Delhi Technological University

SOFTWARE TESTING

SE-302

Submitted By:

Parv Gupta

2K17/SE/79

Semester:6

Submitted To:

Ms. Pratima

DTU

INDEX:

S.No.	**Aim**
1.	Conduct a survey of different automated testing tools available.
2.	Write a program to find the maximum in three numbers input by the user and generate test cases for the program using Boundary Value Analysis.
3.	Write a program to find the maximum in three numbers input by the user and generate test cases for the program using Robust Approach.
4.	Write a program to find the maximum in three numbers input by the user and generate test cases for the program using Worst Boundary Value Analysis.
5.	Write a program to find the maximum in three numbers input by the user and generate test cases for the program using Worst Robust Approach.
6.	Write a program to find the type of the triangle on the basis of sides input by the user and generate test cases to test the program using Equivalence Class Testing.
7.	Write a program to find the type of the triangle on the basis of sides input by the user and generate test cases to test the program using Decision Table Testing.
8.	Write a program to find Cyclomatic complexity of a program.
9.	Write a program to input graph matrix and perform DD path testing.
10.	Write a program to perform Mutation Testing.

EXPERIMENT:1

AIM: Conduct a survey of (atleast 5) different automated testing tools available.

1.SELENIUM:

Selenium is a set of different software tools each with a different approach to supporting test automation. Most Selenium QA Engineers focus on the one or two tools that most meet the needs of their project, however learning all the tools will give you many different options for approaching different test automation problems. The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior. One of Selenium's key features is the support for executing one's tests on multiple browser platforms.

Selenium first came to life in 2004 when Jason Huggins was testing an internal application at ThoughtWorks. Being a smart guy, he realized there were better uses of his time than manually stepping through the same tests with every change he made. He developed a Javascript library that could drive interactions with the page, allowing him to automatically rerun tests against multiple browsers. That library eventually became Selenium Core, which underlies all the functionality of Selenium Remote Control (RC) and Selenium IDE. Selenium RC was ground-breaking because no other product allowed you to control a browser from a language of your choice.

Advantages:

1. Selenium is pure open source, freeware and portable tool.

2. Selenium supports variety of languages that include Java, Perl, Python, C#, Ruby, Groovy, Java Script, and VB Script. etc.

3. Selenium supports many operating systems like Windows, Macintosh, Linux, Unix etc.

4. Selenium supports many browsers like Internet explorer, Chrome, Firefox, Opera, Safari etc.

5. Selenium can be integrated with ANT or Maven kind of framework for source code compilation.

6. Selenium can be integrated with TestNG testing framework for testing our applications and generating reports.

7. Selenium can be integrated with Jenkins or Hudson for continuous integration.

Disadvantages:

1.Selenium needs very much expertise resources. The resource should also be very well versed in framework architecture.

2.Selenium only supports web based application and does not support windows based application.

3.It is difficult to test Image based application.

4.Selenium need outside support for report generation activity like dependence on TestNG or Jenkins.

5.Selenium does not support built in add-ins support.

6.Selenium user lacks online support for the problems they face.

7.Selenium does not provide any built in IDE for script generation and it need other IDE like Eclipse for writing scripts.

2. **WATIR:** Watir (Web Application Testing in Ruby, pronounced water), is an open-source (BSD) family of Ruby libraries for automating web browsers. It drives Firefox, Chrome, Opera and Safari, and is available as a RubyGems gem. Watir was primarily developed by Bret Pettichord and Paul Rogers.

Pros:

1. No external server needed to run a test. Just run "ruby test.rb" and it goes, this simplifies any scripts you use to execute your tests through continuous integration servers like Hudson or Bamboo or whatever.

2. GREAT API! this is the kind of API rubyists dream of... its easy and fun and once you get the basics you can just guess how to write the code without needed alot of documentation.

Tests can often be run through alternative tools... watir-webdriver, webrat, celerity and more all more-or-less support the watir API.

Cons:

1.Each browser's implementation is a little bit different, and the driver for each browser is written independently. This can mean inconsistent test results between browsers, but it hasnt really been much an issue for me.

2. API is ruby only (but if you use ruby, this is actually an advantage because you get a truly ruby-esque tool)

3. Doesnt seem to be as widely used as selenium, a lil bit harder to find engineers (finding good ones is pretty much impossible).

3.Quick Test Professional:

HP Unified Functional Testing (UFT) software, formerly known as HP QuickTest Professional (QTP), provides functionaland regression test automation for software applications and environments. HP Unified Functional Testing can be used for enterprise quality assurance.

HP Unified Functional Testing supports keyword and scripting interfaces and features a graphical user interface. It uses the Visual Basic Scripting Edition (VBScript) scripting language to specify a test procedure, and to manipulate the objects and controls of the application under test.

HP Unified Functional Testing was originally written by Mercury Interactive and called QuickTest Professional. Mercury Interactive was subsequently acquired by Hewlett Packard(HP) in 2006. HP Unified Functional Testing 11.5 combined HP QuickTest Professional and HP Service Test into a single software package, which is currently available from the HP Software Division. The integrated HP Unified Functional Testing software allows developers to test from a single console all three layers of a program's operations: the interface, the service layer and the database layer.

Advantages:

1. QTP has record and playback facility with editing the scripts generated after the recording is completed. It supports different recording mode i.e. Normal, Analog and Low level which facilitates to automate different types of applications.

2. QTP supports almost all popular automation frameworks like Linear, Keyword, Data Driven, Hybrid etc. for automation purpose.

3. QTP is very much user friendly and anyone can start using it very less expertise.

4. QTP has built in IDE associated which is very much easy to use.

5. Its primary scripting language is VB script, thus it does need skilled code to start automation work. Also this tool is packed with many add-ins which supports it very much.

6. QTP can be integrated with one of the most useful test management tool i.e. Quality Center (QC). This integration very much supports the testing activity.

Disadvantages:

1. QTP is used with license and the license cost of using it is very high.

2. For getting online support we need to keep on the license renewal.

3. QTP supports many add-ins, but for using it we need to buy it.

4. QTP takes loads of RAM and CPU and comparatively slow in execution.

5. QTP supports limited version of browsers for script execution

6. QTP does not support multithreading i.e. running multiple instance of script in different browsers at same time.

4.WebLOAD:

WebLOAD is load testing tool, performance testing, stress test web applications. This web and mobile load testing and analysis tool is from RadView Software. Load testing tool WebLOAD combines performance, scalability, and integrity as a single process for the verification of web and mobile applications. It can simulate hundreds of thousands of concurrent users making it possible to test large loads and report bottlenecks, constraints, and weak points within an application.

Using its multi-protocol support, WebLOAD simulates traffic of hundreds of thousands of users and provides analysis data on how the application behaves under load. WebLOAD monitors and incorporates statistics from the various components of the system under test: servers, application server, database, network, load-balancer, firewall, etc., and can also monitor the End User Experience and Service Level Agreement (SLA) compliance in production environments.

WebLOAD's features include:

1. IDE An integrated development environment for visually recording, editing & debugging load test scripts. WebLOAD's proxy-based recorder records HTTP activity. Test are generated in JavaScript and can be enhanced and edited using various tools in the IDE.

2. Correlation Automatic correlation of dynamic values such as Session IDs, enables a script to be executed dynamically with multiple virtual clients.

3. Load Generation WebLOAD generates load from on-premises machines or from the cloud.

4. Analytics A set of predefined analysis reports provides performance data, helping users identify bottlenecks. Reports and analysis data can also be viewed remotely via customizable Web Dashboard.

5. PMM Collects server-side statistics during test runs, providing users with additional data for root-cause analysis.

6. Web Dashboard Analyzing performance test results from any browser or Mobile device.

Advantages:

1. Uses Java script for script programming

2. Allows for session handling, outside of cookies

3. Supports Ajax, SOAP and Web services

4. Has a Forum for Support related question

5. Allows for automated script recording

□

Disadvantages:

1. Some developers complain that WebLoad is not an open source in true sense. Contrary to what their press release and website says, it contains proprietary components that are released in binary form with no source code.

5.FITNESSE: FitNesse is a web server, a wiki and an automated testing tool for software. It is based on Ward Cunningham's Framework for Integrated Test and is designed to support acceptance testing rather than unit testing in that it facilitates detailed readable description of system function. FitNesse allows users of a developed system to enter specially formatted input (its format is accessible to non-programmers). This input is interpreted and tests are created automatically. These tests are then executed by the system and output is returned to the user. The advantage of this approach is very fast feedback from users. The developer of the system to be tested needs to provide some support (classes named "fixtures", conforming to certain conventions).

FitNesse is written in Java (by Robert C. Martin and others). The program first supported only Java, but versions for several other languages have been added over time (C++, Python, Ruby, Delphi, C#, etc.).

Advantages:

1. FitNesse tests can give us feature feedback very early in the project. In fact, the tests ought to be written first, so programmers can code to the tests.

2. FitNesse tests can give us feature feedback very frequently. They can be run manually or automatically by anyone with web access to the server, as frequently as required. Every week, every day, every hour in a crunch.

3. FitNesse tests are deterministic: they either run green or red. If they run green for a given requirement, either the requirement is done and we move on to others, or the set of tests is not yet exactly right, in which case we refine them. Either way, we are successively refining the system in an orderly way. With each new test running green, we can all see the system getting better, more valuable, closer to what we need.

4. Being based on example data, FitNesse tests exercise more paths through the business logic. When you use FitNesse, you run less risk of missing important features or functional behavior.

Disadvantages:

1.No capture & replay possible

2.Cannot explicitly test a web front end.

3.Web interface can make setting up expectations time consuming.

4.Learning curve to understand the framework is steeper if we have to modify the framework.

EXPERIMENT:2

AIM: Write a program to find the maximum in three numbers input by the user and generate test cases for the program using Boundary Value Analysis.

THEORY: Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values. So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called "boundary testing".

The basic idea in boundary value testing is to select input variable values at their:

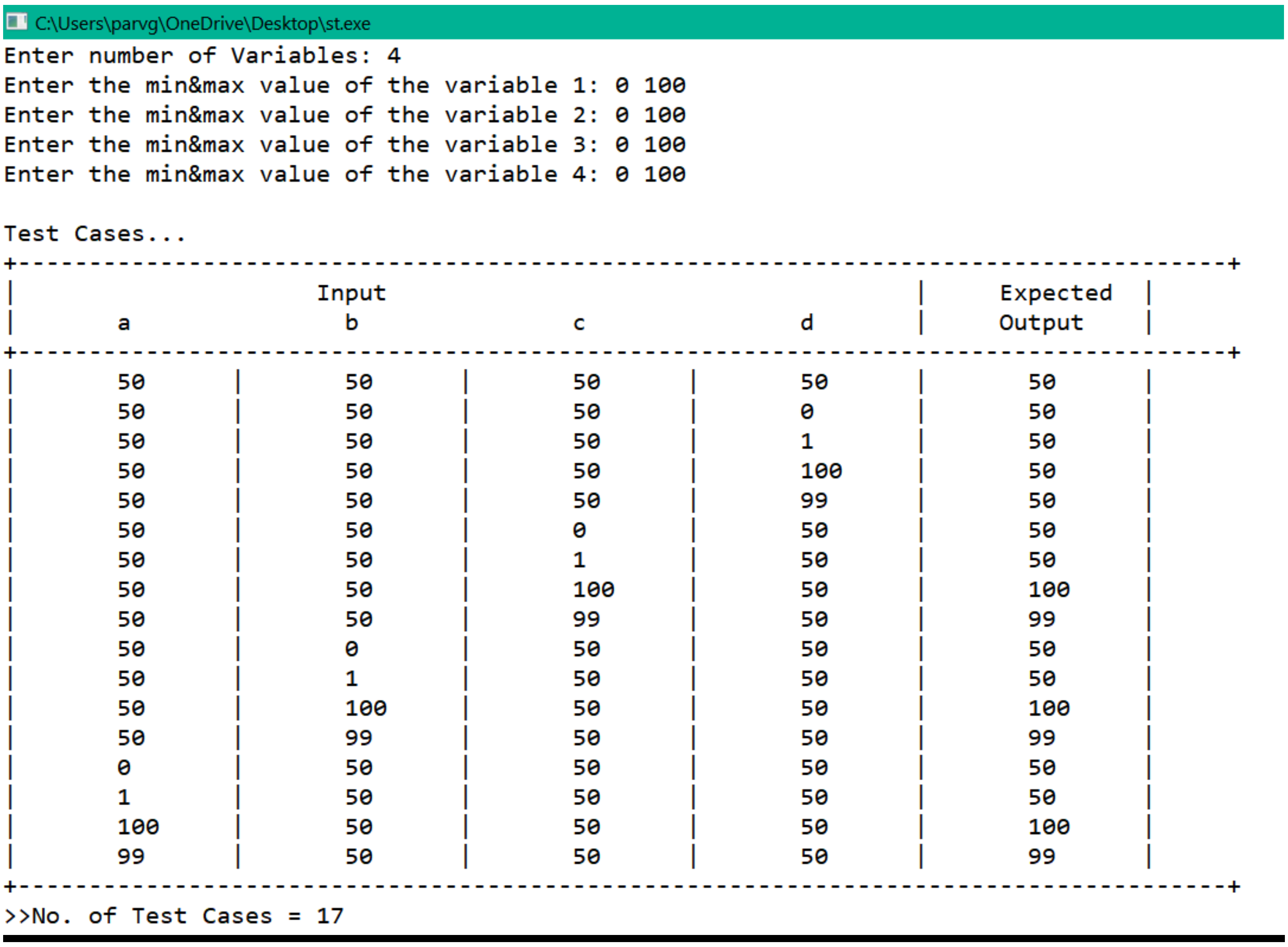
1. Minimum
2. Just above the minimum
3. A nominal value
4. Just below the maximum
5. Maximum

