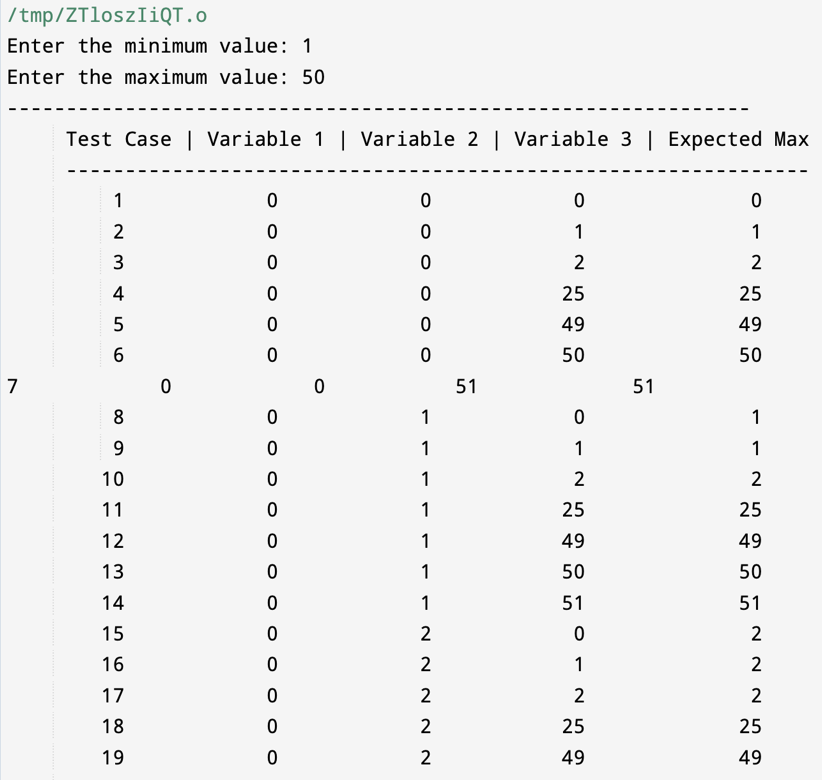
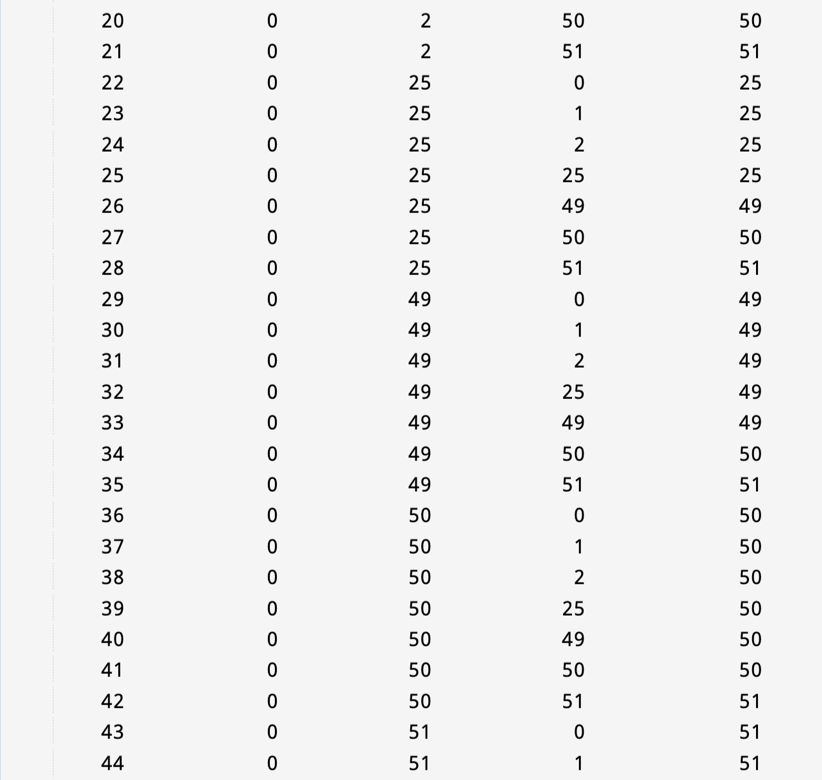
}

