SWE:502 SOFTWARE TESTING PRACTICAL FILE

Submitted towards the partial fulfilment of the requirements of the award of the degree of

Master of Technology
In
Software Engineering

Submitted To:-

Dr. Ruchika Malhotra
Department of Software Engineering

Submitted By:-

Aman Chauhan (24/SWE/08)
II SEM, I YEAR

Delhi Technological University

(FORMERLY Delhi College of Engineering)
Bawana Road, New Delhi-110042

April, 2025

INDEX

Program 1

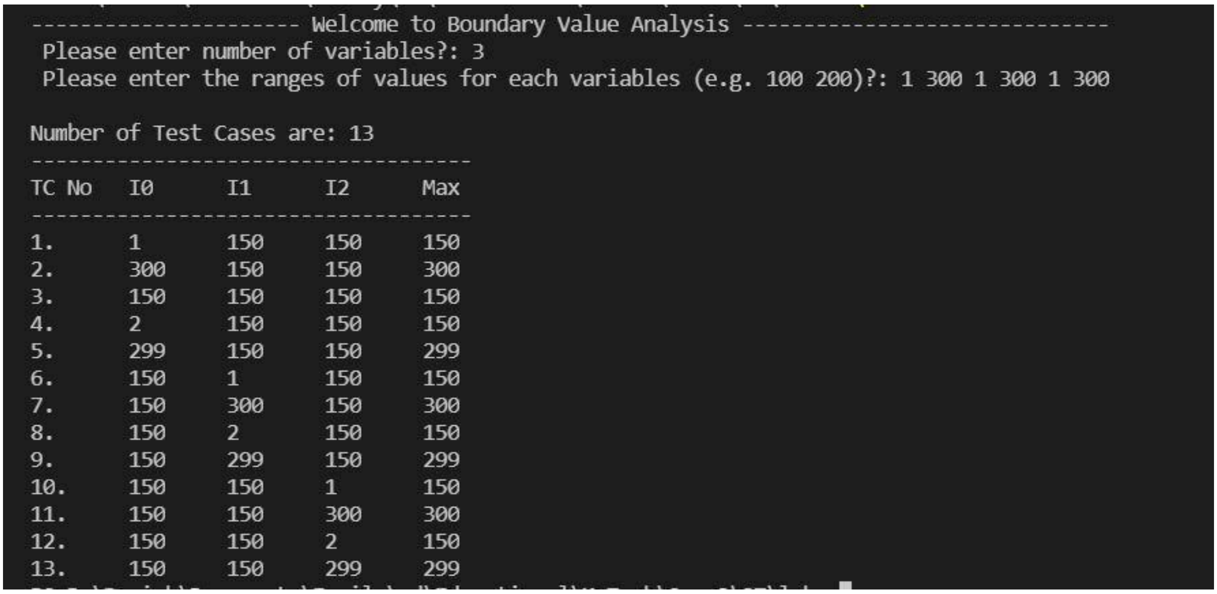
Aim: Write a program to find the maximum in three numbers input by the user and generate test cases for the program using Boundary Value Analysis.

Source Code: Boundary Value Analysis

```
#include <iostream>
#include <string>

using namespace std;
int maximum(int x, int y){
        int max = x;
        if (x < y)
                max = y;
        return max;
}

int main(){
        int n, x, y, z;
        cout << "--------------------- Welcome to Boundary Value Analysis_\n ";
        cout<< "Please enter number of variables?: "; cin >> n;
        int values[n][5]; // 0- min value, 1- max value, 2-nominal value, 3- Just above the
minimum, 4-just below the maximum
        cout << " Please enter the ranges of values for each variables (e.g. 100 200)?:
";
        for (int i = 0; i < n; i++){
                cin >> values[i][0] >> values[i][1];
                values[i][2] = (values[i][0] + values[i][1]) / 2; values[i][3] = values[i][0]
                + 1;
                values[i][4] = values[i][1] - 1;
        }
        int num_of_cols = 4 * n + 1;
        cout << "\nNumber of Test Cases are: " << num_of_cols
<<endl;
        cout<<"________________________________________\nTC No\t";
        for (int i = 0; i < n; i++) cout << "I"<<
                i<<"\t";
        cout << "Max";
        cout << "\n_________________________________________" << endl;
        int r = 0, index = 1;
        bool a_printed = false; while (r <
        n){
                for (int i = 0; i < 5; i++){
                        if (a_printed == false || (a_printed == true && i
!= 2)){
```

```
                int max = INT_MIN;
                cout << index << ".\t";
                for (int j = 0; j < n; j++){
                    if (j != r){
                        cout << values[j][2] << "\t";
                        max = maximum(max, values[j][2]);
                    }
                    else{
                        cout << values[r][i] << "\t";
                        max = maximum(max, values[r][i]);
                    }
                }
                cout << max << "\t\n"; index++;
            }
        }
        a_printed = true; r++;
    }
    return 0;
}
```

Output:



```
--------------------- Welcome to Boundary Value Analysis ------------------------------
 Please enter number of variables?: 3
 Please enter the ranges of values for each variables (e.g. 100 200)?: 1 300 1 300 1 300

Number of Test Cases are: 13
------------------------------------
TC No   I0      I1      I2      Max
------------------------------------
1.      1       150     150     150
2.      300     150     150     300
3.      150     150     150     150
4.      2       150     150     150
5.      299     150     150     299
6.      150     1       150     150
7.      150     300     150     300
8.      150     2       150     150
9.      150     299     150     299
10.     150     150     1       150
11.     150     150     300     300
12.     150     150     2       150
13.     150     150     299     299
```

Learning Outcome:

1. Successfully completed boundary value analysis.
2. Boundary value analysis of maximum of three numbers.

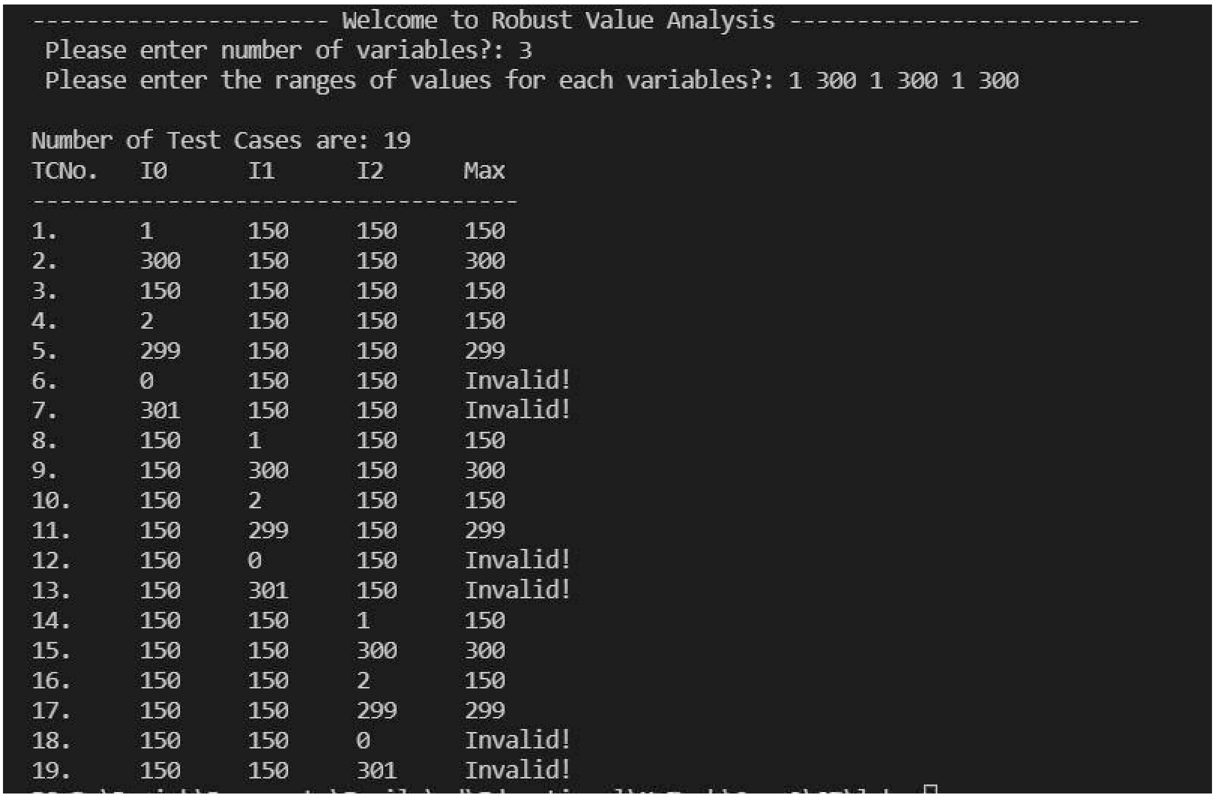
Program 2

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.