CODE:

```
#include<iostream>
using namespace std;
int findMaximum(int x[3], int min[3], int max[3]){
   for(int i=0;i<3;i++)
          if(x[i]<min[i] || x[i]>max[i])
               return -1;
     int largest = x[0];

     if(largest<x[1])
          largest = x[1];
     else if(largest<x[2])
          largest = x[2];

     return largest;
}
```

```
void Boundaryvalanalysis(int n)
{
    int min[10], max[10], option[10];

    for(int i=0;i<n;i++)
    {
        cout<<"Enter the min&max value of the variable "<<(i+1)<<": ";
        cin>>min[i]>>max[i];
        option[i] = (max[i]+min[i])/2;
    }

    int testCases[4*n+1][n];
    for(int i=0;i<n;i++)
        testCases[0][i] = option[i];

    for(int i=0;i<n;i++)
    {
        int flag=0;
        for(int j=1;j<=4*n;j++)
        {
            if(j > ((4*n)-((i+1)*4)) && j <= ((4*n)-(i*4)))
            {
                flag++;
                switch(flag)
                {
                    case 1: testCases[j][i] = min[i];
                        break;
                    case 2: testCases[j][i] = min[i]+1;
                        break;
                    case 3: testCases[j][i] = max[i];
                        break;
                    case 4: testCases[j][i] = max[i]-1;
                        break;
                }
                }
                else
                        testCases[j][i] = option[i];
        }
    }

    cout<<"\nTest Cases...";
    cout<<"\n+";    for(int i=0;i<85;i++)  cout<<"-";   cout<<"+";
    cout<<"\n|\t\t     Input\t\t\t\t|     Expected\t|";
    cout<<"\n|\ta\t\tb\t\tc\t\td\t|     Output\t|";
    cout<<"\n+";    for(int i=0;i<85;i++)  cout<<"-";   cout<<"+\n";
    for(int i=0;i<4*n+1;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(j==0)        cout<<"|\t";
            cout<<testCases[i][j]<<"\t|\t";
        }
        int expOut = findMaximum(testCases[i],min,max);
        if(expOut<0)
             cout<<"--";
        else cout<<expOut;
        cout<<"\t|"<<endl;
    }
    cout<<"+";  for(int i=0;i<85;i++)  cout<<"-";   cout<<"+";
    cout<<"\n>>No. of Test Cases = "<<4*n+1;
}

int main()
{
    int numofvar;
    cout<<"Enter number of Variables: ";
    cin>>numofvar;
    Boundaryvalanalysis(numofvar);

    return 0;

}
```

OUTPUT:



C:\Users\parvg\OneDrive\Desktop\st.exe

```
Enter number of Variables: 4
Enter the min&max value of the variable 1: 0 100
Enter the min&max value of the variable 2: 0 100
Enter the min&max value of the variable 3: 0 100
Enter the min&max value of the variable 4: 0 100

Test Cases...
+---------------------------------------------------------------------------------------+
|                      Input                                     |     Expected    |
|       a               b               c               d        |     Output      |
+---------------------------------------------------------------------------------------+
|       50      |       50      |       50      |       50      |       50      |
|       50      |       50      |       50      |       0       |       50      |
|       50      |       50      |       50      |       1       |       50      |
|       50      |       50      |       50      |       100     |       50      |
|       50      |       50      |       50      |       99      |       50      |
|       50      |       50      |       0       |       50      |       50      |
|       50      |       50      |       1       |       50      |       50      |
|       50      |       50      |       100     |       50      |       100     |
|       50      |       50      |       99      |       50      |       99      |
|       50      |       0       |       50      |       50      |       50      |
|       50      |       1       |       50      |       50      |       50      |
|       50      |       100     |       50      |       50      |       100     |
|       50      |       99      |       50      |       50      |       99      |
|       0       |       50      |       50      |       50      |       50      |
|       1       |       50      |       50      |       50      |       50      |
|       100     |       50      |       50      |       50      |       100     |
|       99      |       50      |       50      |       50      |       99      |
+---------------------------------------------------------------------------------------+
>>No. of Test Cases = 17
```

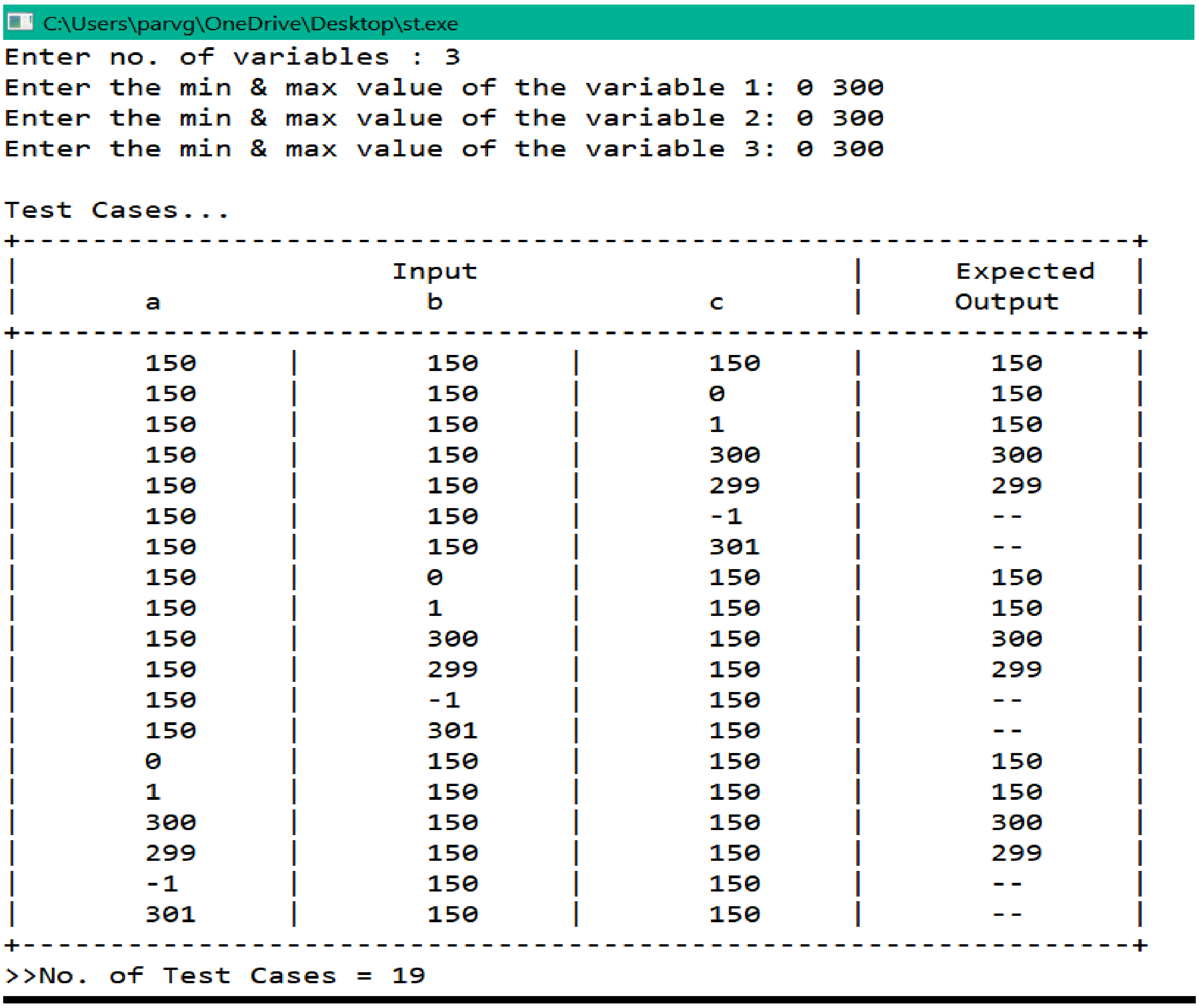
EXPERIMENT:3

AIM: Write a program to find the maximum in three numbers input by the user and generate test cases for the program using Robust Approach.

CODE:

```
#include<iostream>
using namespace std;
int findmaximum(int arr[3], int minarr[3], int maxarr[3])
{
        for(int i=0;i<3;i++)
                if(arr[i]<minarr[i] || arr[i]>maxarr[i])
                        return -1;

        int largest = arr[0];

        if(largest<arr[1])
                largest = arr[1];
        else if(largest<arr[2])
                largest = arr[2];

        return largest;
}

void robustnesstesting(int n)
{
        int minarr[10], maxarr[10], opt[10];

        for(int i=0;i<n;i++)
        {
                cout<<"Enter the min & max value of the variable "<<(i+1)<<": ";
                cin>>minarr[i]>>maxarr[i];
        opt[i] = (maxarr[i]+minarr[i])/2;
}

int testCases[6*n+1][n];
for(int i=0;i<n;i++)
        testCases[0][i] = opt[i];

for(int i=0;i<n;i++)
{
        int flag=0;
        for(int j=1;j<=6*n;j++)
        {
                if(j > ((6*n)-((i+1)*6)) && j <= ((6*n)-(i*6)))
                {
                        flag++;
                        switch(flag)
                        {
                                case 1: testCases[j][i] = minarr[i];
                                        break;
                                case 2: testCases[j][i] = minarr[i]+1;
                                        break;
                                case 3: testCases[j][i] = maxarr[i];
                                        break;
                                case 4: testCases[j][i] = maxarr[i] - 1;
                                        break;
                                case 5: testCases[j][i] = minarr[i] - 1;
                                        break;
                                case 6: testCases[j][i] = maxarr[i] + 1;
                                        break;
                        }
                }
                else
                        testCases[j][i] = opt[i];
        }
        }

        cout<<"\nTest Cases...";
        cout<<"\n+";       for(int i=0;i<63;i++)  cout<<"-";    cout<<"+";
        cout<<"\n|\t\t      Input\t\t\t|     Expected\t|";
        cout<<"\n|\ta\t\tb\t\tc\t|     Output\t|";
        cout<<"\n+";       for(int i=0;i<63;i++)  cout<<"-";    cout<<"+\n";
        for(int i=0;i<6*n+1;i++)
        {
                for(int j=0;j<n;j++)
                {
                        if(j==0)        cout<<"|\t";
                        cout<<testCases[i][j]<<"\t|\t";
                }
                int expOut = findmaximum(testCases[i],minarr,maxarr);
                if(expOut<0)
                         cout<<"--";
                else        cout<<expOut;
                cout<<"\t|"<<endl;
        }
        cout<<"+";   for(int i=0;i<63;i++)  cout<<"-";    cout<<"+";
        cout<<"\n>>No. of Test Cases = "<<6*n+1;
}

int main()
{
        int numofvar;
        cout<<"Enter no. of variables : ";
        cin>>numofvar;
    robustnesstesting(numofvar);
    return 0;
}
```