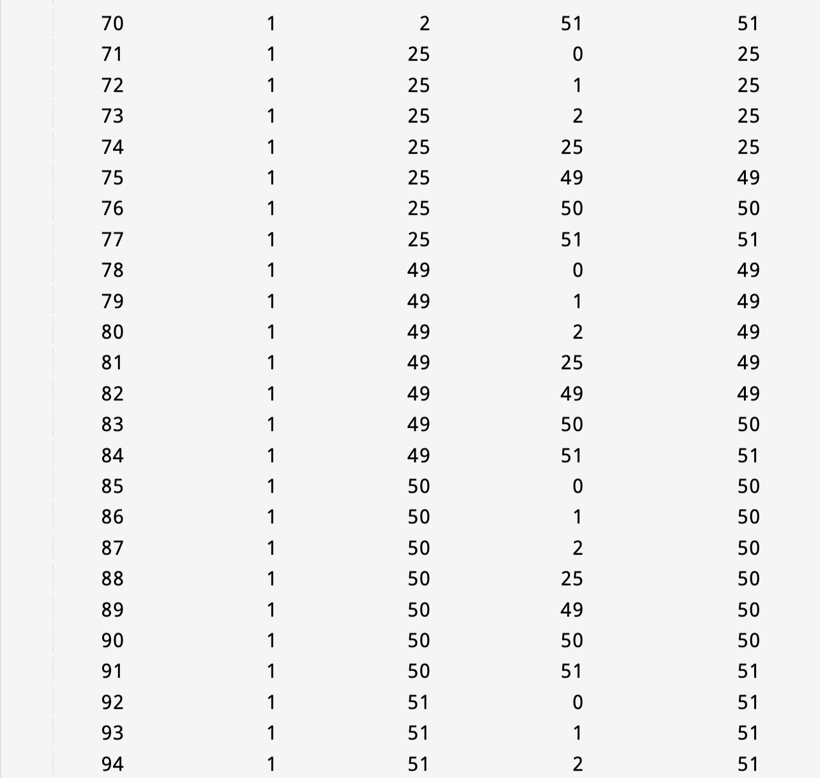
cout << "-------------------------------------------------------------" << endl;

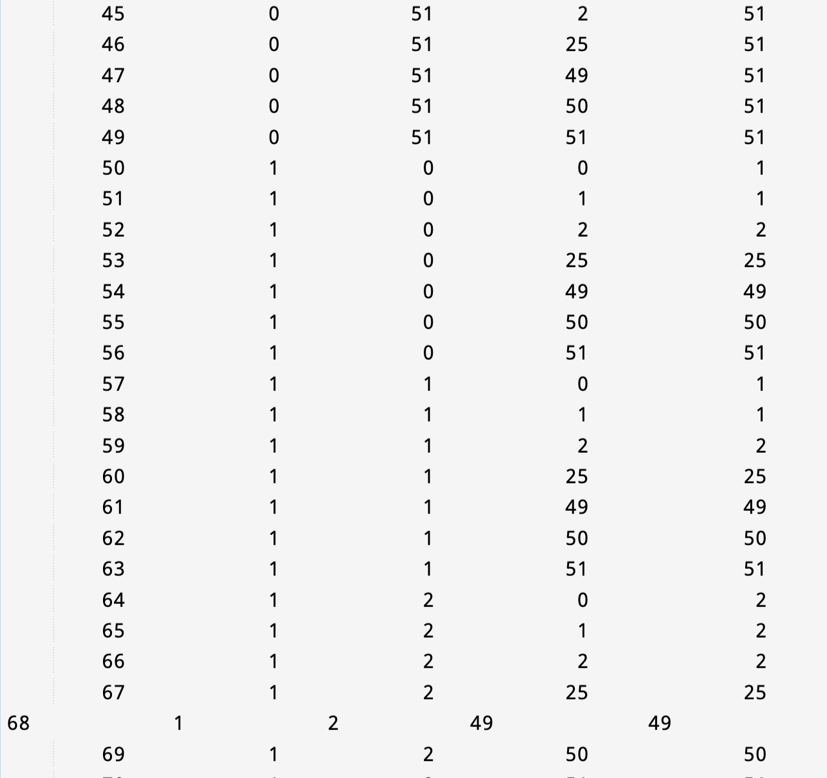
return 0;