Source Code: Robust Approach

```
#include <iostream>
#include <string>
using namespace std;
int maximum(int x, int y){
        int max = x;
        if (x < y)
                max = y;
        return max;
}
int main(){
        int n, x, y, z;
        cout << "--------------------- Welcome to Robust Value Analysis____\n ";
        cout<< "Please enter number of variables?: "; cin >> n;
        int arr[n][7]; // 0- min value, 1- max value, 2-nominal value, 3- Just above the
minimum,4 - just below the maximum
        cout<< " Please enter the ranges of values for each variables?: ";
        for (int i = 0; i < n; i++){
                cin >> arr[i][0] >> arr[i][1];
                arr[i][2] = (arr[i][0] + arr[i][1]) / 2; arr[i][3] = arr[i][0] + 1;
                arr[i][4] = arr[i][1] - 1;
                arr[i][5] = arr[i][0] - 1;
                arr[i][6] = arr[i][1] + 1;
        }
        int num_of_cols = 6 * n + 1;
        cout << "\nNumber of Test Cases are: " << num_of_cols << "\nTCNo.\t";
        for (int i = 0; i < n; i++) cout << "I"<<
                i<<"\t";
        cout << "Max";
        cout << "\n________________________________________" << endl;
        int r = 0, index_of_array = 1;
        bool printed_a = false; while (r <
        n){
                for (int i = 0; i < 7; i++){
                        if (printed_a == false || (printed_a == true && i
!= 2)){
                                int max = INT_MIN;
                cout << index_of_array << ".\t";
                for (int j = 0; j < n; j++){
                    if (j != r){
                        cout << arr[j][2] << "\t";
                        max = maximum(max, arr[j][2]);
                    }
                    else{
                        cout << arr[r][i] << "\t";
                        max = maximum(max, arr[r][i]);
                    }
                }
                  if (i >= 5){
                       cout << "Invalid!\n";
                }
                else
                    cout << max << "\t\n";
                index_of_array++;
            }
        }
        if (printed_a != true) printed_a
            = true;
        r++;
    }
    return 0;
}
```

Output:



```
---------------------- Welcome to Robust Value Analysis --------------------------
 Please enter number of variables?: 3
 Please enter the ranges of values for each variables?: 1 300 1 300 1 300

Number of Test Cases are: 19
TCNo.   I0      I1      I2      Max
------------------------------------
1.      1       150     150     150
2.      300     150     150     300
3.      150     150     150     150
4.      2       150     150     150
5.      299     150     150     299
6.      0       150     150     Invalid!
7.      301     150     150     Invalid!
8.      150     1       150     150
9.      150     300     150     300
10.     150     2       150     150
11.     150     299     150     299
12.     150     0       150     Invalid!
13.     150     301     150     Invalid!
14.     150     150     1       150
15.     150     150     300     300
16.     150     150     2       150
17.     150     150     299     299
18.     150     150     0       Invalid!
19.     150     150     301     Invalid!
```

Learning Outcome:

1. Successfully completed Robust Approach.
2. Robust Approach in maximum of three numbers.

Program 3

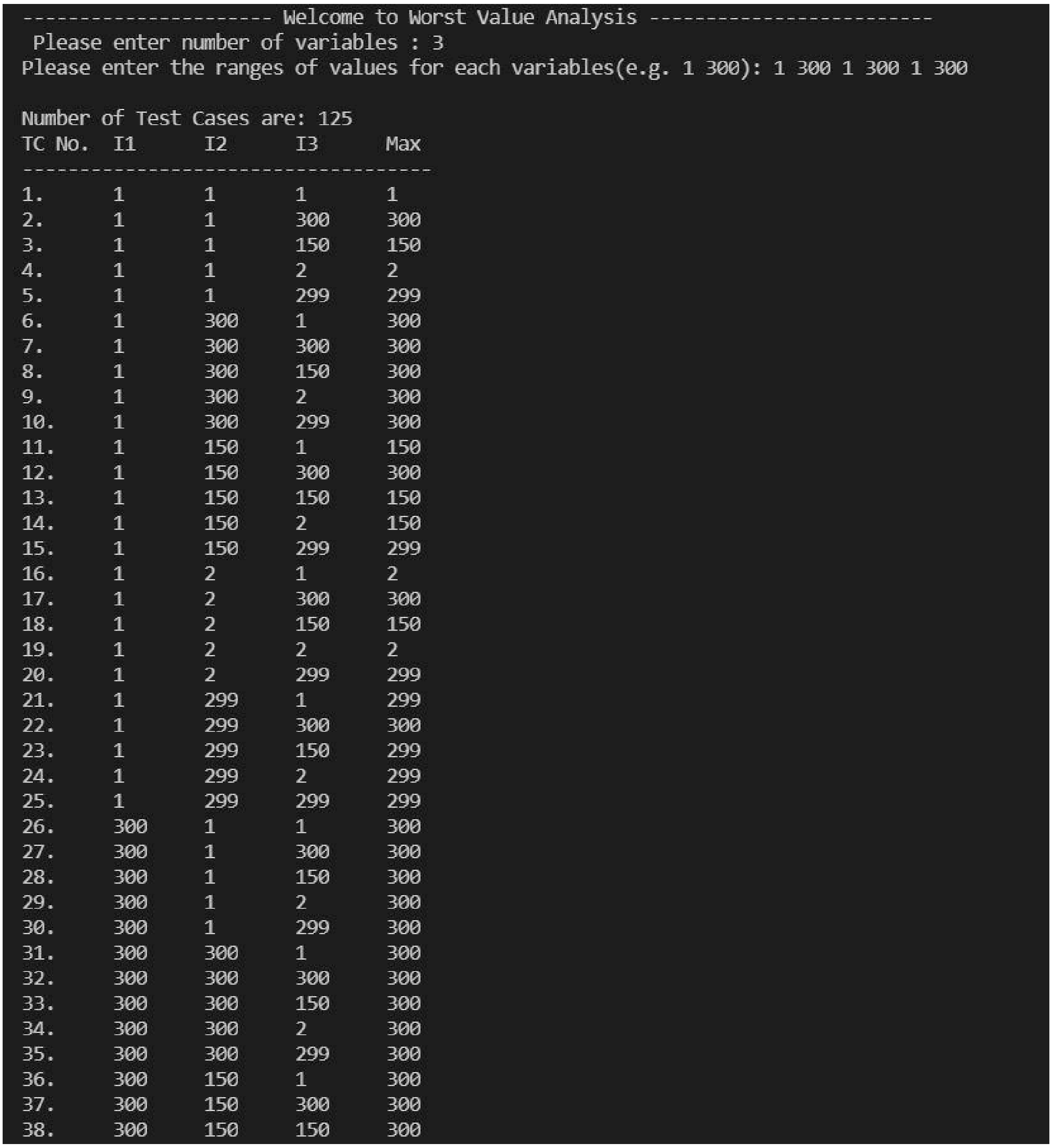
Aim: Write a program to find the maximum in three numbers input by the user and generate test cases for the program using Worst Boundary Value Analysis.

Source Code: Worst Boundary Value Analysis

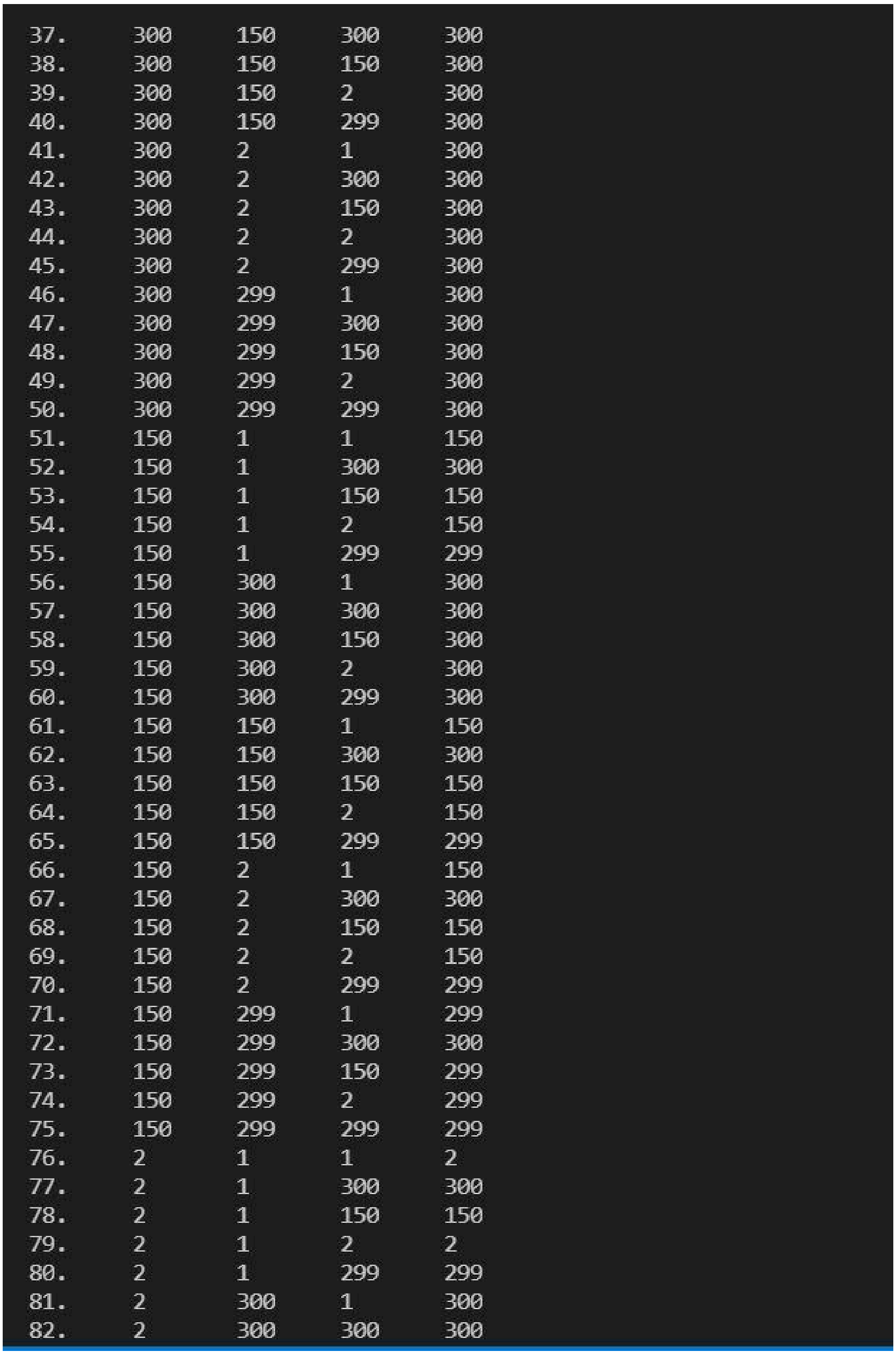
```
#include <iostream>
#include <cmath> #include
<string>
using namespace std;
int maximum(int x, int y){
        int max = x;
        if (x < y)
                max = y;
        return max;
}
int main()
{
        int n;
        cout << "--------------------- Welcome to Worst Value Analysis_____\n ";
        cout<< "Please enter number of variables : "; cin >> n;
        int values[n][5]; // 0- min value, 1- max value, 2-nominal value, 3- Just above the
minimum,4 - just below the maximum
        cout<< "Please enter the ranges of values for each variables(e.g. 1 300): ";
        for (int i = 0; i < n; i++){
                cin >> values[i][0] >> values[i][1];
                values[i][2] = (values[i][0] + values[i][1]) / 2; values[i][3] = values[i][0]
                + 1;
                values[i][4] = values[i][1] - 1;
        }
        int num_of_cols = pow(5, n);
        cout <<"\nNumber of Test Cases are: " << num_of_cols << "\nTC No.\t";
        for (int i = 0; i < n; i++) cout <<"I" <<i +
                1<<"\t";
        cout <<"Max";
        cout << "\n____________________________________________________" << endl;
        int r = 0, index_of_array = 1;
        for (int i = 0; i < 5; i++){
                for (int j = 0; j < 5; j++){
                        for (int k = 0; k < 5; k++){
```

```
                cout << index_of_array << ".\t" << values[0][i] <<"\t"<<
values[1][j] <<"\t"<< values[2][k] << "\t";
                cout << maximum(maximum(values[0][i],
values[1][j]), values[2][k]) << "\t"<<endl;
                    index_of_array++;
            }
        }
    }
    return 0;
}
```