OUTPUT:


C:\Users\parvg\OneDrive\Desktop\st.exe

```
Enter no. of variables : 3
Enter the min & max value of the variable 1: 0 300
Enter the min & max value of the variable 2: 0 300
Enter the min & max value of the variable 3: 0 300

Test Cases...
+----------------------------------------------------------------+
|                      Input                       |  Expected  |
|       a                b                c         |   Output   |
+----------------------------------------------------------------+
|       150      |       150      |      150       |    150     |
|       150      |       150      |      0         |    150     |
|       150      |       150      |      1         |    150     |
|       150      |       150      |      300       |    300     |
|       150      |       150      |      299       |    299     |
|       150      |       150      |      -1        |    --      |
|       150      |       150      |      301       |    --      |
|       150      |       0        |      150       |    150     |
|       150      |       1        |      150       |    150     |
|       150      |       300      |      150       |    300     |
|       150      |       299      |      150       |    299     |
|       150      |       -1       |      150       |    --      |
|       150      |       301      |      150       |    --      |
|       0        |       150      |      150       |    150     |
|       1        |       150      |      150       |    150     |
|       300      |       150      |      150       |    300     |
|       299      |       150      |      150       |    299     |
|       -1       |       150      |      150       |    --      |
|       301      |       150      |      150       |    --      |
+----------------------------------------------------------------+
>>No. of Test Cases = 19
```

EXPERIMENT:4

AIM: Write a program to find the maximum in three numbers input by the user and generate test cases for the program using Worst Boundary Value Analysis.

THEORY: This is a special form of boundary value analysis where we don’t consider the ‘single fault’ assumption theory of reliability. Now, failures are also due to occurrence of more than one fault simultaneously. The implication of this concept in boundary value analysis is that all input values may have one of the following:

1. Minimum value
2. Just above minimum value
3. Just below maximum value
4. Maximum value
5. Nominal (Average) value

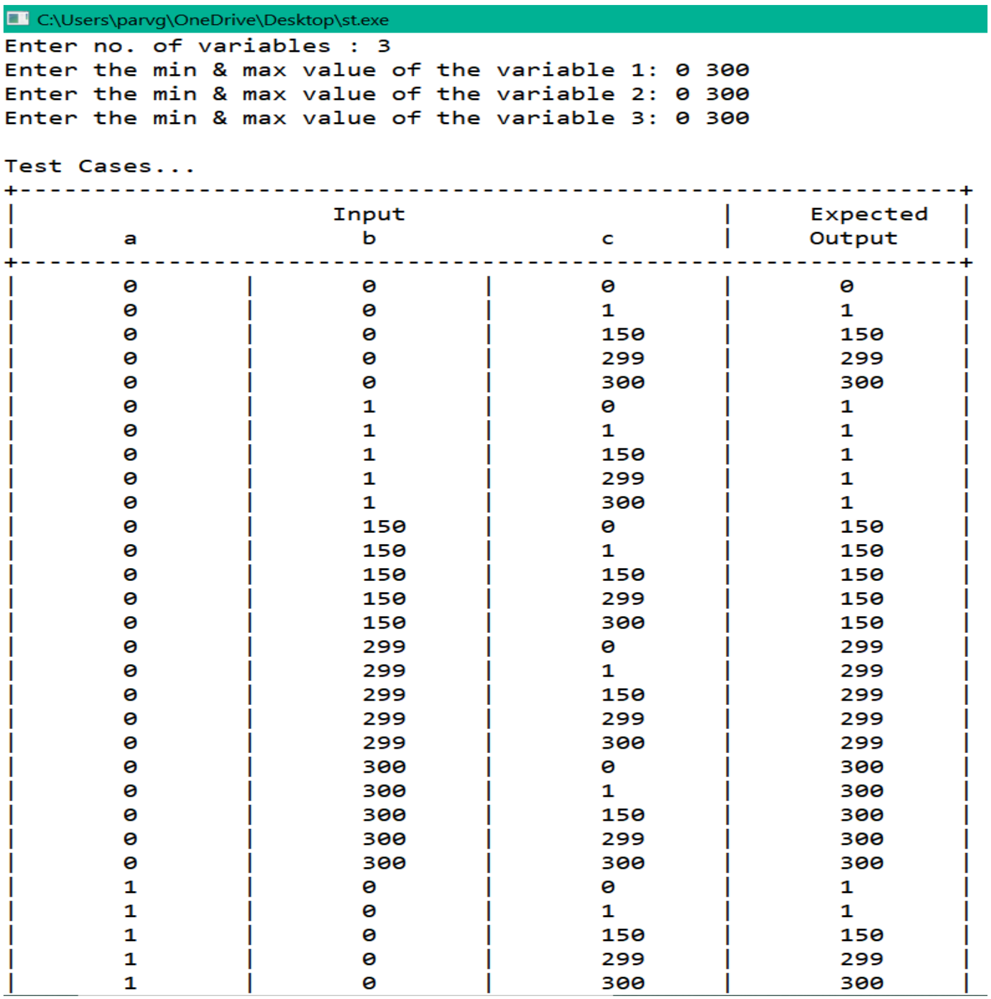
CODE:

```
#include<iostream>

#include<math.h>

using namespace std;

int findmaximum(int arr[3], int minarr[3], int maxarr[3]){

    for(int i=0;i<3;i++)

        if(arr[i]<minarr[i] || arr[i]>maxarr[i])

            return -1;


    int largest = arr[0];


    if(largest<arr[1])

        largest = arr[1];

    else if(largest<arr[2])

        largest = arr[2];


    return largest;

}


void worstboundaryvalueanalysis(int n){

    int minarr[10], maxarr[10], opt[10];
```

```
for(int i=0;i<n;i++){
    cout<<"Enter the min & max value of the variable "<<(i+1)<<": ";
    cin>>minarr[i]>>maxarr[i];
    opt[i] = (maxarr[i]+minarr[i])/2;
}

int numTestCases = pow(5,n);
int testCases[numTestCases][n];

for(int i=n-1;i>=0;i--){
    int flag=0, t=1, k;
    for(int k=0;k<(n-i-1);k++)
        t*=5;
    for(int j=0;j<numTestCases;j++)
    {
        if(j%t==0)    flag++;
        flag = flag%5;
        switch(flag)
        {
            case 0:   testCases[j][i] = maxarr[i];
                break;
            case 1: testCases[j][i] = minarr[i];
                break;
            case 2: testCases[j][i] = minarr[i]+1;
                break;
            case 3: testCases[j][i] = opt[i];
                break;
            case 4: testCases[j][i] = maxarr[i]-1;
                break;
        }
    }
}

cout<<"\nTest Cases...";
```

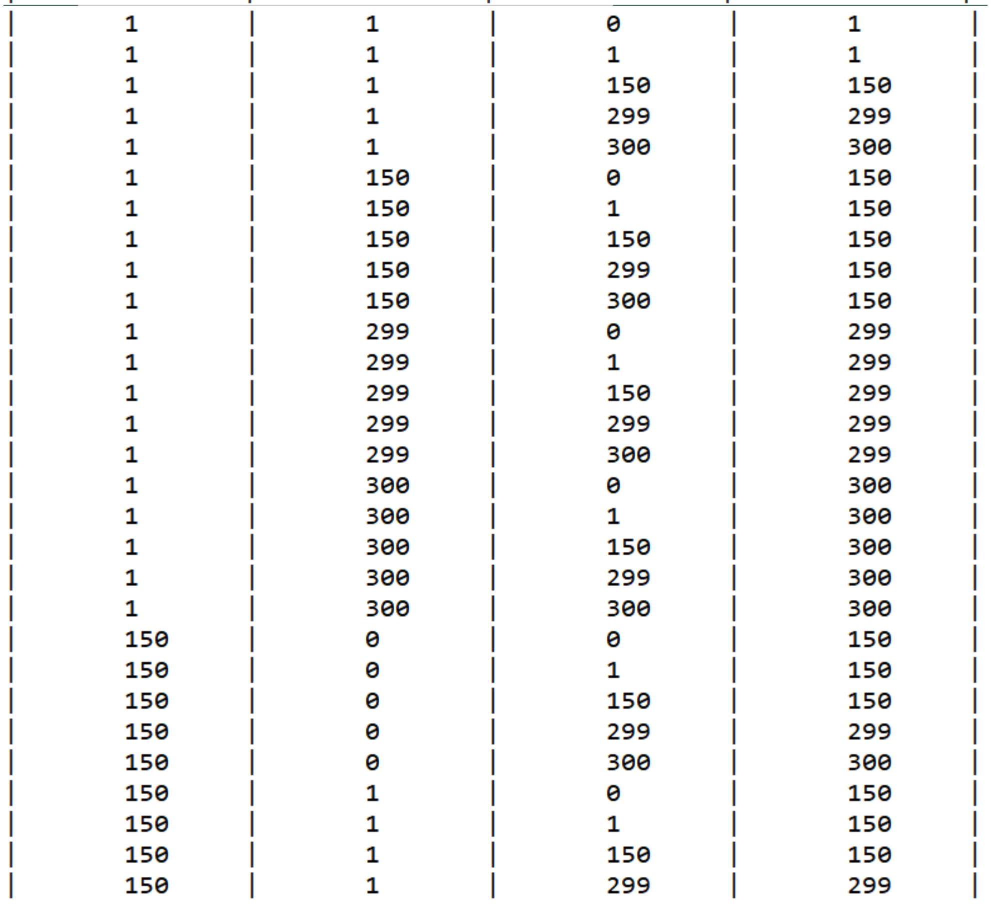
```
	cout<<"\n+";	for(int i=0;i<63;i++)  cout<<"-";    cout<<"+";
	cout<<"\n|\t\t		Input\t\t\t|		Expected\t|";
	cout<<"\n|\ta\t\tb\t\tc\t|		Output\t|";
	cout<<"\n+";	for(int i=0;i<63;i++)  cout<<"-";    cout<<"+\n";
	for(int i=0;i<numTestCases;i++)
	{
		for(int j=0;j<n;j++)
		{
			if(j==0)	cout<<"|\t";
			cout<<testCases[i][j]<<"\t|\t";
		}
		int expOut = findmaximum(testCases[i],minarr,maxarr);
		if(expOut<0)
			  cout<<"--";
		else	cout<<expOut;
		cout<<"\t|"<<endl;
	}
	cout<<"+";   for(int i=0;i<63;i++)  cout<<"-";    cout<<"+";
	cout<<"\n>>No. of Test Cases = "<<numTestCases;
}

int main()
{
	int numofvar;
	cout<<"Enter no. of variables : ";
	cin>>numofvar;
	 worstboundaryvalueanalysis(numofvar);
	 return 0;
}
```

OUTPUT:



C:\Users\parvg\OneDrive\Desktop\st.exe

```
Enter no. of variables : 3
Enter the min & max value of the variable 1: 0 300
Enter the min & max value of the variable 2: 0 300
Enter the min & max value of the variable 3: 0 300

Test Cases...
+-----------------------------------------------------------------+
|                         Input                     |  Expected   |
|       a                   b                c      |  Output     |
+-----------------------------------------------------------------+
|       0       |       0       |       0       |       0       |
|       0       |       0       |       1       |       1       |
|       0       |       0       |       150     |       150     |
|       0       |       0       |       299     |       299     |
|       0       |       0       |       300     |       300     |
|       0       |       1       |       0       |       1       |
|       0       |       1       |       1       |       1       |
|       0       |       1       |       150     |       1       |
|       0       |       1       |       299     |       1       |
|       0       |       1       |       300     |       1       |
|       0       |       150     |       0       |       150     |
|       0       |       150     |       1       |       150     |
|       0       |       150     |       150     |       150     |
|       0       |       150     |       299     |       150     |
|       0       |       150     |       300     |       150     |
|       0       |       299     |       0       |       299     |
|       0       |       299     |       1       |       299     |
|       0       |       299     |       150     |       299     |
|       0       |       299     |       299     |       299     |
|       0       |       299     |       300     |       299     |
|       0       |       300     |       0       |       300     |
|       0       |       300     |       1       |       300     |
|       0       |       300     |       150     |       300     |
|       0       |       300     |       299     |       300     |
|       0       |       300     |       300     |       300     |
|       1       |       0       |       0       |       1       |
|       1       |       0       |       1       |       1       |
|       1       |       0       |       150     |       150     |
|       1       |       0       |       299     |       299     |
|       1       |       0       |       300     |       300     |
```



```
|       1       |       1       |       0       |       1       |
|       1       |       1       |       1       |       1       |
|       1       |       1       |       150     |       150     |
|       1       |       1       |       299     |       299     |
|       1       |       1       |       300     |       300     |
|       1       |       150     |       0       |       150     |
|       1       |       150     |       1       |       150     |
|       1       |       150     |       150     |       150     |
|       1       |       150     |       299     |       150     |
|       1       |       150     |       300     |       150     |
|       1       |       299     |       0       |       299     |
|       1       |       299     |       1       |       299     |
|       1       |       299     |       150     |       299     |
|       1       |       299     |       299     |       299     |
|       1       |       299     |       300     |       299     |
|       1       |       300     |       0       |       300     |
|       1       |       300     |       1       |       300     |
|       1       |       300     |       150     |       300     |
|       1       |       300     |       299     |       300     |
|       1       |       300     |       300     |       300     |
|       150     |       0       |       0       |       150     |
|       150     |       0       |       1       |       150     |
|       150     |       0       |       150     |       150     |
|       150     |       0       |       299     |       299     |
|       150     |       0       |       300     |       300     |
|       150     |       1       |       0       |       150     |
|       150     |       1       |       1       |       150     |
|       150     |       1       |       150     |       150     |
|       150     |       1       |       299     |       299     |
|       150       |       1         |       300       |       300       |
|       150       |       150       |       0         |       150       |
|       150       |       150       |       1         |       150       |
|       150       |       150       |       150       |       150       |
|       150       |       150       |       299       |       299       |
|       150       |       150       |       300       |       300       |
|       150       |       299       |       0         |       299       |
|       150       |       299       |       1         |       299       |
|       150       |       299       |       150       |       299       |
|       150       |       299       |       299       |       299       |
|       150       |       299       |       300       |       299       |
|       150       |       300       |       0         |       300       |
|       150       |       300       |       1         |       300       |
|       150       |       300       |       150       |       300       |
|       150       |       300       |       299       |       300       |
|       150       |       300       |       300       |       300       |
|       299       |       0         |       0         |       299       |
|       299       |       0         |       1         |       299       |
|       299       |       0         |       150       |       299       |
|       299       |       0         |       299       |       299       |
|       299       |       0         |       300       |       300       |
|       299       |       1         |       0         |       299       |
|       299       |       1         |       1         |       299       |
|       299       |       1         |       150       |       299       |
|       299       |       1         |       299       |       299       |
|       299       |       1         |       300       |       300       |
|       299       |       150       |       0         |       299       |
|       299       |       150       |       1         |       299       |
|       299       |       150       |       150       |       299       |
|       299       |       150       |       299       |       299       |
|       299       |       150       |       300       |       300       |
|       299       |       299       |       0         |       299       |
|       299       |       299       |       1         |       299       |
|       299       |       299       |       150       |       299       |
|       299       |       299       |       299       |       299       |
|       299       |       299       |       300       |       300       |
|       299       |       300       |       0         |       300       |
|       299       |       300       |       1         |       300       |
|       299       |       300       |       150       |       300       |
|       299       |       300       |       299       |       300       |
|       299       |       300       |       300       |       300       |
|       300       |       0         |       0         |       300       |
|       300       |       0         |       1         |       300       |
|       300       |       0         |       150       |       300       |
|       300       |       0         |       299       |       300       |
|       300       |       0         |       300       |       300       |
|       300       |       1         |       0         |       300       |
|       300       |       1         |       1         |       300       |
|       300       |       1         |       150       |       300       |
|       300       |       1         |       299       |       300       |
|       300       |       1         |       300       |       300       |
|       300       |       150       |       0         |       300       |
|       300       |       150       |       1         |       300       |
|       300       |       150       |       150       |       300       |
|       300       |       150       |       299       |       300       |
|       300       |       150       |       300       |       300       |
|       300       |       299       |       0         |       300       |
|       300       |       299       |       1         |       300       |
|       300       |       299       |       150       |       300       |
|       300       |       299       |       299       |       300       |
|       300       |       299       |       300       |       300       |
|       300       |       300       |       0         |       300       |
|       300       |       300       |       1         |       300       |
|       300       |       300       |       150       |       300       |
|       300       |       300       |       299       |       300       |
+-----------------------------------------------------------------------+
>>No. of Test Cases = 124
```

EXPERIMENT:5

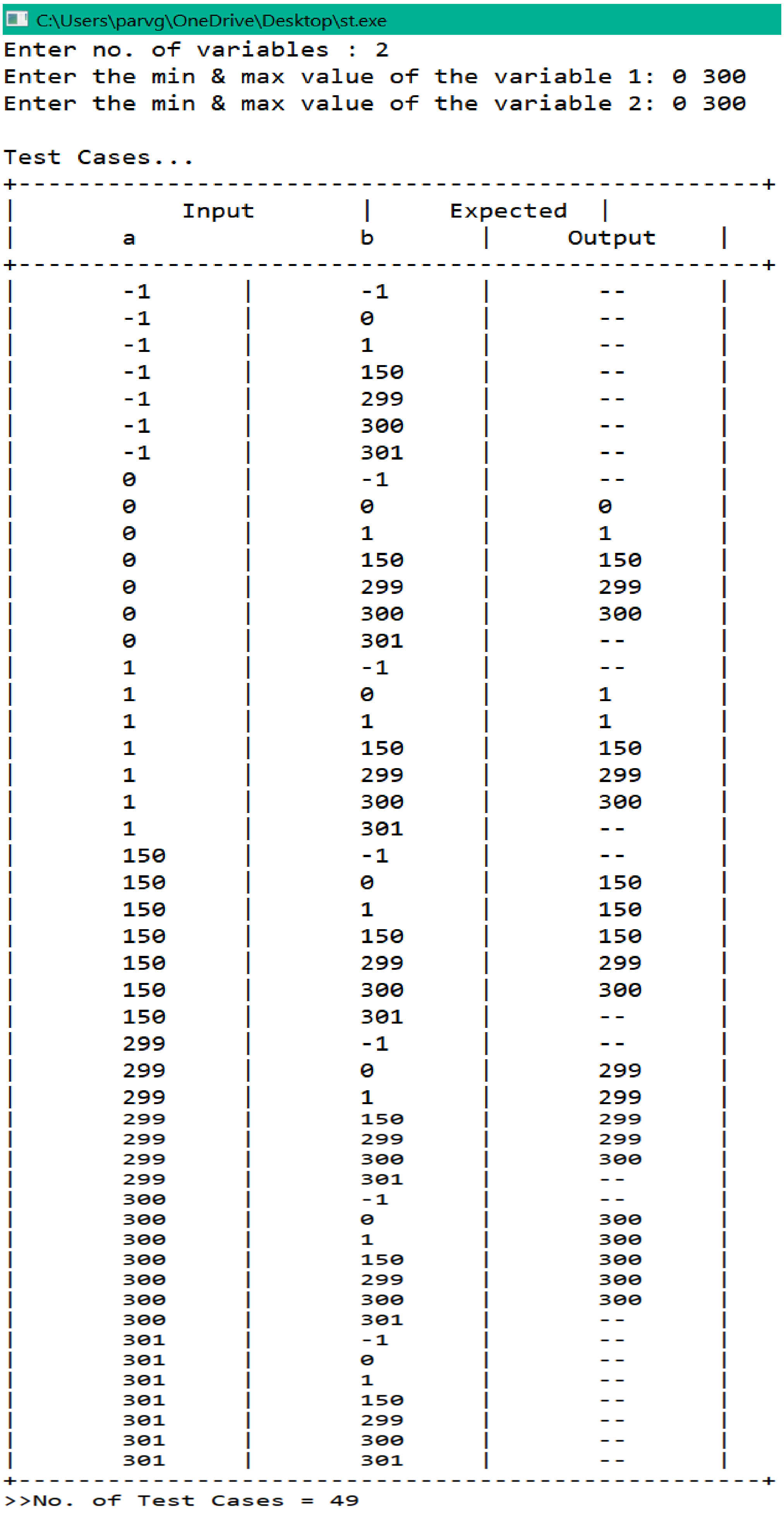
AIM: Write a program to find the maximum in three numbers input by the user and generate test cases for the program using Worst Robust Approach.

THEORY: In robustness testing, we add two more states i.e. just below minimum value (minimum value–)and just above maximum value (maximum value+). We also give invalid inputs and observe the behaviour of the program. A program should be able to handle invalid input values, otherwise it may fail and give unexpected output values. There are seven states (minimum -,minimum, minimum +, nominal, maximum –, maximum, maximum +) and a total of 7^n test cases will be generated. This will be the largest set of test cases and requires the maximum effort to generate such test Cases.

CODE:

```
#include<iostream>
#include<math.h>
using namespace std;
int findmaximum(int arr[3], int minarr[3], int maxarr[3]){
    for(int i=0;i<3;i++)
        if(arr[i]<minarr[i] || arr[i]>maxarr[i])
            return -1;

    int largest = arr[0];

    if(largest<arr[1])
        largest = arr[1];
    else if(largest<arr[2] )
        largest = arr[2];

    return largest;
}

void WorstRobustApproach(int n){
    int minarr[10], maxarr[10], opt[10];

    for(int i=0;i<n;i++)
    {
        cout<<"Enter the min & max value of the variable "<<(i+1)<<": ";
        cin>>minarr[i]>>maxarr[i];
        opt[i] = (maxarr[i]+minarr[i])/2;
}

int numTestCases = pow(7,n);
int testCases[numTestCases][n];

for(int i=n-1;i>=0;i--)
{
        int flag=0, t=1, k;
        for(int k=0;k<(n-i-1);k++)
                t*=7;
        for(int j=0;j<numTestCases;j++)
        {
                if(j%t==0)    flag++;
                flag = flag%7;
                switch(flag)
                {
                        case 0:   testCases[j][i] = maxarr[i]+1;
                                break;
                        case 1: testCases[j][i] = minarr[i]-1;
                                break;
                        case 2: testCases[j][i] = minarr[i];
                                break;
                        case 3: testCases[j][i] = minarr[i]+1;
                                break;
                        case 4: testCases[j][i] = opt[i];
                                break;
                        case 5: testCases[j][i] = maxarr[i]-1;
                                        break;
                      case 6: testCases[j][i] = maxarr[i];
                                                        break;
                }
        }
    }

    cout<<"\nTest Cases...";
    cout<<"\n+";    for(int i=0;i<50;i++)  cout<<"-";   cout<<"+";
    cout<<"\n|\t      Input\t|        Expected\t|";
    cout<<"\n|\ta\t\tb\t|        Output\t|";
    cout<<"\n+";    for(int i=0;i<50;i++)  cout<<"-";   cout<<"+\n";
    for(int i=0;i<numTestCases;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(j==0)      cout<<"|\t";
            cout<<testCases[i][j]<<"\t|\t";
        }
        int expOut = findmaximum(testCases[i],minarr,maxarr);
        if(expOut<0)
             cout<<"--";
        else      cout<<expOut;
        cout<<"\t|"<<endl;
    }
    cout<<"+";  for(int i=0;i<50;i++)  cout<<"-";   cout<<"+";
    cout<<"\n>>No. of Test Cases = "<<numTestCases;
}

int main(){
    int numofvar;
    cout<<"Enter no. of variables : ";
    cin>>numofvar;
     WorstRobustApproach(numofvar);
    return 0;
}
```