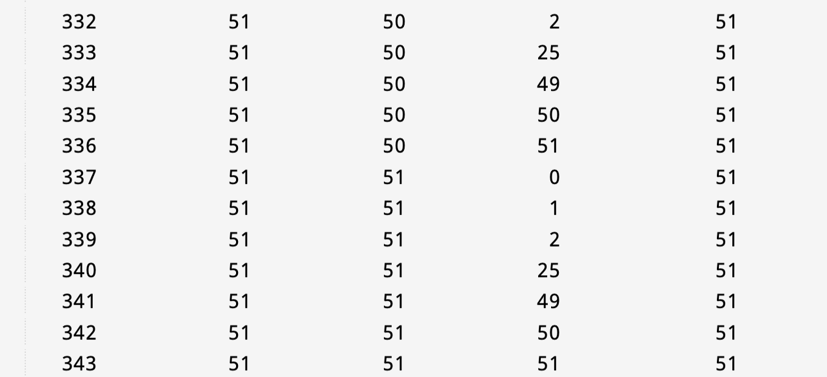
}

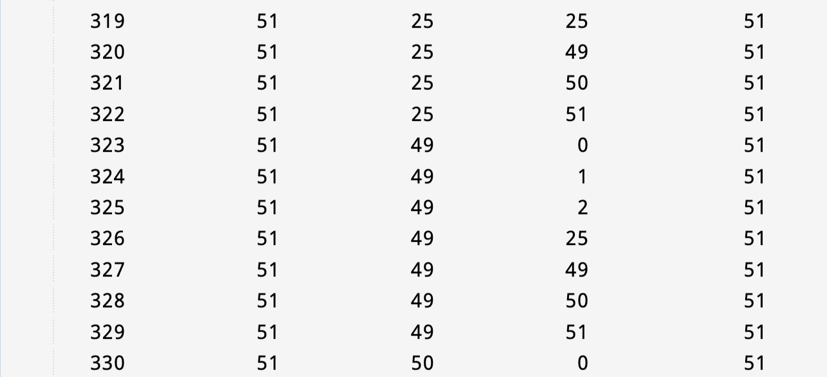
**Output:**

****









**Experiment - 5**

**Aim:** Write a program to find the Triangle types on the basis of the input sides given and generate test cases using Equivalence Class Testing.

**Code:**

#include <iostream>

#include <string>

using namespace std;

string Output\_classification(int a , int b , int c){

if((a+b<c||a+c<b||b+c<a)||(a==0||b==0||c==0)){

return "Not A triangle";

}

else{

if(a!=b&&b!=c&&c!=a){

return "Scalene";

}

else if(a==b&&b==c){

return "Equilateral";

}

else {

return "isosceles";

}

}

}

int main(){

//input class

int rl;

int rh;

cout<<"Enter Lower Range for triangle for input ";

cin>>rl;

cout<<endl;

cout<<"Enter Higher Range for triangle for input ";

cin>>rh;

cout<<endl;

int arr[3]={(rl-1),(rl+rh)/2,(rh+1)};

int a=-1;

int b=-1;

int flag=0;

int count=1;

for(int i=0;i<=26;i++){

if(b%3==0){b++;}

if(i%9==0&&i%3==0){

a++;

b=0;}

if(i==0){

a=0;

b=0;