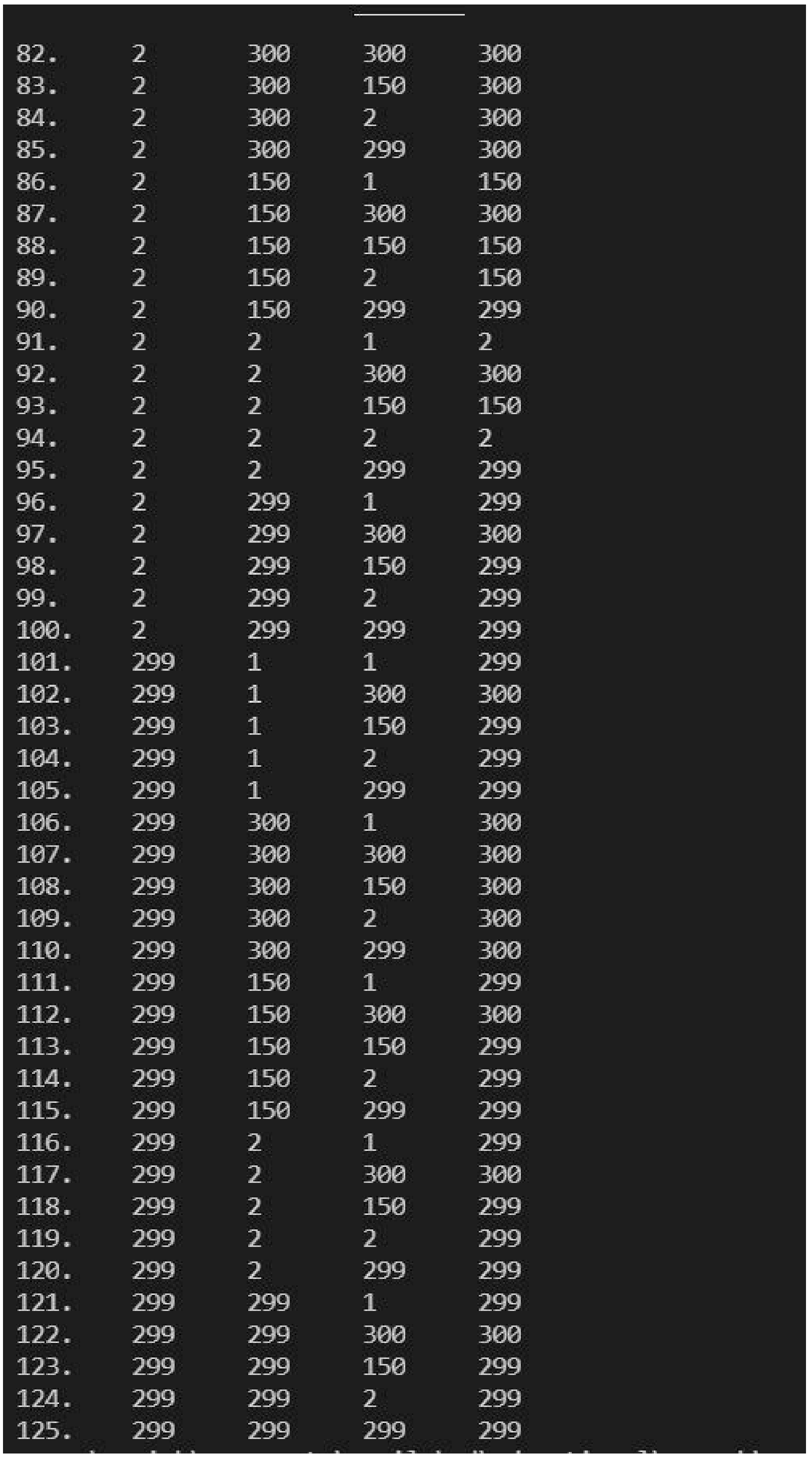
Output:



```
---------------------- Welcome to Worst Value Analysis -------------------------
 Please enter number of variables : 3
Please enter the ranges of values for each variables(e.g. 1 300): 1 300 1 300 1 300

Number of Test Cases are: 125
TC No.  I1      I2      I3      Max
------------------------------------
1.      1       1       1       1
2.      1       1       300     300
3.      1       1       150     150
4.      1       1       2       2
5.      1       1       299     299
6.      1       300     1       300
7.      1       300     300     300
8.      1       300     150     300
9.      1       300     2       300
10.     1       300     299     300
11.     1       150     1       150
12.     1       150     300     300
13.     1       150     150     150
14.     1       150     2       150
15.     1       150     299     299
16.     1       2       1       2
17.     1       2       300     300
18.     1       2       150     150
19.     1       2       2       2
20.     1       2       299     299
21.     1       299     1       299
22.     1       299     300     300
23.     1       299     150     299
24.     1       299     2       299
25.     1       299     299     299
26.     300     1       1       300
27.     300     1       300     300
28.     300     1       150     300
29.     300     1       2       300
30.     300     1       299     300
31.     300     300     1       300
32.     300     300     300     300
33.     300     300     150     300
34.     300     300     2       300
35.     300     300     299     300
36.     300     150     1       300
37.     300     150     300     300
38.     300     150     150     300
```



```
37.     300     150     300     300
38.     300     150     150     300
39.     300     150     2       300
40.     300     150     299     300
41.     300     2       1       300
42.     300     2       300     300
43.     300     2       150     300
44.     300     2       2       300
45.     300     2       299     300
46.     300     299     1       300
47.     300     299     300     300
48.     300     299     150     300
49.     300     299     2       300
50.     300     299     299     300
51.     150     1       1       150
52.     150     1       300     300
53.     150     1       150     150
54.     150     1       2       150
55.     150     1       299     299
56.     150     300     1       300
57.     150     300     300     300
58.     150     300     150     300
59.     150     300     2       300
60.     150     300     299     300
61.     150     150     1       150
62.     150     150     300     300
63.     150     150     150     150
64.     150     150     2       150
65.     150     150     299     299
66.     150     2       1       150
67.     150     2       300     300
68.     150     2       150     150
69.     150     2       2       150
70.     150     2       299     299
71.     150     299     1       299
72.     150     299     300     300
73.     150     299     150     299
74.     150     299     2       299
75.     150     299     299     299
76.     2       1       1       2
77.     2       1       300     300
78.     2       1       150     150
79.     2       1       2       2
80.     2       1       299     299
81.     2       300     1       300
82.     2       300     300     300
```



```
82.     2       300     300     300
83.     2       300     150     300
84.     2       300     2       300
85.     2       300     299     300
86.     2       150     1       150
87.     2       150     300     300
88.     2       150     150     150
89.     2       150     2       150
90.     2       150     299     299
91.     2       2       1       2
92.     2       2       300     300
93.     2       2       150     150
94.     2       2       2       2
95.     2       2       299     299
96.     2       299     1       299
97.     2       299     300     300
98.     2       299     150     299
99.     2       299     2       299
100.    2       299     299     299
101.    299     1       1       299
102.    299     1       300     300
103.    299     1       150     299
104.    299     1       2       299
105.    299     1       299     299
106.    299     300     1       300
107.    299     300     300     300
108.    299     300     150     300
109.    299     300     2       300
110.    299     300     299     300
111.    299     150     1       299
112.    299     150     300     300
113.    299     150     150     299
114.    299     150     2       299
115.    299     150     299     299
116.    299     2       1       299
117.    299     2       300     300
118.    299     2       150     299
119.    299     2       2       299
120.    299     2       299     299
121.    299     299     1       299
122.    299     299     300     300
123.    299     299     150     299
124.    299     299     2       299
125.    299     299     299     299
```

Learning Outcome:

1. Successfully completed Worst Boundary Value Analysis.
2. Worst Boundary Value Analysis in maximum of three numbers.

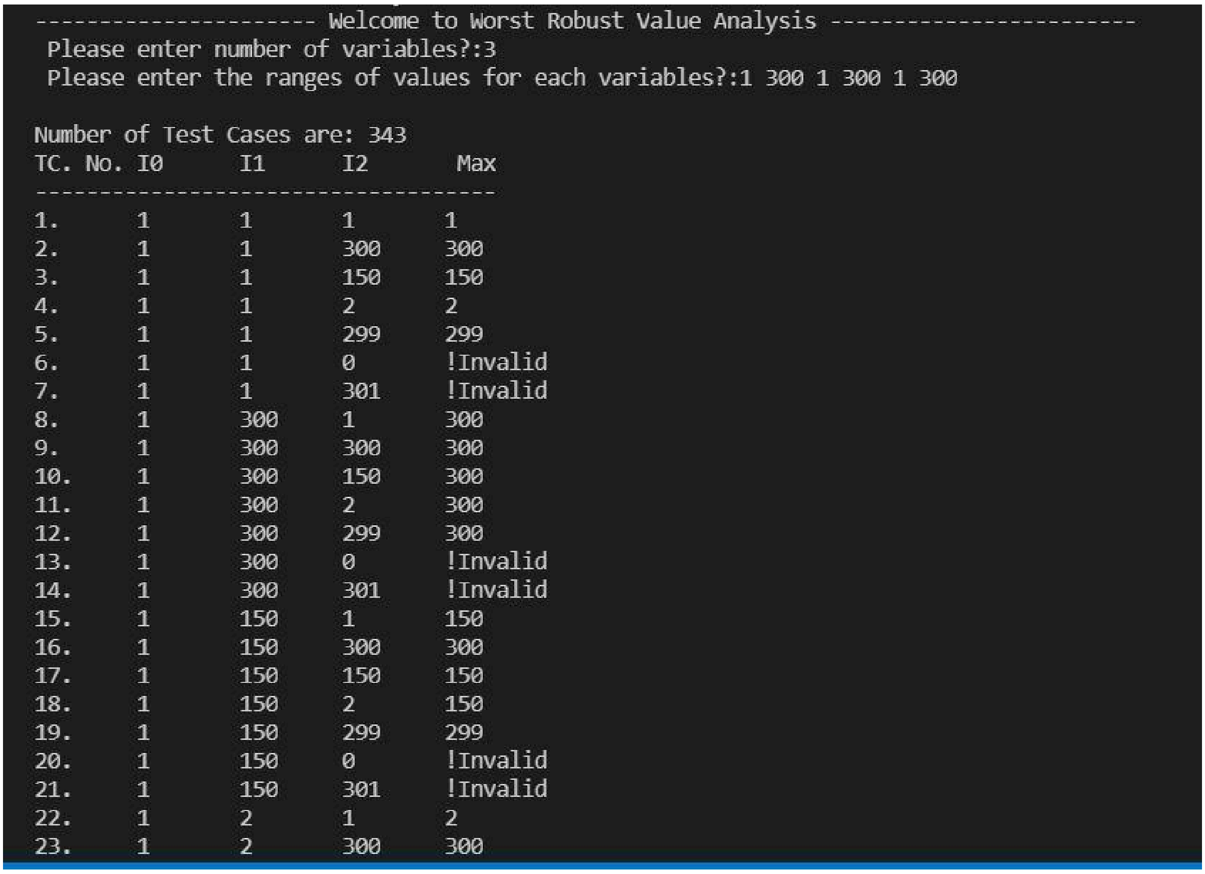
Program 4

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

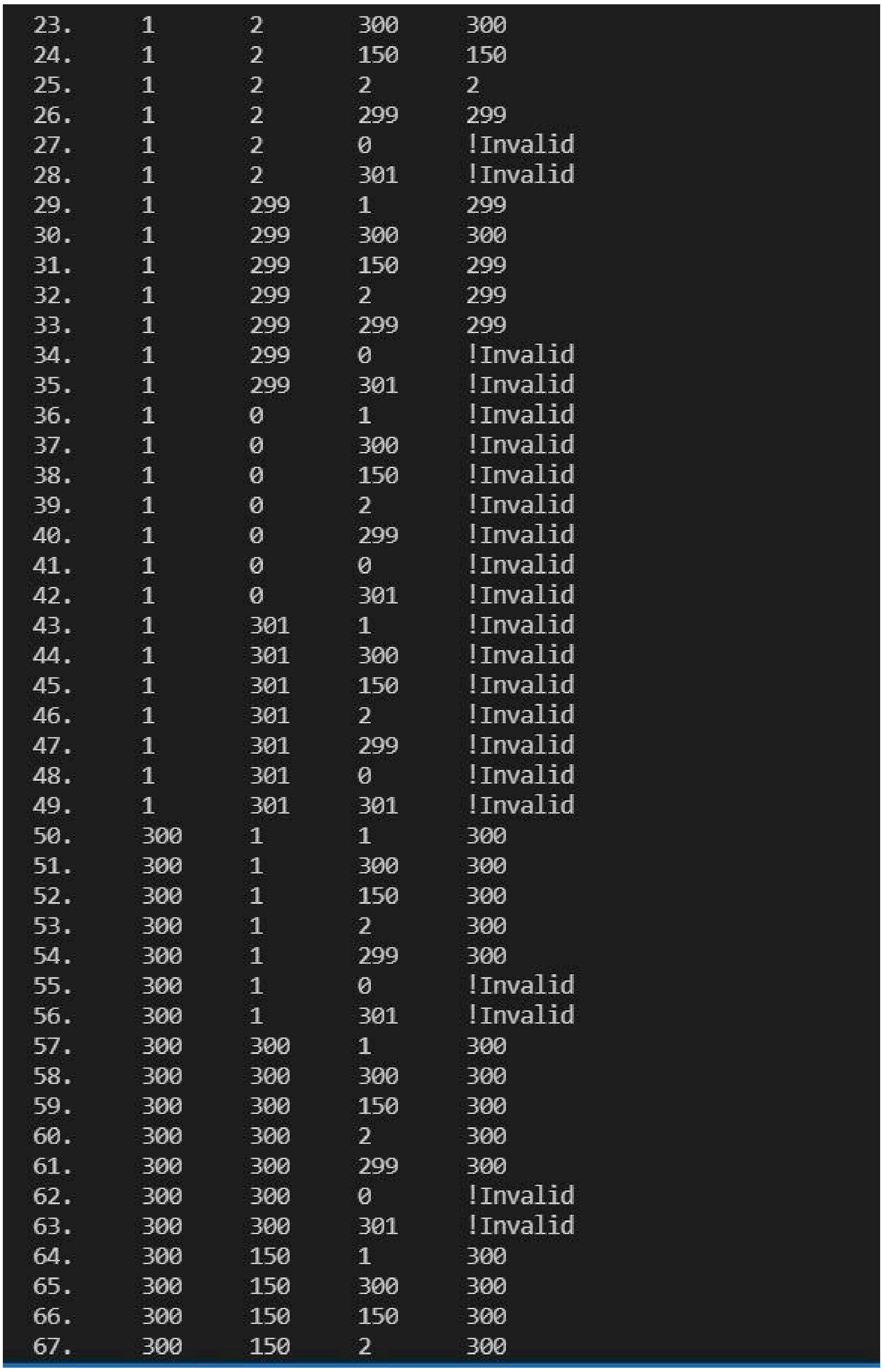
Source Code: Worst Robust Approach

```
#include <iostream>
#include <cmath> #include
<string>

using namespace std;

int maximum(int x, int y){
        int max = x;
        if (x < y)
                max = y;
        return max;
}

int main(){
        int n, x, y, z;
        cout << "--------------------- Welcome to Worst Robust Value Analysis____\n ";
        cout<< "Please enter number of variables?:"; cin >> n;
        int values[n][7]; // 0- min value, 1- max value, 2-nominal value, 3- Just above the
minimum, 4- just below the maximum
        cout<< " Please enter the ranges of values for each variables?:";
        //cout<<"Please Enter min and max ranges. For example: 100-200 \n";
        for (int i = 0; i < n; i++){
                cin >> values[i][0] >> values[i][1];
                values[i][2] = (values[i][0] + values[i][1]) / 2; values[i][3] = values[i][0]
                + 1;
                values[i][4] = values[i][1] - 1;
                values[i][5] = values[i][0] - 1;
                values[i][6] = values[i][1] + 1;
        }

        int num_of_cols = pow(7, n);

        cout << "\nNumber of Test Cases are: " << num_of_cols << "\nTC. No.\t";

        for (int i = 0; i < n; i++) cout <<"I" <<
                i<<"\t";
        cout <<" Max";
        cout << "\n______________________________________________________" << endl;
        int r = 0, index_of_array = 1;

        for (int i = 0; i < 7; i++){
                for (int j = 0; j < 7; j++)
                {
                        for (int k = 0; k < 7; k++)
                        {
                                cout << index_of_array << ".\t" << values[0][i] << "\t" <<
values[1][j] << "\t" << values[2][k]
<< "\t";
                                if (i > 4 || j > 4 || k > 4) cout <<
                                        "!Invalid\n";
                                else
                                        cout << maximum(maximum(values[0][i],
values[1][j]), values[2][k]) << "\t" << endl;
                                index_of_array++;
                        }
                }
        }
        return 0;
}
```