OUTPUT:



C:\Users\parvg\OneDrive\Desktop\st.exe

```
Enter no. of variables : 2
Enter the min & max value of the variable 1: 0 300
Enter the min & max value of the variable 2: 0 300

Test Cases...
+-----------------------------------------------------+
|            Input         |       Expected    |
|        a                b        |      Output      |
+-----------------------------------------------------+
|        -1        |      -1       |      --       |
|        -1        |      0        |      --       |
|        -1        |      1        |      --       |
|        -1        |      150      |      --       |
|        -1        |      299      |      --       |
|        -1        |      300      |      --       |
|        -1        |      301      |      --       |
|        0         |      -1       |      --       |
|        0         |      0        |      0        |
|        0         |      1        |      1        |
|        0         |      150      |      150      |
|        0         |      299      |      299      |
|        0         |      300      |      300      |
|        0         |      301      |      --       |
|        1         |      -1       |      --       |
|        1         |      0        |      1        |
|        1         |      1        |      1        |
|        1         |      150      |      150      |
|        1         |      299      |      299      |
|        1         |      300      |      300      |
|        1         |      301      |      --       |
|        150       |      -1       |      --       |
|        150       |      0        |      150      |
|        150       |      1        |      150      |
|        150       |      150      |      150      |
|        150       |      299      |      299      |
|        150       |      300      |      300      |
|        150       |      301      |      --       |
|        299       |      -1       |      --       |
|        299       |      0        |      299      |
|        299       |      1        |      299      |
|        299       |      150      |      299      |
|        299       |      299      |      299      |
|        299       |      300      |      300      |
|        299       |      301      |      --       |
|        300       |      -1       |      --       |
|        300       |      0        |      300      |
|        300       |      1        |      300      |
|        300       |      150      |      300      |
|        300       |      299      |      300      |
|        300       |      300      |      300      |
|        300       |      301      |      --       |
|        301       |      -1       |      --       |
|        301       |      0        |      --       |
|        301       |      1        |      --       |
|        301       |      150      |      --       |
|        301       |      299      |      --       |
|        301       |      300      |      --       |
|        301       |      301      |      --
+-----------------------------------------------------+
>>No. of Test Cases = 49
```

EXPERIMENT:6

AIM: Write a program to find the type of the triangle on the basis of sides input by the user and generate test cases to test the program using Equivalence Class Testing.

THEORY: A large number of test cases are generated for any program. It is neither feasible nor desirable to execute all such test cases. We want to select a few test cases and still wish to achieve a reasonable level of coverage. Many test cases do not test any new thing and they just execute the same lines of source code again and again. We may divide input domain into various categories with some relationship and expect that every test case from a category exhibits the same behaviour. If categories are well selected, we may assume that if one representative test case works correctly, others may also give the same results. This assumption allows us to select exactly one test case from each category and if there are four categories, four test cases may be selected. Each category is called an equivalence class and this type of testing is known as equivalence class testing.

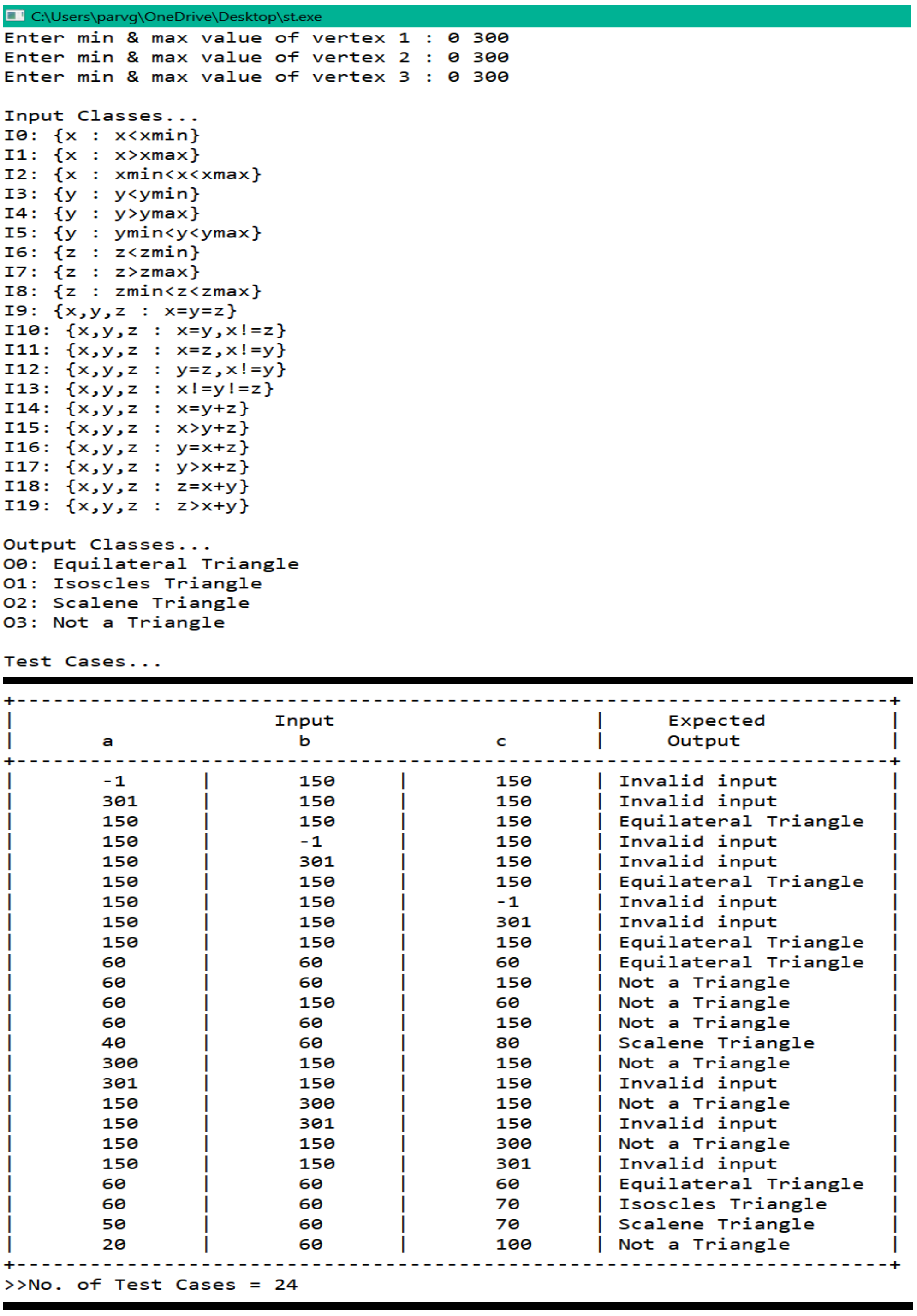
CODE:

```
/* Function to check the type of the triangle
Input: 3 vertices
Output:
-1 - Inavlid input
0 - Equilateral triangle
1 - Isosceles triangle
2 - Scalene triangle
3 - Not a triangle
*/
#include<iostream>
#include<conio.h>
using namespace std;
int triangle(int arr[3], int minarr[3], int maxarr[3]){
    for(int i=0;i<3;i++)
        if(arr[i]<minarr[i] || arr[i]>maxarr[i])
            return -1;

    if(arr[0]+arr[1]>arr[2] && arr[1]+arr[2]>arr[0] && arr[2]+arr[0]>arr[1])
    {
        if(arr[0]==arr[1] && arr[1]==arr[2] && arr[2]==arr[0])
            return 0;
        else if(arr[0]==arr[1] || arr[1]==arr[2] || arr[2]==arr[0])
            return 1;
        else
            return 2;
    }
    else
        return 3;
}
string inputClass[20], outputClass[4];
void generateClasses(){
    inputClass[0] = "{x : x<xmin}";
    inputClass[1] = "{x : x>xmax}";
    inputClass[2] = "{x : xmin<x<xmax}";
    inputClass[3] = "{y : y<ymin}";
    inputClass[4] = "{y : y>ymax}";
    inputClass[5] = "{y : ymin<y<ymax}";
    inputClass[6] = "{z : z<zmin}";
    inputClass[7] = "{z : z>zmax}";
    inputClass[8] = "{z : zmin<z<zmax}";
    inputClass[9] = "{x,y,z : x=y=z}";
    inputClass[10] = "{x,y,z : x=y,x!=z}";
    inputClass[11] = "{x,y,z : x=z,x!=y}";
    inputClass[12] = "{x,y,z : y=z,x!=y}";
    inputClass[13] = "{x,y,z : x!=y!=z}";
    inputClass[14] = "{x,y,z : x=y+z}";
    inputClass[15] = "{x,y,z : x>y+z}";
    inputClass[16] = "{x,y,z : y=x+z}";
    inputClass[17] = "{x,y,z : y>x+z}";
    inputClass[18] = "{x,y,z : z=x+y}";
    inputClass[19] = "{x,y,z : z>x+y}";

    outputClass[0] = "Equilateral Triangle";
```

```
        outputClass[1] = "Isoscles Triangle";
        outputClass[2] = "Scalene Triangle";
        outputClass[3] = "Not a Triangle";
}
void EquivalenceClassTesting(int minarr[3], int maxarr[3]){
        int expOut[24], testCases[24][3], i=0, x=0, y=0, z=0;

        generateClasses();

        cout<<"\nInput Classes...";
        for(i=0;i<20;i++)
                cout<<"\nI"<<i<<": "<<inputClass[i];

        cout<<"\n\nOutput Classes...";
        for(i=0;i<4;i++)
                cout<<"\nO"<<i<<": "<<outputClass[i];
        //for input equivalance class
        for(i=0;i<20;i++){
                switch(i)
                {
                        case 0: x=minarr[0]-1;              y=(minarr[1]+maxarr[1])/2;    z=(minarr[2]+maxarr[2])/2;
                                break;
                        case 1: x=maxarr[0]+1;              y=(minarr[1]+maxarr[1])/2;    z=(minarr[2]+maxarr[2])/2;
                                break;
                        case 2: x=(minarr[0]+maxarr[0])/2;      y=(minarr[1]+maxarr[1])/2;    z=(minarr[2]+maxarr[2])/2;
                                break;
                        case 3: y=minarr[1]-1;              x=(minarr[0]+maxarr[0])/2;    z=(minarr[2]+maxarr[2])/2;
                                break;
                        case 4: y=maxarr[1]+1;              x=(minarr[0]+maxarr[0])/2;    z=(minarr[2]+maxarr[2])/2;
                                break;
                        case 5: y=(minarr[1]+maxarr[1])/2;      x=(minarr[0]+maxarr[0])/2;    z=(minarr[2]+maxarr[2])/2;
                                break;
                        case 6: z=minarr[0]-1;              y=(minarr[1]+maxarr[1])/2;    x=(minarr[0]+maxarr[0])/2;
                                break;
                        case 7: z=maxarr[0]+1;          y=(minarr[1]+maxarr[1])/2;    x=(minarr[0]+maxarr[0])/2;
                                break;
                        case 8: z=(minarr[2]+maxarr[2])/2;      y=(minarr[1]+maxarr[1])/2;    x=(minarr[0]+maxarr[0])/2;
                                break;
                        case 9: x=y=z=60;
                                break;
                        case 10: x=y=60; z=(minarr[2]+maxarr[2])/2;
                                break;
                        case 11: x=z=60; y=(minarr[1]+maxarr[1])/2;
                                break;
                        case 12: y=z=60; z=(minarr[0]+maxarr[0])/2;
                                break;
                        case 13: x=40;    y=60;    z=80;
                                break;
                        case 14: y=(minarr[1]+maxarr[1])/2;    z=(minarr[2]+maxarr[2])/2;    x=y+z;
                                break;
                        case 15: y=(minarr[1]+maxarr[1])/2;    z=(minarr[2]+maxarr[2])/2;    x=y+z+1;
                                break;
                        case 16: x=(minarr[0]+maxarr[0])/2;    z=(minarr[2]+maxarr[2])/2;    y=x+z;
                                break;
                        case 17: x=(minarr[0]+maxarr[0])/2;    z=(minarr[2]+maxarr[2])/2;    y=x+z+1;
                                break;
                        case 18: y=(minarr[1]+maxarr[1])/2;    x=(minarr[0]+maxarr[0])/2;    z=x+y;
                                break;
                        case 19: y=(minarr[1]+maxarr[1])/2;    x=(minarr[0]+maxarr[0])/2;    z=x+y+1;
                                break;
                }
                testCases[i][0]=x;     testCases[i][1]=y;     testCases[i][2]=z;
        }
        //for output equivalance class
        for( ;i<24;i++){
                switch(i-20)
                {
                        case 0: x=y=z=60;
```

```
                        break;
                case 1: x=y=60;  z=70;
                        break;
                case 2: x=50; y=60; z=70;
                        break;
                case 3: x=20;     y=60;    z=100;
                        break;
            }
            testCases[i][0]=x;     testCases[i][1]=y;     testCases[i][2]=z;
        }
        cout<<"\n\nTest Cases...";
        cout<<"\n+";      for(int i=0;i<71;i++)  cout<<"-";    cout<<"+";
        cout<<"\n|\t\t          Input\t\t\t|        Expected\t\t|";
        cout<<"\n|\ta\t\tb\t\tc\t|         Output\t\t|";
        cout<<"\n+";      for(int i=0;i<71;i++)  cout<<"-";    cout<<"+\n";
        for(int i=0;i<24;i++)
        {
            for(int j=0;j<3;j++)
            {
                if(j==0) cout<<"|";
                cout<<"\t"<<testCases[i][j]<<"\t|";
            }
            int expOut=triangle(testCases[i],minarr,maxarr);
            if(expOut<0)
                 cout<<" Invalid input\t";
            else cout<<" "<<outputClass[expOut];
            cout<<"\t|"<<endl;
        }
        cout<<"+";   for(int i=0;i<71;i++)  cout<<"-";    cout<<"+";
        cout<<"\n>>No. of Test Cases = "<<sizeof(inputClass)/sizeof(inputClass[0]) +
sizeof(outputClass)/sizeof(outputClass[0]);
}
int main(){
        int minarr[3], maxarr[3];
        for(int i=0;i<3;i++)
        {
            cout<<"Enter min & max value of vertex "<<i+1<<" : ";
            cin>>minarr[i]>>maxarr[i];
        }
        EquivalenceClassTesting(minarr,maxarr);
        return 0;
}
```