}

// if(arr[a]==arr[b]&&arr[b]==arr[i%3] && flag==1) continue;

// if(arr[a]==arr[b]&&arr[b]==arr[i%3] && flag==0) flag=1;

if(arr[a]==arr[b]&&arr[b]==arr[i%3]&&flag==0&&((arr[a]!=0||arr[b]!=0||arr[i%3]!=0))){

flag=1;

count++;

string ans=Output\_classification(arr[a],arr[b],arr[i%3]);

cout<<to\_string(count)<<"\t"<<to\_string(arr[a])<<"\t"<<to\_string(arr[b])<<"\t"<<to\_string(arr[i%3])<<"\t"<<ans<<endl;

}

if(arr[a]==arr[b]&&arr[b]==arr[i%3]&&flag==1){}

else{

string ans=Output\_classification(arr[a],arr[b],arr[i%3]);

cout<<to\_string(count)<<"\t"<<to\_string(arr[a])<<"\t"<<to\_string(arr[b])<<"\t"<<to\_string(arr[i%3])<<"\t"<<ans<<endl;

count++;

}

}

return 0;

}

**Output:**

****

**Experiment - 6**

**Aim:** Write a program to find the Triangle types on the basis of the input sides given and generate test cases using Decision Table Testing.

**Code:**

// Online C++ compiler to run C++ program online

#include <iostream>

using namespace std;

int main() {

// Write C++ code here

cout<<"-------------------- Decision Table Testing --------------------"<<endl;

cout<<"INV - Invalid Triangle"<<endl;

cout<<"RAT - Right Angled Triangle"<<endl;

cout<<"OAT - Obtuse Angled Triangle"<<endl;

cout<<"AAT - Acute Angled Triangle"<<endl;

cout<<"IMP - Impossible"<<endl;

cout<<endl;

cout<<"Condtition "<<" 1"<<" 2"<<" 3"<<" 4"<<" 5"<<" 6"<<" 7"<<" 8"<<" 9"<<" 10"<<" 11"<<endl;

cout<<"c1:a<b+c? "<<" F"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<endl;

cout<<"c2:b<a+c? "<<" -"<<" F"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<endl;

cout<<"c3:c<a+b? "<<" -"<<" -"<<" F"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<" T"<<endl;

cout<<"c4:a^2=b^2+c^2 ? "<<" -"<<" -"<<" -"<<" T"<<" T"<<" T"<<" T"<<" F"<<" F"<<" F"<<" F"<<endl;

cout<<"c5:a^2>b^2+c^2 ? "<<" -"<<" -"<<" -"<<" T"<<" T"<<" F"<<" F"<<" T"<<" T"<<" F"<<" F"<<endl;

cout<<"c6:a^2<b^2+c^2 ? "<<" -"<<" -"<<" -"<<" T"<<" F"<<" T"<<" F"<<" T"<<" F"<<" T"<<" F"<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"Rule Count "<<" 32"<<" 16"<<" 8"<<" 1"<<" 1"<<" 1"<<" 1"<<" 1"<<" 1"<<" 1"<<" 1"<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"INV "<<" X"<<" X"<<" X"<<" "<<" "<<" "<<" "<<" "<<" "<<" "<<" "<<endl;

cout<<"RAT "<<" "<<" "<<" "<<" "<<" "<<" "<<" X"<<" "<<" "<<" "<<" "<<endl;

cout<<"OAT "<<" "<<" "<<" "<<" "<<" "<<" "<<" "<<" "<<" X"<<" "<<" "<<endl;