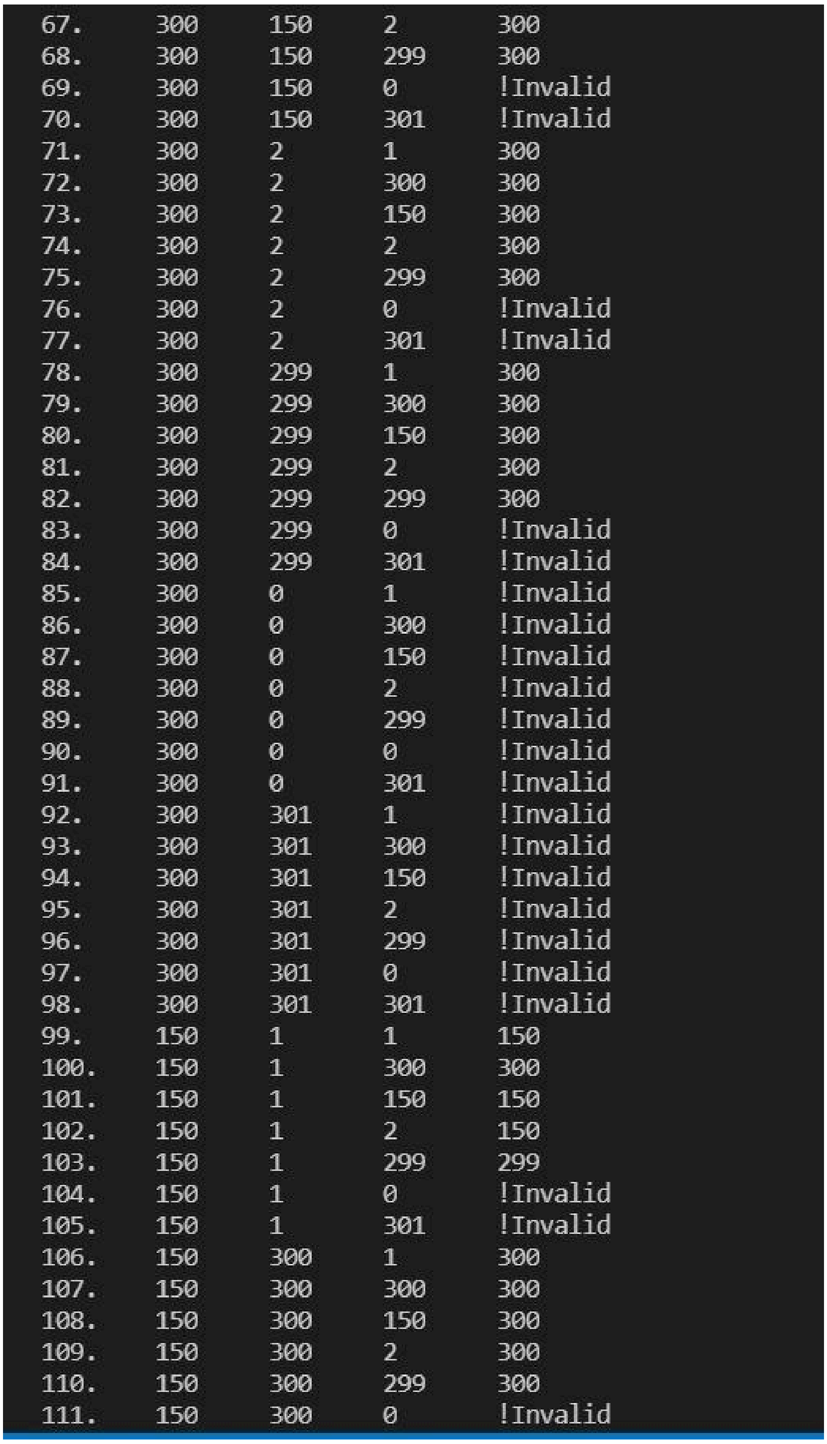
Output:



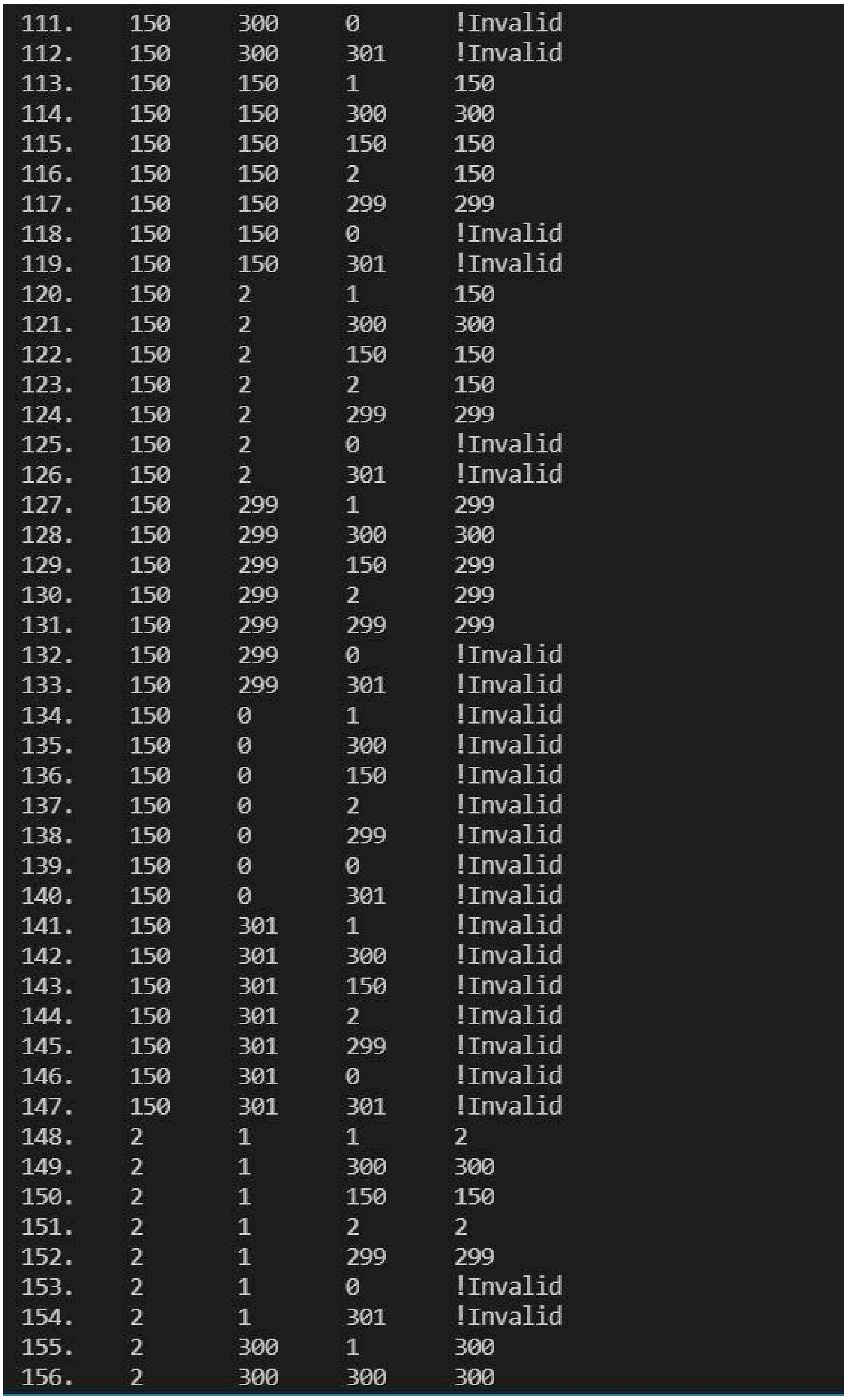
```
---------------------- Welcome to Worst Robust Value Analysis ------------------------
 Please enter number of variables?:3
 Please enter the ranges of values for each variables?:1 300 1 300 1 300

Number of Test Cases are: 343
TC. No. I0      I1      I2       Max
------------------------------------
1.      1       1       1       1
2.      1       1       300     300
3.      1       1       150     150
4.      1       1       2       2
5.      1       1       299     299
6.      1       1       0       !Invalid
7.      1       1       301     !Invalid
8.      1       300     1       300
9.      1       300     300     300
10.     1       300     150     300
11.     1       300     2       300
12.     1       300     299     300
13.     1       300     0       !Invalid
14.     1       300     301     !Invalid
15.     1       150     1       150
16.     1       150     300     300
17.     1       150     150     150
18.     1       150     2       150
19.     1       150     299     299
20.     1       150     0       !Invalid
21.     1       150     301     !Invalid
22.     1       2       1       2
23.     1       2       300     300
```