OUTPUT:



```
C:\Users\parvg\OneDrive\Desktop\st.exe
Enter min & max value of vertex 1 : 0 300
Enter min & max value of vertex 2 : 0 300
Enter min & max value of vertex 3 : 0 300

Input Classes...
I0: {x : x<xmin}
I1: {x : x>xmax}
I2: {x : xmin<x<xmax}
I3: {y : y<ymin}
I4: {y : y>ymax}
I5: {y : ymin<y<ymax}
I6: {z : z<zmin}
I7: {z : z>zmax}
I8: {z : zmin<z<zmax}
I9: {x,y,z : x=y=z}
I10: {x,y,z : x=y,x!=z}
I11: {x,y,z : x=z,x!=y}
I12: {x,y,z : y=z,x!=y}
I13: {x,y,z : x!=y!=z}
I14: {x,y,z : x=y+z}
I15: {x,y,z : x>y+z}
I16: {x,y,z : y=x+z}
I17: {x,y,z : y>x+z}
I18: {x,y,z : z=x+y}
I19: {x,y,z : z>x+y}

Output Classes...
O0: Equilateral Triangle
O1: Isosceles Triangle
O2: Scalene Triangle
O3: Not a Triangle

Test Cases...
+-----------------------------------------------------------------------------+
|                        Input                           |      Expected       |
|       a                 b                   c         |       Output        |
+-----------------------------------------------------------------------------+
|      -1        |       150       |       150       | Invalid input        |
|      301       |       150       |       150       | Invalid input        |
|      150       |       150       |       150       | Equilateral Triangle |
|      150       |       -1        |       150       | Invalid input        |
|      150       |       301       |       150       | Invalid input        |
|      150       |       150       |       150       | Equilateral Triangle |
|      150       |       150       |       -1        | Invalid input        |
|      150       |       150       |       301       | Invalid input        |
|      150       |       150       |       150       | Equilateral Triangle |
|      60        |       60        |       60        | Equilateral Triangle |
|      60        |       60        |       150       | Not a Triangle       |
|      60        |       150       |       60        | Not a Triangle       |
|      60        |       60        |       150       | Not a Triangle       |
|      40        |       60        |       80        | Scalene Triangle     |
|      300       |       150       |       150       | Not a Triangle       |
|      301       |       150       |       150       | Invalid input        |
|      150       |       300       |       150       | Not a Triangle       |
|      150       |       301       |       150       | Invalid input        |
|      150       |       150       |       300       | Not a Triangle       |
|      150       |       150       |       301       | Invalid input        |
|      60        |       60        |       60        | Equilateral Triangle |
|      60        |       60        |       70        | Isosceles Triangle   |
|      50        |       60        |       70        | Scalene Triangle     |
|      20        |       60        |       100       | Not a Triangle       |
+-----------------------------------------------------------------------------+
>>No. of Test Cases = 24
```

EXPERIMENT:7

AIM: Write a program to find the type of the triangle on the basis of sides input by the user and generate test cases to test the program using Decision Table Testing.

THEORY: Decision tables are used in many engineering disciplines to represent complex logical relationships. An output may be dependent on many input conditions and decision tables give a pictorial view of various combinations of input conditions. There are four portions of the decision table. The four parts of the decision table are given as:

1. **Condition Stubs:** All the conditions are represented in this upper left section of the decision table. These conditions are used to determine a particular action or set of actions.
2. **Action Stubs:** All possible actions are listed in this lower left portion of the decision table.
3. **Condition Entries:** In the condition entries portion of the decision table, we have a number of columns and each column represents a rule. Values entered in this upper right portion of the table are known as inputs.
4. **Action Entries:** Each entry in the action entries portion has some associated action or set of actions in this lower right portion of the table. These values are known as outputs and are dependent upon the functionality of the program.

CODE:

```
#include<iostream>

#include<conio.h>

#include<stdlib.h>

#include<math.h>

using namespace std;

/* Function to check the type of the triangle

Input: 3 vertices

Output:

-1 - Inavlid input

0 - Not a triangle

1 - Equilateral triangle

2 - Isoscles triangle

3 - Scalene triangle

*/

int triangle(int arr[3], int minarr[3], int maxarr[3])

{

    for(int i=0;i<3;i++)

        if(arr[i]<minarr[i] || arr[i]>maxarr[i])
```

```
            return -1;

      if(arr[0]+arr[1]>arr[2] && arr[1]+arr[2]>arr[0] && arr[2]+arr[0]>arr[1])
      {
            if(arr[0]==arr[1] && arr[1]==arr[2] && arr[2]==arr[0])
                  return 1;
            else if(arr[0]==arr[1] || arr[1]==arr[2] || arr[2]==arr[0])
                  return 2;
            else
                  return 3;
      }
      else
            return 0;
}

string conditionStub[10];
string actionStub[10];
char conditionEntries[10][11];
char actionEntries[10][11];
void generateStubAndEntries()
{
      conditionStub[0] = "a<b+c?";
      conditionStub[1] = "b<c+a?";
      conditionStub[2] = "c<a+b?";
      conditionStub[3] = "a=b?";
      conditionStub[4] = "b=c?";
      conditionStub[5] = "c=a?";

      actionStub[0] = "Not a triangle";
      actionStub[1] = "Equilateral triangle";
      actionStub[2] = "Isoscles triangle";
      actionStub[3] = "Scalene triangle";
      actionStub[4] = "Impossible";
```

```
//generating Condition Entries
//for a<b+c?
int i=0;
conditionEntries[0][i]='F';
for(i=1;i<11;i++)
        conditionEntries[0][i]='T';

//for b<c+a?
i=0;    conditionEntries[1][i]='-';
i=1;    conditionEntries[1][i]='F';
for(i=2;i<11;i++)
        conditionEntries[1][i]='T';

//for c<a+b?
i=0;    conditionEntries[2][i]='-';
i=1;    conditionEntries[2][i]='-';
i=2;    conditionEntries[2][i]='F';
for(i=3;i<11;i++)
        conditionEntries[2][i]='T';

//for a=b?
for(i=0;i<3;i++)
        conditionEntries[3][i]='-';
for(i=3;i<11;i++)
        if(i<7)
                conditionEntries[3][i]='T';
        else
                conditionEntries[3][i]='F';

//for a=c?
for(i=0;i<3;i++)
        conditionEntries[4][i]='-';
for(i=3;i<11;i++)
        if(i<5 || (i>7&&i<9))
```