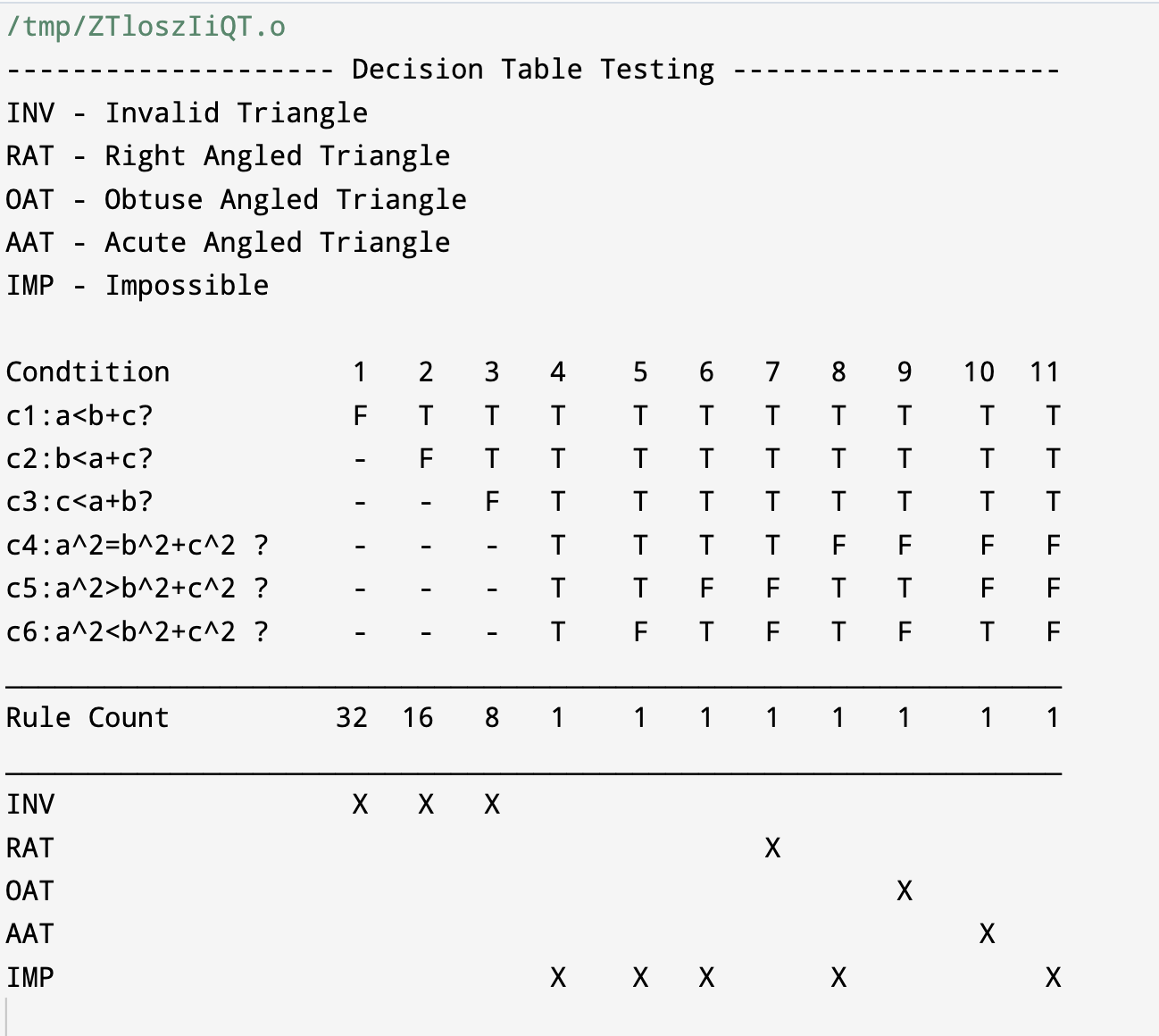
cout<<"AAT "<<" "<<" "<<" "<<" "<<" "<<" "<<" "<<" "<<" "<<" X"<<" "<<endl;

cout<<"IMP "<<" "<<" "<<" "<<" X"<<" X"<<" X"<<" "<<" X"<<" "<<" "<<" X"<<endl;

return 0;