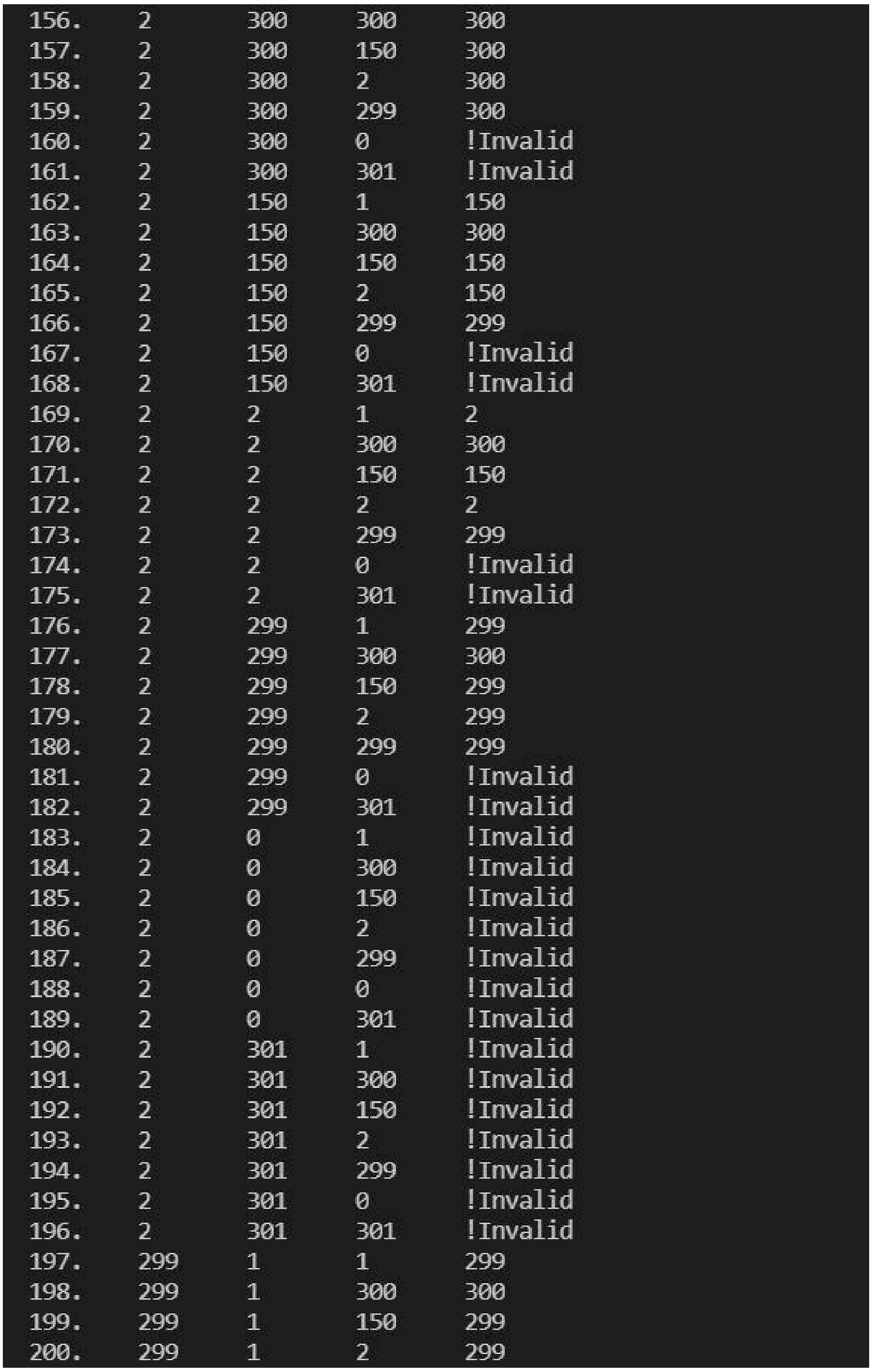
```
  23.     1       2       300     300
  24.     1       2       150     150
  25.     1       2       2       2
  26.     1       2       299     299
  27.     1       2       0       !Invalid
  28.     1       2       301     !Invalid
  29.     1       299     1       299
  30.     1       299     300     300
  31.     1       299     150     299
  32.     1       299     2       299
  33.     1       299     299     299
  34.     1       299     0       !Invalid
  35.     1       299     301     !Invalid
  36.     1       0       1       !Invalid
  37.     1       0       300     !Invalid
  38.     1       0       150     !Invalid
  39.     1       0       2       !Invalid
  40.     1       0       299     !Invalid
  41.     1       0       0       !Invalid
  42.     1       0       301     !Invalid
  43.     1       301     1       !Invalid
  44.     1       301     300     !Invalid
  45.     1       301     150     !Invalid
  46.     1       301     2       !Invalid
  47.     1       301     299     !Invalid
  48.     1       301     0       !Invalid
  49.     1       301     301     !Invalid
  50.     300     1       1       300
  51.     300     1       300     300
  52.     300     1       150     300
  53.     300     1       2       300
  54.     300     1       299     300
  55.     300     1       0       !Invalid
  56.     300     1       301     !Invalid
  57.     300     300     1       300
  58.     300     300     300     300
  59.     300     300     150     300
  60.     300     300     2       300
  61.     300     300     299     300
  62.     300     300     0       !Invalid
  63.     300     300     301     !Invalid
  64.     300     150     1       300
  65.     300     150     300     300
  66.     300     150     150     300
  67.     300     150     2       300
```



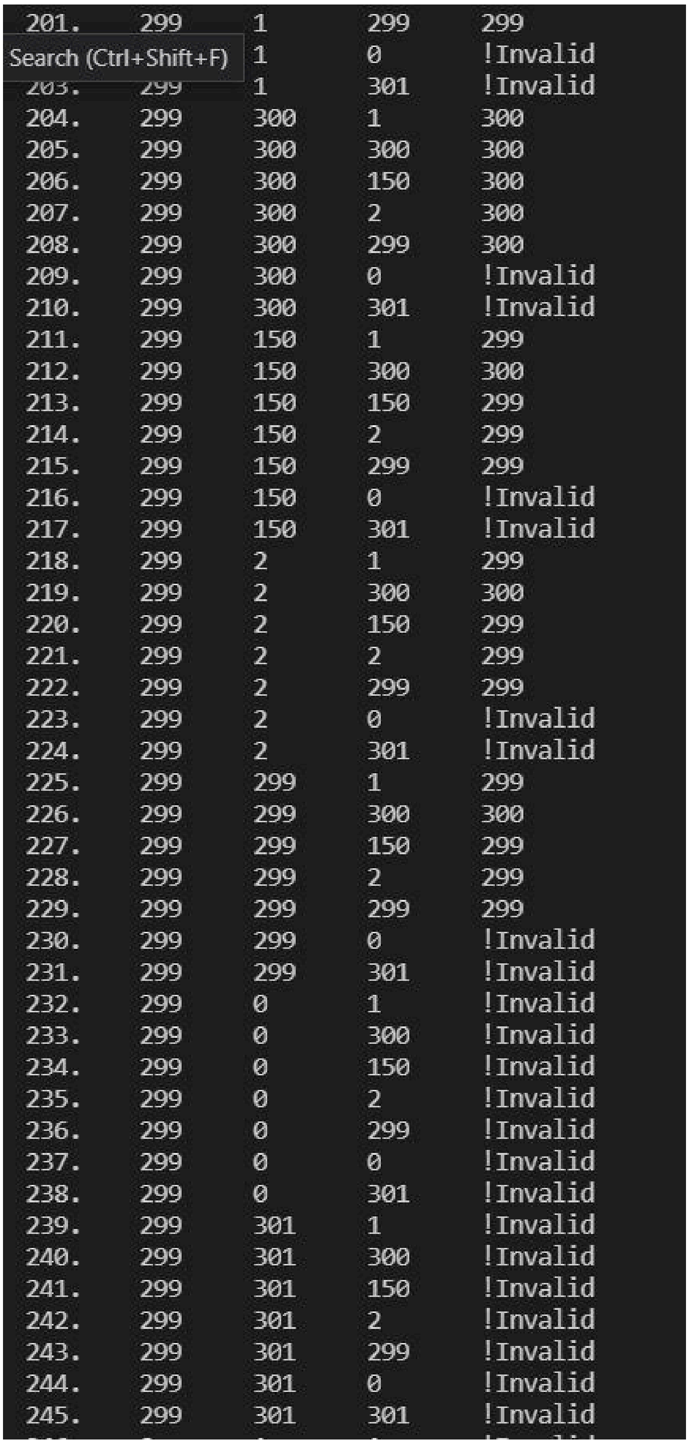
```
67.     300     150     2       300
68.     300     150     299     300
69.     300     150     0       !Invalid
70.     300     150     301     !Invalid
71.     300     2       1       300
72.     300     2       300     300
73.     300     2       150     300
74.     300     2       2       300
75.     300     2       299     300
76.     300     2       0       !Invalid
77.     300     2       301     !Invalid
78.     300     299     1       300
79.     300     299     300     300
80.     300     299     150     300
81.     300     299     2       300
82.     300     299     299     300
83.     300     299     0       !Invalid
84.     300     299     301     !Invalid
85.     300     0       1       !Invalid
86.     300     0       300     !Invalid
87.     300     0       150     !Invalid
88.     300     0       2       !Invalid
89.     300     0       299     !Invalid
90.     300     0       0       !Invalid
91.     300     0       301     !Invalid
92.     300     301     1       !Invalid
93.     300     301     300     !Invalid
94.     300     301     150     !Invalid
95.     300     301     2       !Invalid
96.     300     301     299     !Invalid
97.     300     301     0       !Invalid
98.     300     301     301     !Invalid
99.     150     1       1       150
100.    150     1       300     300
101.    150     1       150     150
102.    150     1       2       150
103.    150     1       299     299
104.    150     1       0       !Invalid
105.    150     1       301     !Invalid
106.    150     300     1       300
107.    150     300     300     300
108.    150     300     150     300
109.    150     300     2       300
110.    150     300     299     300
111.    150     300     0       !Invalid
```