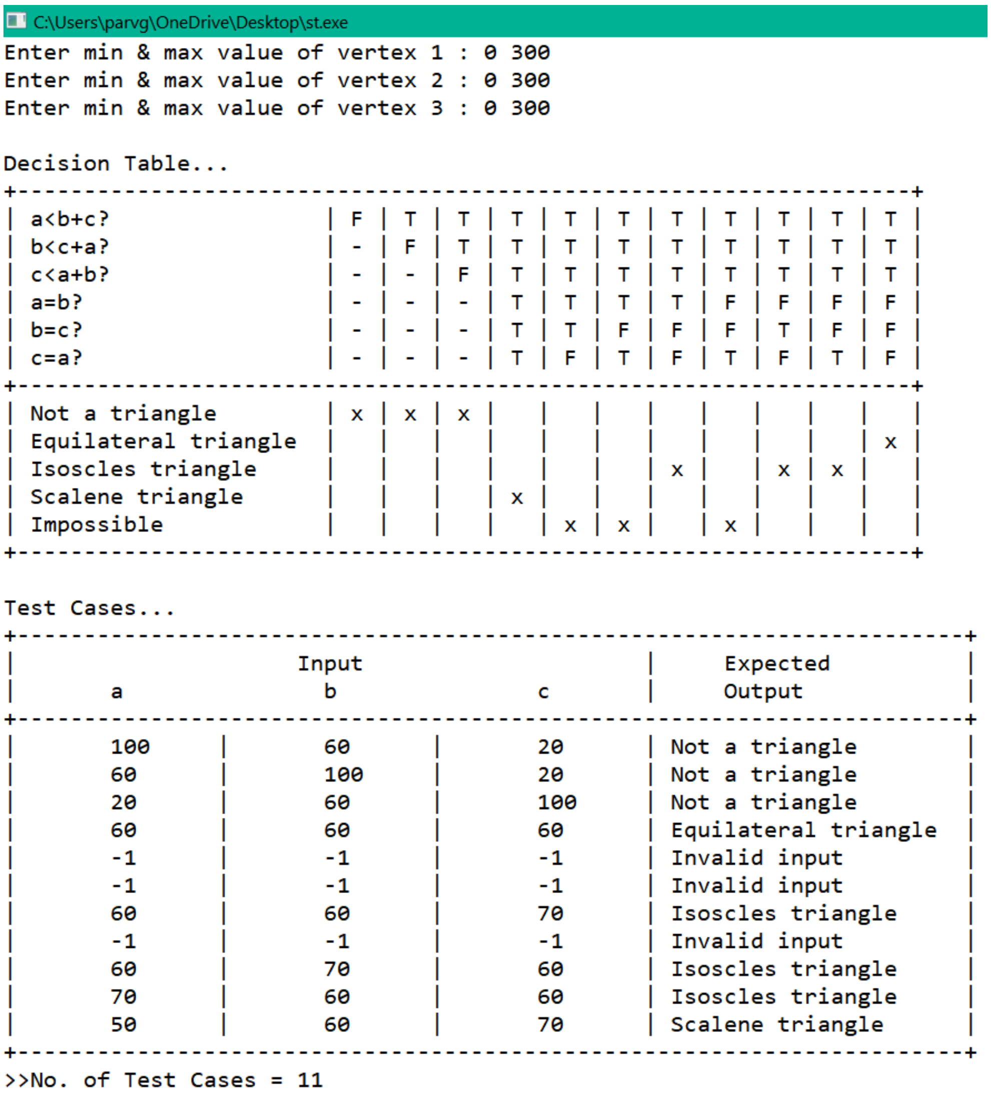
```
                conditionEntries[4][i]='T';
        else
                conditionEntries[4][i]='F';

//for b=c?
for(i=0;i<3;i++)
        conditionEntries[5][i]='-';
for(i=3;i<11;i++)
        if(i%2)
                conditionEntries[5][i]='T';
        else
                conditionEntries[5][i]='F';



//generating Action Entries
for(i=0;i<5;i++)
        for(int j=0;j<11;j++)
                actionEntries[i][j]=' ';

actionEntries[0][0]='x';
actionEntries[0][1]='x';
actionEntries[0][2]='x';
actionEntries[3][3]='x';
actionEntries[4][4]='x';
actionEntries[4][5]='x';
actionEntries[2][6]='x';
actionEntries[4][7]='x';
actionEntries[2][8]='x';
actionEntries[2][9]='x';
actionEntries[1][10]='x';

cout<<"\nDecision Table...\n";
cout<<"+";   for(i=0;i<67;i++)cout<<"-";    cout<<"+\n";
for(i=0;i<6;i++)
```

```
	{
		cout<<"| "<<conditionStub[i]<<"\t\t";
		if(i>=3) cout<<"\t";
		cout<<"| ";
		for(int j=0;j<11;j++)
		{
			cout<<conditionEntries[i][j]<<" | ";
		}
		cout<<endl;
	}
	cout<<"+";   for(i=0;i<67;i++)cout<<"-";    cout<<"+\n";
	for(i=0;i<5;i++)
	{
		cout<<"| "<<actionStub[i]<<"\t";
		if(i==4) cout<<"\t";
		cout<<"| ";
		for(int j=0;j<11;j++)
		{
			cout<<actionEntries[i][j]<<" | ";
		}
		cout<<endl;
	}
	cout<<"+";   for(i=0;i<67;i++)cout<<"-";    cout<<"+\n";
}

void DecisionTableTesting(int minarr[3], int maxarr[3])
{
	int expOut[11], testCases[11][3], i=0, x=0, y=0, z=0;

	generateStubAndEntries();
	for(i=0;i<11;i++)
	{
		switch(i)
		{
```

```
		case 0: x=100;	y=60;	z=20;
			break;
		case 1: x=60;	y=100;	z=20;
			break;
		case 2: x=20;	y=60;	z=100;
			break;
		case 3: x=y=z=60;
			break;
		case 4: x=y=z=-1;
			break;
		case 5: x=y=z=-1;
			break;
		case 6: x=y=60;	z=70;
			break;
		case 7: x=y=z=-1;
			break;
		case 8: x=z=60;	y=70;
			break;
		case 9: y=z=60;	x=70;
			break;
		case 10: x=50; y=60; z=70;
			break;
	}
	testCases[i][0]=x;	testCases[i][1]=y;	testCases[i][2]=z;
}

cout<<"\nTest Cases...";
cout<<"\n+";	for(int i=0;i<71;i++)  cout<<"-";	cout<<"+";
cout<<"\n|\t\t		Input\t\t\t|		Expected\t\t|";
cout<<"\n|\ta\t\tb\t\tc\t|		Output\t\t|";
cout<<"\n+";	for(int i=0;i<71;i++)  cout<<"-";	cout<<"+\n";
for(int i=0;i<11;i++)
{
	for(int j=0;j<3;j++)
```

```
            {
                if(j==0) cout<<"|";
                cout<<"\t"<<testCases[i][j]<<"\t|";
            }
            int expOut=triangle(testCases[i],minarr,maxarr);
            if(expOut<0)
                 cout<<" Invalid input\t";
            else cout<<" "<<actionStub[expOut];
            cout<<"\t|"<<endl;
        }
        cout<<"+";   for(int i=0;i<71;i++)  cout<<"-";    cout<<"+";
        cout<<"\n>>No. of Test Cases = 11";
}

int main()
{
        int minarr[3], maxarr[3];
        for(int i=0;i<3;i++){
            cout<<"Enter min & max value of vertex "<<i+1<<" : ";
            cin>>minarr[i]>>maxarr[i];
        }
        DecisionTableTesting(minarr,maxarr);
        return 0;
}
```

OUTPUT:



```
Enter min & max value of vertex 1 : 0 300
Enter min & max value of vertex 2 : 0 300
Enter min & max value of vertex 3 : 0 300

Decision Table...
+---------------------------------------------------------------------+
| a<b+c?                | F | T | T | T | T | T | T | T | T | T | T |
| b<c+a?                | - | F | T | T | T | T | T | T | T | T | T |
| c<a+b?                | - | - | F | T | T | T | T | T | T | T | T |
| a=b?                  | - | - | - | T | T | T | T | F | F | F | F |
| b=c?                  | - | - | - | T | T | F | F | F | T | F | F |
| c=a?                  | - | - | - | T | F | T | F | T | F | T | F |
+---------------------------------------------------------------------+
| Not a triangle        | x | x | x |   |   |   |   |   |   |   |   |
| Equilateral triangle  |   |   |   |   |   |   |   |   |   |   | x |
| Isoscles triangle     |   |   |   |   |   |   | x |   | x | x |   |
| Scalene triangle      |   |   |   | x |   |   |   |   |   |   |   |
| Impossible            |   |   |   |   | x | x |   | x |   |   |   |
+---------------------------------------------------------------------+

Test Cases...
+-----------------------------------------------------------------------------+
|                      Input                       |       Expected           |
|       a               b               c           |       Output             |
+-----------------------------------------------------------------------------+
|       100     |       60      |       20          | Not a triangle           |
|       60      |       100     |       20          | Not a triangle           |
|       20      |       60      |       100         | Not a triangle           |
|       60      |       60      |       60          | Equilateral triangle     |
|       -1      |       -1      |       -1          | Invalid input            |
|       -1      |       -1      |       -1          | Invalid input            |
|       60      |       60      |       70          | Isoscles triangle        |
|       -1      |       -1      |       -1          | Invalid input            |
|       60      |       70      |       60          | Isoscles triangle        |
|       70      |       60      |       60          | Isoscles triangle        |
|       50      |       60      |       70          | Scalene triangle         |
+-----------------------------------------------------------------------------+
>>No. of Test Cases = 11
```

EXPERIMENT:8

AIM: Write a program to find Cyclomatic complexity of a program.

THEORY: There are many paths in any program. If there are loops in a program, the number of paths increases drastically. In such situations, we may be traversing the same nodes and edges again and again. However, as defined earlier, an independent path should have at least one new node or edge which is to be traversed. We should identify every independent path of a program and pay special attention to these paths during testing. A few concepts of graph theory are used in testing techniques which may help us to identify independent paths. This concept involves using cyclomatic number of graph theory which has been redefined as cyclomatic complexity. This is nothing but the number of independent paths through a program.

(i) V(G) = e – n + 2P
where V(G) = Cyclomatic complexity
G : program graph
n : number of nodes
e : number of edges
P : number of connected components

(ii) Cyclomatic complexity is equal to the number of regions of the program graph.

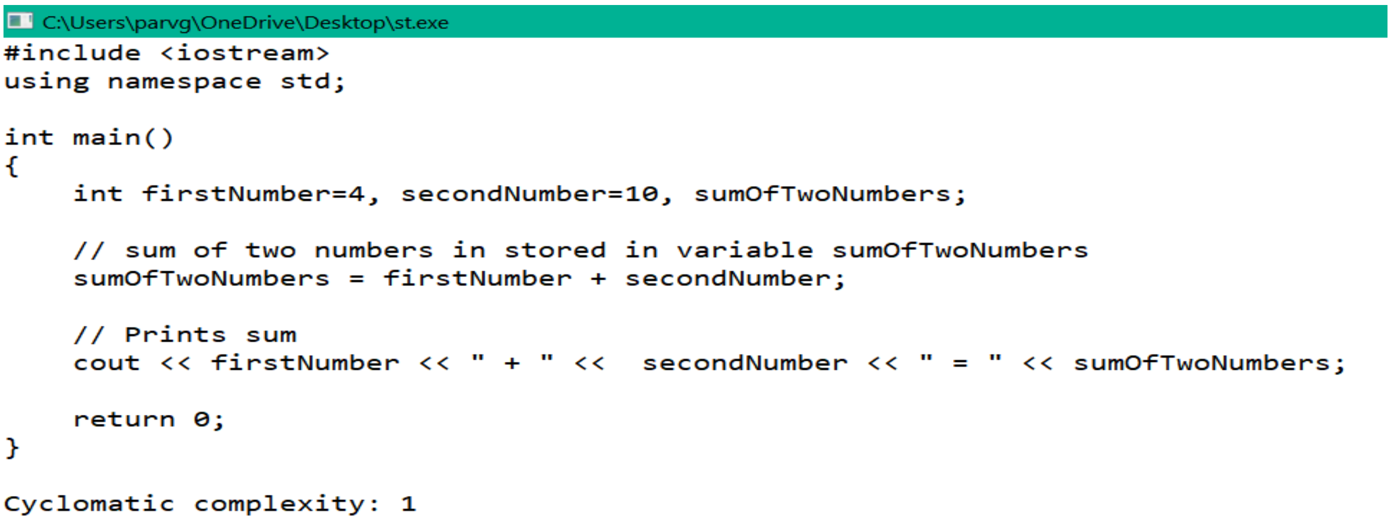
(iii) Cyclomatic complexity
V(G) =pi+ 1 (where pi is no. Of predicate nodes)

CODE:

```
#include<fstream>

#include<iostream>

#include<string.h>

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

using namespace std;

int main()

{

    fstream testFile;

    testFile.open("C:\\Users\\parvg\\Downloads\\my stlabfile\\tempfile.txt",ios::in);

    int i=0, cyclo=0, len=0;

    char a[1000];


    if(testFile.is_open())
```

```
{
        while(testFile)
        {
                a[i]=testFile.get();
                cout<<a[i++];
        }
        len=strlen(a);
}

for(i=0;i<len-3;i++)
{
        if(a[i]=='i')
        {
                i++;
                if(a[i]=='f')
                {
                        i++;
                        if(a[i]=='(')
                                cyclo++;
                }
        }
        else if(a[i]=='w')
        {
                i++;
                if(a[i]=='h')
                {
                        i++;
                        if(a[i]=='i')
                        {
                                i++;
                                if(a[i]=='l')
                                {
                                        i++;
                                        if(a[i]=='e')
                                {
                                        i++;
                                        if(a[i]=='(')
                                                cyclo++;
                                }
                        }
                }
        }
    }


}


cout<<"\n\nCyclomatic complexity: "<<cyclo+1;
testFile.close();
return 0;
}
```