}

**Output:**

****

**Experiment - 7**

**Aim:** Write a program to find the Cyclomatic Complexity of a program.

**Code:**

#include <iostream>

#include <string>

#include <regex>

using namespace std;

int calculateCyclomaticComplexity(const string& code) {

regex decisionPointRegex(R"(if|else|for|while|case|default|do|return|\|\||&&|\?)");

auto decisionPoints =sregex\_iterator(code.begin(), code.end(), decisionPointRegex);

int complexity = distance(decisionPoints,sregex\_iterator());

// The final complexity is the number of decision points plus 1

return complexity + 1;

}

int main() {

string code = R"(

void exampleFunction(int x) {

if (x > 0) {

// Some code

}

if(a>b){

//

}

for (int i = 0; i < 10; ++i) {

// Another code block

}

while{

//

}

}

)";

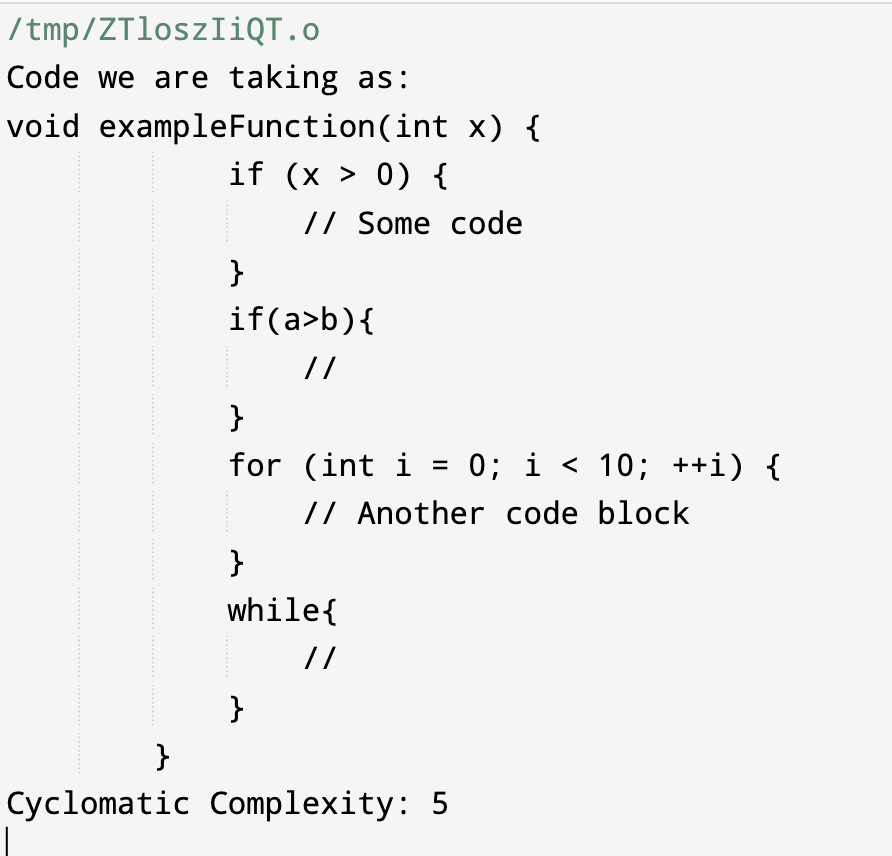
int complexity = calculateCyclomaticComplexity(code);

cout << "Cyclomatic Complexity: " << complexity <<endl;

return 0;

}

**Output:**



**Experiment - 8**

**Aim:** Write a program to perform Mutation Testing.

**Code:**

// Online C++ compiler to run C++ program online

#include <bits/stdc++.h>

using namespace std;

int greatest(int A, int B, int C) {

// changed here A > B to A < B

if(A<B) {

if(A>C) {

// cout<<"Largest Number is "<<A<<endl;

return A;