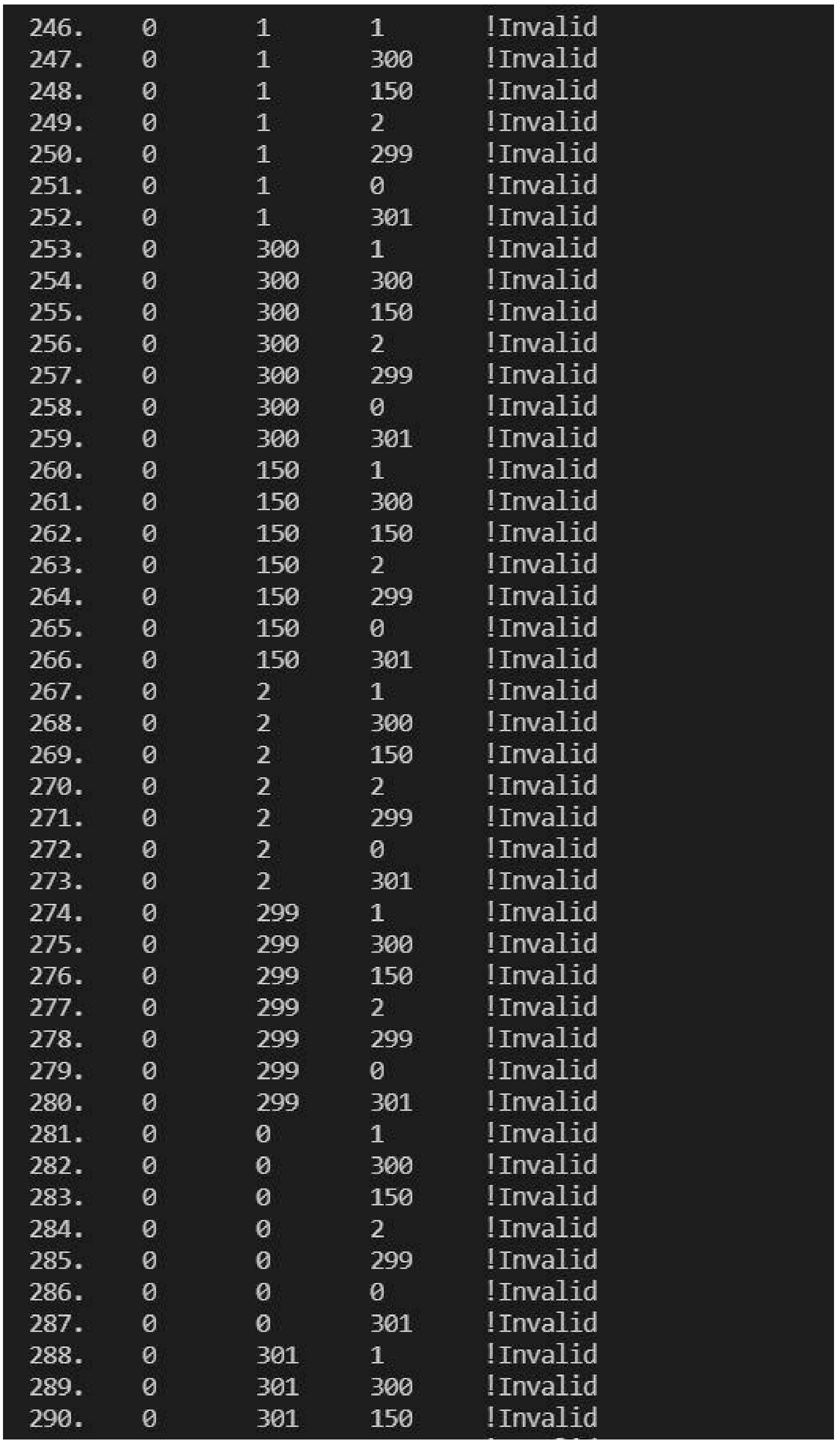
```
111.    150     300     0       !Invalid
112.    150     300     301     !Invalid
113.    150     150     1       150
114.    150     150     300     300
115.    150     150     150     150
116.    150     150     2       150
117.    150     150     299     299
118.    150     150     0       !Invalid
119.    150     150     301     !Invalid
120.    150     2       1       150
121.    150     2       300     300
122.    150     2       150     150
123.    150     2       2       150
124.    150     2       299     299
125.    150     2       0       !Invalid
126.    150     2       301     !Invalid
127.    150     299     1       299
128.    150     299     300     300
129.    150     299     150     299
130.    150     299     2       299
131.    150     299     299     299
132.    150     299     0       !Invalid
133.    150     299     301     !Invalid
134.    150     0       1       !Invalid
135.    150     0       300     !Invalid
136.    150     0       150     !Invalid
137.    150     0       2       !Invalid
138.    150     0       299     !Invalid
139.    150     0       0       !Invalid
140.    150     0       301     !Invalid
141.    150     301     1       !Invalid
142.    150     301     300     !Invalid
143.    150     301     150     !Invalid
144.    150     301     2       !Invalid
145.    150     301     299     !Invalid
146.    150     301     0       !Invalid
147.    150     301     301     !Invalid
148.    2       1       1       2
149.    2       1       300     300
150.    2       1       150     150
151.    2       1       2       2
152.    2       1       299     299
153.    2       1       0       !Invalid
154.    2       1       301     !Invalid
155.    2       300     1       300
156.    2       300     300     300
```



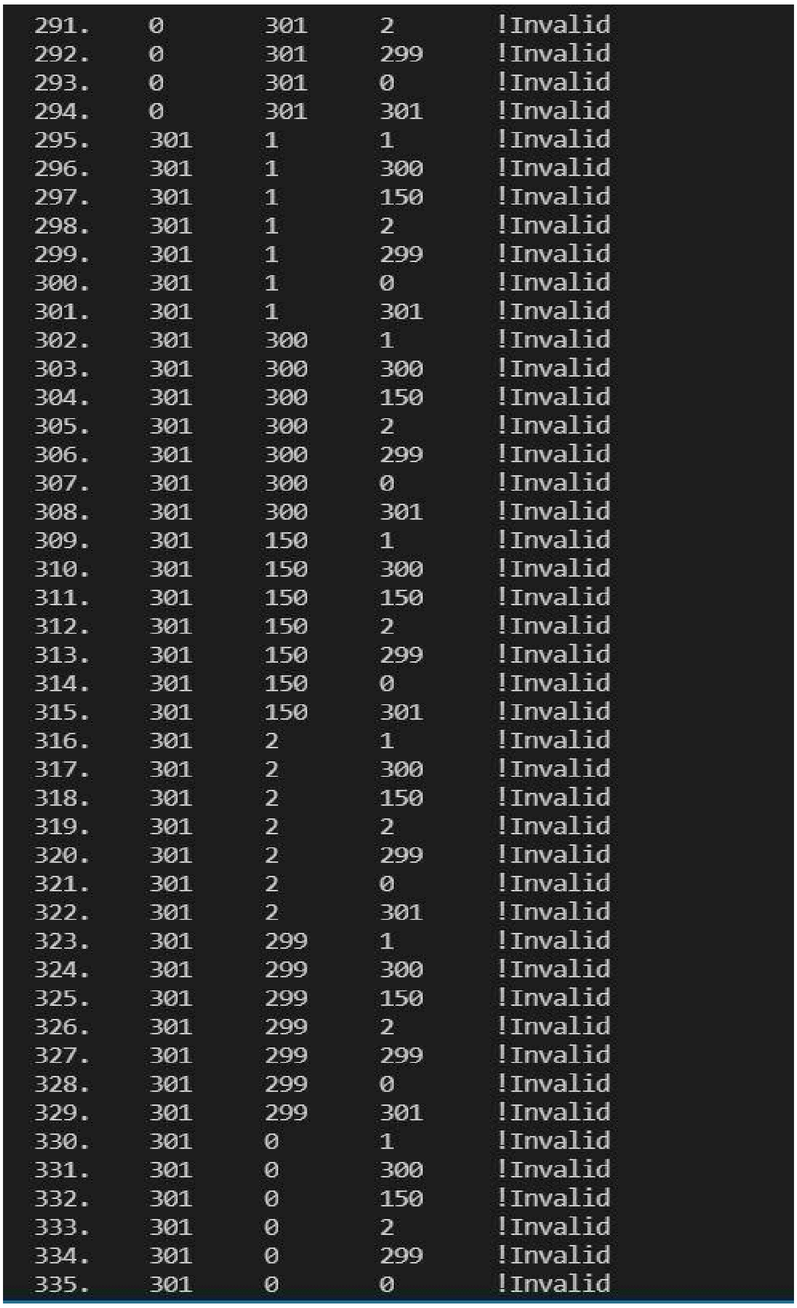
```
156.    2       300     300     300
157.    2       300     150     300
158.    2       300     2       300
159.    2       300     299     300
160.    2       300     0       !Invalid
161.    2       300     301     !Invalid
162.    2       150     1       150
163.    2       150     300     300
164.    2       150     150     150
165.    2       150     2       150
166.    2       150     299     299
167.    2       150     0       !Invalid
168.    2       150     301     !Invalid
169.    2       2       1       2
170.    2       2       300     300
171.    2       2       150     150
172.    2       2       2       2
173.    2       2       299     299
174.    2       2       0       !Invalid
175.    2       2       301     !Invalid
176.    2       299     1       299
177.    2       299     300     300
178.    2       299     150     299
179.    2       299     2       299
180.    2       299     299     299
181.    2       299     0       !Invalid
182.    2       299     301     !Invalid
183.    2       0       1       !Invalid
184.    2       0       300     !Invalid
185.    2       0       150     !Invalid
186.    2       0       2       !Invalid
187.    2       0       299     !Invalid
188.    2       0       0       !Invalid
189.    2       0       301     !Invalid
190.    2       301     1       !Invalid
191.    2       301     300     !Invalid
192.    2       301     150     !Invalid
193.    2       301     2       !Invalid
194.    2       301     299     !Invalid
195.    2       301     0       !Invalid
196.    2       301     301     !Invalid
197.    299     1       1       299
198.    299     1       300     300
199.    299     1       150     299
200.    299     1       2       299
```