OUTPUT:



```
#include <iostream>
using namespace std;

int main()
{
    int firstNumber=4, secondNumber=10, sumOfTwoNumbers;

    // sum of two numbers in stored in variable sumOfTwoNumbers
    sumOfTwoNumbers = firstNumber + secondNumber;

    // Prints sum
    cout << firstNumber << " + " <<  secondNumber << " = " << sumOfTwoNumbers;

    return 0;
}

Cyclomatic complexity: 1
```

EXPERIMENT:9

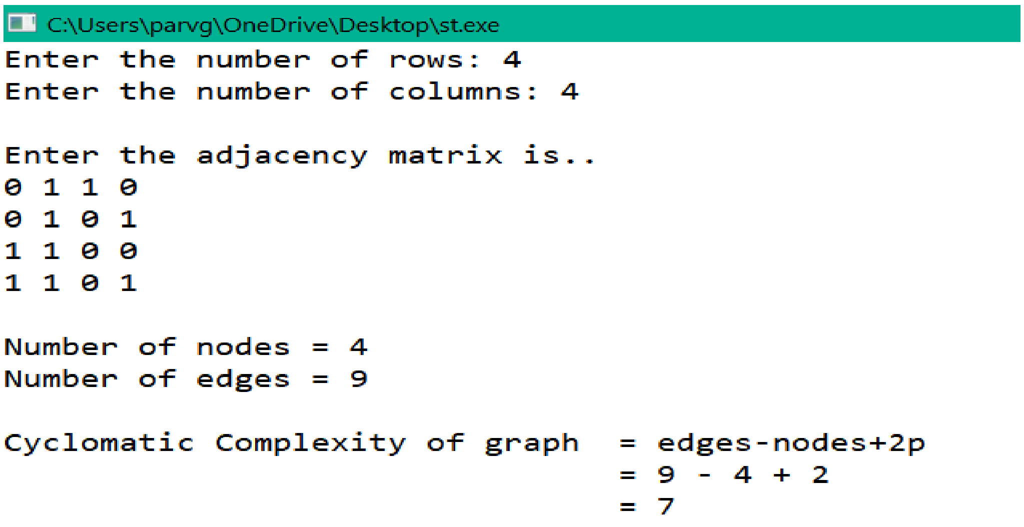
AIM: Write a program to input graph matrix and perform DD path testing.

CODE:

```
#include<iostream>
#define MAX 20
using namespace std;
int main(){
	int row, col, i, j, edges=0, nodes=0;
	char ch;
	int graphmatrix[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>>graphmatrix[i][j];

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

	for(i=0;i<=row;i++)
		for(j=0;j<=col;j++)
			if(graphmatrix[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;
        return 0;
}
```

OUTPUT:


C:\Users\parvg\OneDrive\Desktop\st.exe

```
Enter the number of rows: 4
Enter the number of columns: 4

Enter the adjacency matrix is..
0 1 1 0
0 1 0 1
1 1 0 0
1 1 0 1

Number of nodes = 4
Number of edges = 9

Cyclomatic Complexity of graph   = edges-nodes+2p
                                 = 9 - 4 + 2
                                 = 7
```

EXPERIMENT:10

AIM: Write a program to perform Mutation Testing.

THEORY: Mutation testing is used to design new software tests and evaluate the quality of existing software tests. Mutation testing involves modifying a program in small ways. Each mutated version is called a *mutant* and tests detect and reject mutants by causing the behavior of the original version to differ from the mutant. This is called *killing* the mutant. Test suites are measured by the percentage of mutants that they kill. New tests can be designed to kill additional mutants. Mutants are based on well-defined *mutation operators* that either mimic typical programming errors (such as using the wrong operator or variable name) or force the creation of valuable tests (such as dividing each expression by zero). The purpose is to help the tester develop effective tests or locate weaknesses in the test data used for the program or in sections of the code that are seldom or never accessed during execution. Mutation testing is a form of white-box testing.

CODE:

```
#include<iostream>

#include<conio.h>

#include<math.h>

using namespace std;

class Rectangle

{

    int length;

    int breadth;

    static int count;

public:

Rectangle();

int getLength();

int getBreadth();

void storeData(int , int );

int calculateArea(int , int );

int objectCount();

};

int Rectangle :: count = 0;

Rectangle :: Rectangle()

{
        count++;
        length = 0;
        breadth = 0;
}
int Rectangle :: getLength()
{
        return length;
}
int Rectangle :: getBreadth(){
        return breadth;
}

void Rectangle :: storeData(int l, int b){
        length = l;
        breadth = b;
}

int Rectangle :: calculateArea(int length, int breadth)
{
        return length * breadth;
}

int Rectangle :: objectCount()
{
        return count;
}
int main()
{
        char ch='y';
        do{
                Rectangle R;

                int l, b;
                cout<<"\nEnter the dimensions of the rectangle...\n";
                cout<<"Length = ";
                cin>>l;
                cout<<"Breadth = ";
                cin>>b;
                R.storeData(l,b);

                cout<<"Area of the rectangle = "<<R.calculateArea(R.getLength(),R.getBreadth());

                cout<<"\n\nDo you want to calculate more? (y/n): ";
                cin>>ch;
                if(ch=='n' || ch=='N')
                    cout<<"\nObject count = "<<R.objectCount();
        }while(ch=='y' || ch=='Y');

        return 0;
}
```

Mutated Statements:

Mutant No.	Line No.	Original Line	Modified Line
M1	13	public:	private:
M2	11	static int count;	int count;
M3	41	void Rectangle :: storeData(int l, int b)	void Rectangle :: storeData(int b, int l)
M4	49	return length * breadth;	return length + breadth;
M5	27	length = 0;	length = 10;

Mutant M1:

Test Cases	length	Breadth	Expected Output	Actual Output
1	2	2	4	Compilation Error
2	3	5	15	Compilation Error

Mutant M2:

Test Cases	length	Breadth	Expected Output	Actual Output
1	2	2	4	Compilation Error
2	3	5	15	Compilation Error

Mutant M3:

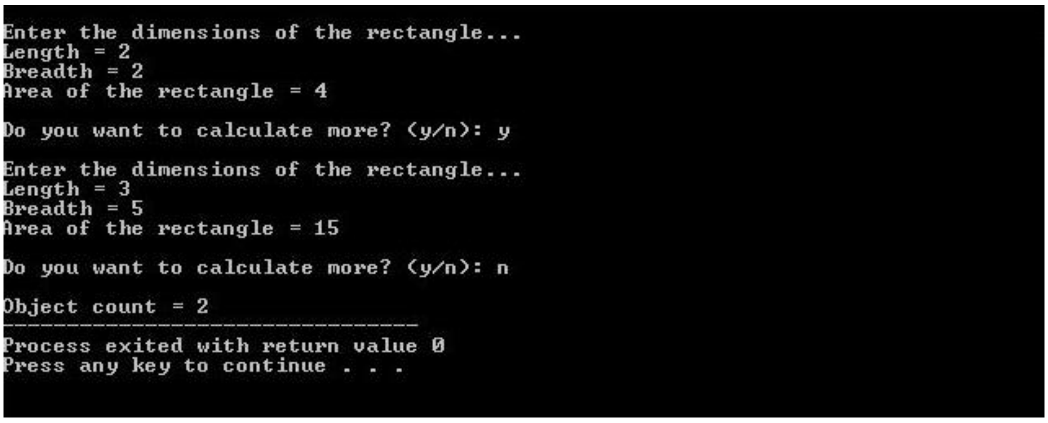
Test Cases	length	Breadth	Expected Output	Actual Output
1	2	2	4	4
2	3	5	15	15

Mutant M4:

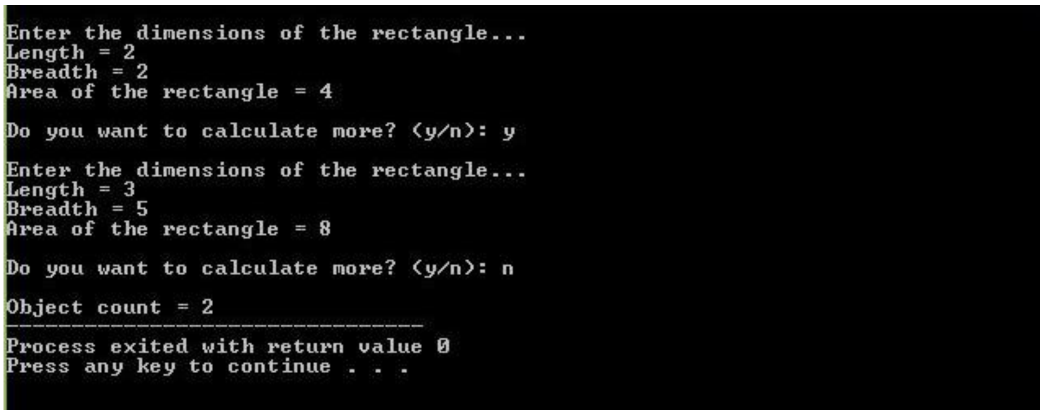
Test Cases	length	breadth	Expected Output	Actual Output
1	2	2	4	4
2	3	5	15	8

Mutant M5:

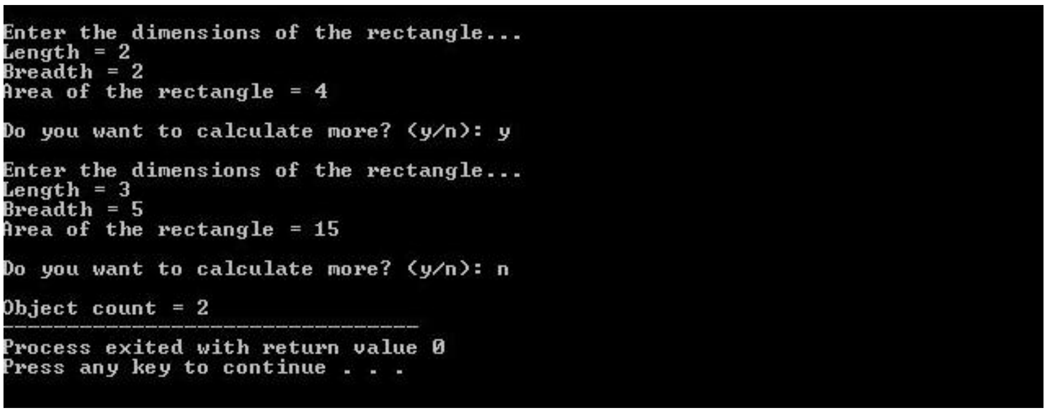
Test Cases	length	breadth	Expected Output	Actual Output
1	2	2	4	4
2	3	5	15	15



```
Enter the dimensions of the rectangle...
Length = 2
Breadth = 2
Area of the rectangle = 4

Do you want to calculate more? (y/n): y

Enter the dimensions of the rectangle...
Length = 3
Breadth = 5
Area of the rectangle = 15

Do you want to calculate more? (y/n): n

Object count = 2
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```



```
Enter the dimensions of the rectangle...
Length = 2
Breadth = 2
Area of the rectangle = 4

Do you want to calculate more? (y/n): y

Enter the dimensions of the rectangle...
Length = 3
Breadth = 5
Area of the rectangle = 8

Do you want to calculate more? (y/n): n

Object count = 2
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```



```
Enter the dimensions of the rectangle...
Length = 2
Breadth = 2
Area of the rectangle = 4

Do you want to calculate more? (y/n): y

Enter the dimensions of the rectangle...
Length = 3
Breadth = 5
Area of the rectangle = 15

Do you want to calculate more? (y/n): n

Object count = 2
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Calculation of Mutation Score:

No. of killed mutants = 3
No. of live mutants = 2
Total no. of mutants = 5
Mutation Score = No. of killed mutants / Total no. of mutants
= 3/5
= **0.6**