}

else {

//cout<<"Largest Number is "<<B<<endl;

return B;

}

}

{

if(C>B) {

//cout<<"Largest Number is "<<C<<endl;

return C;

}

else {

//cout<<"Largest Number is "<<B<<endl;

return B;

}

}

}

int greatest1(int A, int B, int C) {

if(A>B) {

if(A>C) {

// cout<<"Largest Number is "<<A<<endl;

return A;

}

else {

//cout<<"Largest Number is "<<B<<endl;

return B;

}

}

{

if(C>B) {

//cout<<"Largest Number is "<<C<<endl;

return C;

}

else {

//cout<<"Largest Number is "<<B<<endl;

return B;

}

}

}

int main() {

// Write C++ code here

/\*cout<<"Test Suite (randomly taken)"<<endl;

cout<<"a "<<"b "<<"Output "<<endl;

cout<<"10 "<<"10 "<<"20 "<<endl;

cout<<"10 "<<"-1 "<<"0 "<<endl;

cout<<"-1 "<<"-1 "<<"0 "<<endl;

cout<<"0 "<<"0 "<<"0 "<<endl;

int a,b;

// cout<<"Enter two numbers to get the sum"<<endl;

// cin>>a>>b;

cout<<"Before changing the code output of test suit is"<<endl;

vector<vector<int>> arr;

vector<int> a1 = {10, 10};

vector<int> a2 = {10, -1};

vector<int> a3 = {-1, -1};

vector<int> a4 = {0, 0};

arr.push\_back(a1);

arr.push\_back(a2);

arr.push\_back(a3);

arr.push\_back(a4);

cout<<"a "<<"b "<<"Output "<<endl;

for(int i = 0; i<arr.size(); i++) {

vector<int> temp = arr[i];

int S = sum(temp[0], temp[1]);

cout<<temp[0]<<" "<<temp[1]<<" "<<S<<endl;

}

cout<<"After changing the code output of test suit is"<<endl;

cout<<"a "<<"b "<<"Output "<<endl;

for(int i = 0; i<arr.size(); i++) {

vector<int> temp = arr[i];

int S = sum1(temp[0], temp[1]);

cout<<temp[0]<<" "<<temp[1]<<" "<<S<<endl;

}

\*/

vector<vector<int>> arr;

vector<int> a1 = {15, 1, 15};

vector<int> a2 = {15, 2, 15};

vector<int> a3 = {15, 15, 15};

vector<int> a4 = {15, 29, 15};

vector<int> a5 = {15, 30, 15};

vector<int> a6 = {1,15, 15};

vector<int> a7 = {2,15, 15 };

vector<int> a8 = {29, 15, 15};

vector<int> a9 = {30, 15, 15};

arr.push\_back(a1);

arr.push\_back(a2);

arr.push\_back(a3);

arr.push\_back(a4);

arr.push\_back(a5);

arr.push\_back(a6);

arr.push\_back(a7);

arr.push\_back(a8);

cout<<"Test Suite Randomly Generated"<<endl;

cout<<"a "<<"b "<<"c "<<"Output "<<endl;

for(int i =0; i<arr.size(); i++) {

vector<int> temp = arr[i];

int g = greatest(temp[0], temp[1], temp[2]);

//cout<<"a "<<"b "<<"c "<<"Output "<<endl;

cout<<temp[0]<<" "<<temp[1]<<" "<<temp[2]<<" "<<g<<endl;

}

cout<<"Before Changing the code output is"<<endl;

for(int i =0; i<arr.size(); i++) {

vector<int> temp = arr[i];

int g = greatest(temp[0], temp[1], temp[2]);

//cout<<"a "<<"b "<<"c "<<"Output "<<endl;

cout<<temp[0]<<" "<<temp[1]<<" "<<temp[2]<<" "<<g<<endl;

}

cout<<"After Changing the code output is"<<endl;

for(int i =0; i<arr.size(); i++) {

vector<int> temp = arr[i];

int g = greatest1(temp[0], temp[1], temp[2]);

//cout<<"a "<<"b "<<"c "<<"Output "<<endl;

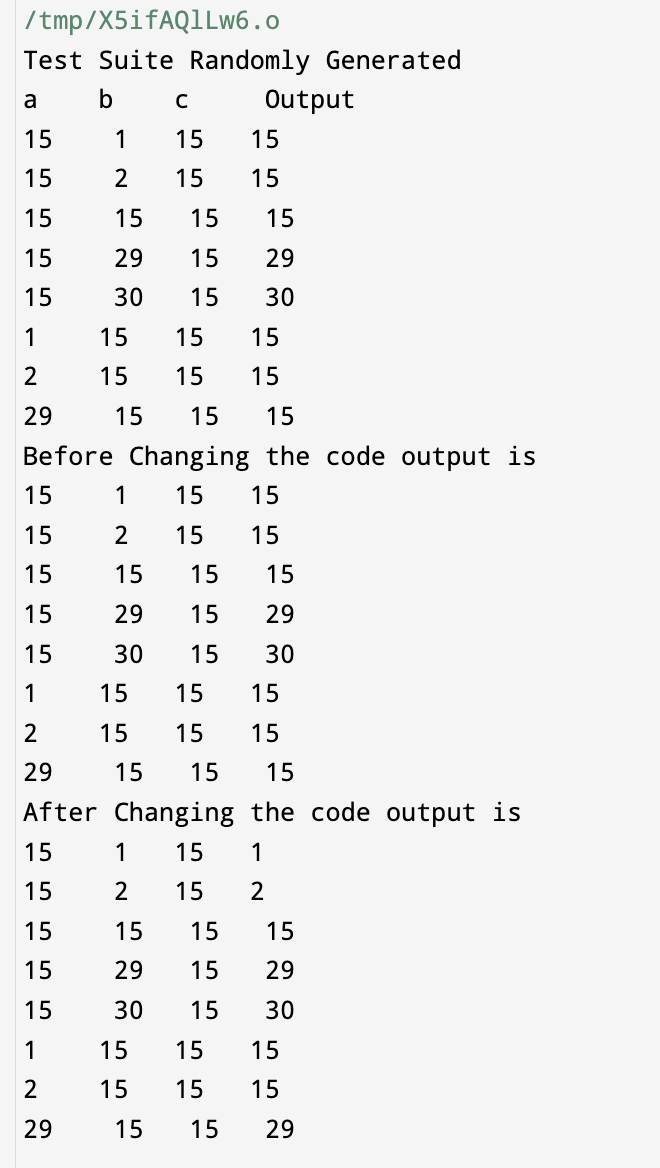
cout<<temp[0]<<" "<<temp[1]<<" "<<temp[2]<<" "<<g<<endl;

}

return 0;

}

**Output:**



**Experiment - 9**

**Aim:** Write a program to perform DD Path Testing.

**Code:**

#include<iostream>

#define MAX 10

using namespace std;

int main()

{

int row, col, i, j, edges=0, nodes=0;

char ch;

int matrix[MAX][MAX];

cout<<"Enter the number of rows: ";

cin>>row;

cout<<"Enter the number of columns: ";

cin>>col;

cout<<"\nEnter the adjacency matrix is.. \n";

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

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

cin>>matrix[i][j];

nodes = row;

cout<<"\nNumber of nodes = "<<nodes;

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

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

if(matrix[i][j]==1)

edges++;

cout<<"\nNumber of edges = "<<edges;

cout<<"\n\nCyclomatic Complexity of graph = edges-nodes+2p\n";

cout<<"\t\t\t\t= "<<edges<<" - "<<nodes<<" + 2";

cout<<"\n\t\t\t\t= "<<edges-nodes+2;

}

**Output:**