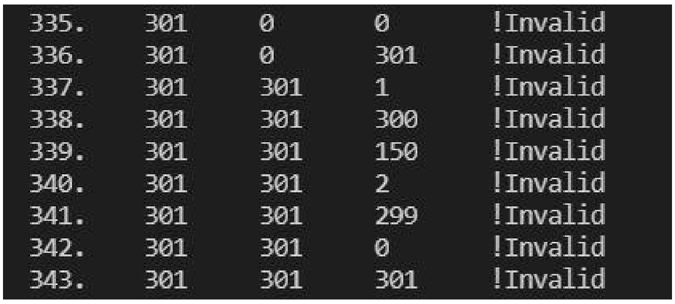
```
 201.    299    1      299    299
Search (Ctrl+Shift+F)   1      0      !Invalid
 203.    299    1      301    !Invalid
 204.    299    300    1      300
 205.    299    300    300    300
 206.    299    300    150    300
 207.    299    300    2      300
 208.    299    300    299    300
 209.    299    300    0      !Invalid
 210.    299    300    301    !Invalid
 211.    299    150    1      299
 212.    299    150    300    300
 213.    299    150    150    299
 214.    299    150    2      299
 215.    299    150    299    299
 216.    299    150    0      !Invalid
 217.    299    150    301    !Invalid
 218.    299    2      1      299
 219.    299    2      300    300
 220.    299    2      150    299
 221.    299    2      2      299
 222.    299    2      299    299
 223.    299    2      0      !Invalid
 224.    299    2      301    !Invalid
 225.    299    299    1      299
 226.    299    299    300    300
 227.    299    299    150    299
 228.    299    299    2      299
 229.    299    299    299    299
 230.    299    299    0      !Invalid
 231.    299    299    301    !Invalid
 232.    299    0      1      !Invalid
 233.    299    0      300    !Invalid
 234.    299    0      150    !Invalid
 235.    299    0      2      !Invalid
 236.    299    0      299    !Invalid
 237.    299    0      0      !Invalid
 238.    299    0      301    !Invalid
 239.    299    301    1      !Invalid
 240.    299    301    300    !Invalid
 241.    299    301    150    !Invalid
 242.    299    301    2      !Invalid
 243.    299    301    299    !Invalid
 244.    299    301    0      !Invalid
 245.    299    301    301    !Invalid
```



```
246.    0       1       1       !Invalid
247.    0       1       300     !Invalid
248.    0       1       150     !Invalid
249.    0       1       2       !Invalid
250.    0       1       299     !Invalid
251.    0       1       0       !Invalid
252.    0       1       301     !Invalid
253.    0       300     1       !Invalid
254.    0       300     300     !Invalid
255.    0       300     150     !Invalid
256.    0       300     2       !Invalid
257.    0       300     299     !Invalid
258.    0       300     0       !Invalid
259.    0       300     301     !Invalid
260.    0       150     1       !Invalid
261.    0       150     300     !Invalid
262.    0       150     150     !Invalid
263.    0       150     2       !Invalid
264.    0       150     299     !Invalid
265.    0       150     0       !Invalid
266.    0       150     301     !Invalid
267.    0       2       1       !Invalid
268.    0       2       300     !Invalid
269.    0       2       150     !Invalid
270.    0       2       2       !Invalid
271.    0       2       299     !Invalid
272.    0       2       0       !Invalid
273.    0       2       301     !Invalid
274.    0       299     1       !Invalid
275.    0       299     300     !Invalid
276.    0       299     150     !Invalid
277.    0       299     2       !Invalid
278.    0       299     299     !Invalid
279.    0       299     0       !Invalid
280.    0       299     301     !Invalid
281.    0       0       1       !Invalid
282.    0       0       300     !Invalid
283.    0       0       150     !Invalid
284.    0       0       2       !Invalid
285.    0       0       299     !Invalid
286.    0       0       0       !Invalid
287.    0       0       301     !Invalid
288.    0       301     1       !Invalid
289.    0       301     300     !Invalid
290.    0       301     150     !Invalid
```



```
291.    0       301     2       !Invalid
292.    0       301     299     !Invalid
293.    0       301     0       !Invalid
294.    0       301     301     !Invalid
295.    301     1       1       !Invalid
296.    301     1       300     !Invalid
297.    301     1       150     !Invalid
298.    301     1       2       !Invalid
299.    301     1       299     !Invalid
300.    301     1       0       !Invalid
301.    301     1       301     !Invalid
302.    301     300     1       !Invalid
303.    301     300     300     !Invalid
304.    301     300     150     !Invalid
305.    301     300     2       !Invalid
306.    301     300     299     !Invalid
307.    301     300     0       !Invalid
308.    301     300     301     !Invalid
309.    301     150     1       !Invalid
310.    301     150     300     !Invalid
311.    301     150     150     !Invalid
312.    301     150     2       !Invalid
313.    301     150     299     !Invalid
314.    301     150     0       !Invalid
315.    301     150     301     !Invalid
316.    301     2       1       !Invalid
317.    301     2       300     !Invalid
318.    301     2       150     !Invalid
319.    301     2       2       !Invalid
320.    301     2       299     !Invalid
321.    301     2       0       !Invalid
322.    301     2       301     !Invalid
323.    301     299     1       !Invalid
324.    301     299     300     !Invalid
325.    301     299     150     !Invalid
326.    301     299     2       !Invalid
327.    301     299     299     !Invalid
328.    301     299     0       !Invalid
329.    301     299     301     !Invalid
330.    301     0       1       !Invalid
331.    301     0       300     !Invalid
332.    301     0       150     !Invalid
333.    301     0       2       !Invalid
334.    301     0       299     !Invalid
335.    301     0       0       !Invalid
```



```
335.    301     0       0       !Invalid
336.    301     0       301     !Invalid
337.    301     301     1       !Invalid
338.    301     301     300     !Invalid
339.    301     301     150     !Invalid
340.    301     301     2       !Invalid
341.    301     301     299     !Invalid
342.    301     301     0       !Invalid
343.    301     301     301     !Invalid
```

Learning Outcome:

1. Successfully completed Worst Robust Approach.
2. Worst Robust Approach in maximum of three numbers.

Program 5

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

Source Code: Equivalence Class

```
Testing #include <iostream>
#include
<stdlib.h>
#include
<conio.h>
#include <ctime>

using namespace std;

/*Function to check the type of the triangle
Input: 3 vertices
Output:
-1 - Inavlid input
0 - Equilateral
triangle 1 - Isoscles
triangle
2 - Scalene
triangle 3 - Not a
triangle
*/
int triangle(int a[3], int min[3], int max[3]){
   for (int i = 0; i < 3; i++)
      if (a[i] < min[i] || a[i] > max[i])
         return -1;
   if (a[0] + a[1] > a[2] && a[1] + a[2] > a[0] && a[2] + a[0] > a[1]){
      if (a[0] == a[1] && a[1] == a[2] && a[2] == a[0]) return
      0; else if (a[0] == a[1] || a[1] == a[2] || a[2] == a[0])
      return 1; else return 2;
   }
   else
      return 3;
}
string input_class[20], output_class[4];
void generateClasses(){
   input_class[0] = "{x :
   x<xmin}"; input_class[1] =
   "{x : x>xmax}";
   input_class[2] = "{x :
   xmin<x<xmax}"; input_class[3] =
   "{y : y<ymin}"; input_class[4] = "{y
   : y>ymax}"; input_class[5] = "{y :
   ymin<y<ymax}";
```

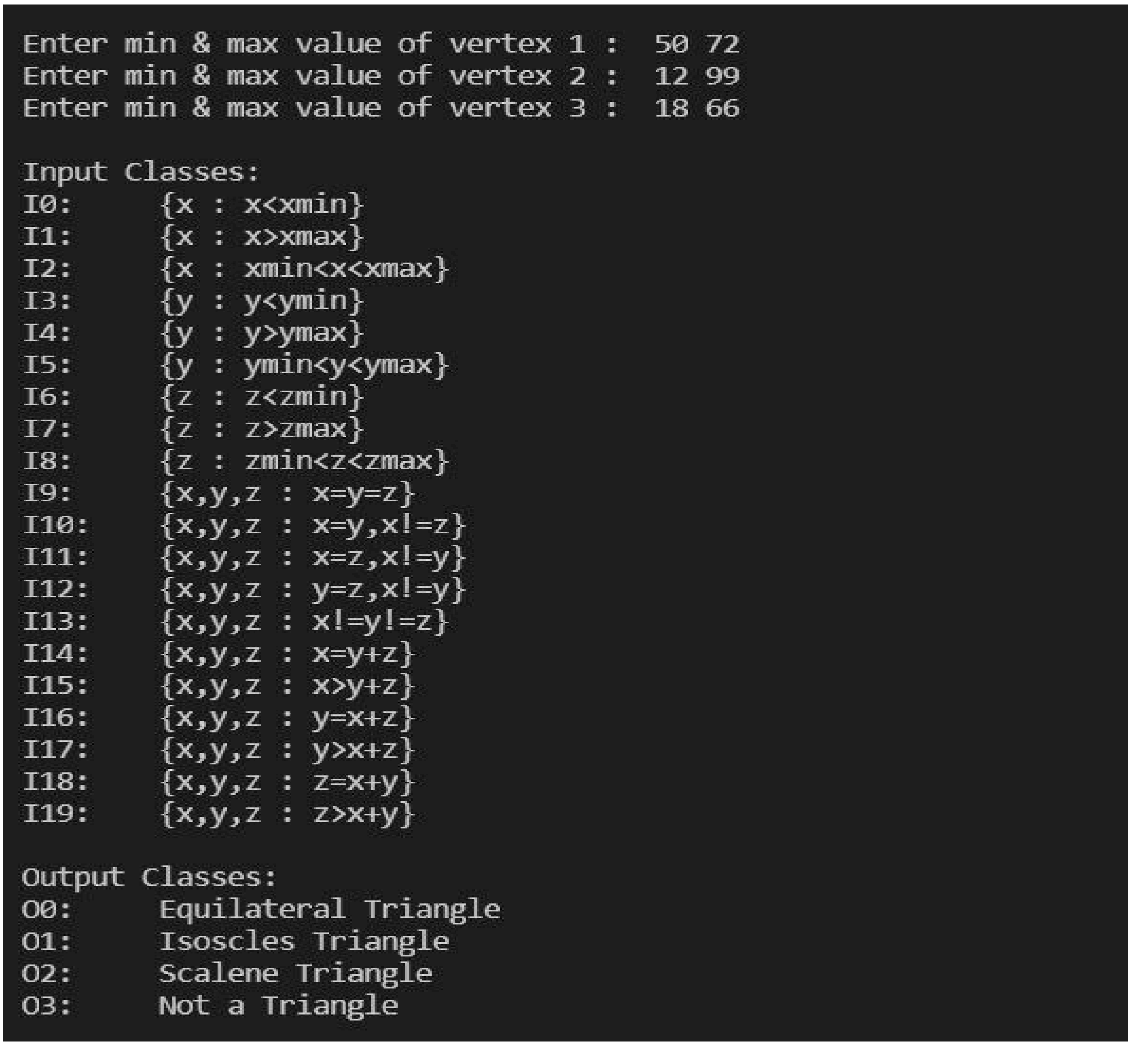
```
  input_class[6] = "{z : z<zmin}";
  input_class[7] = "{z : z>zmax}";
  input_class[8] = "{z :
  zmin<z<zmax}"; input_class[9] =
  "{x,y,z : x=y=z}";
  input_class[10] = "{x,y,z : x=y,x!=z}";
  input_class[11] = "{x,y,z : x=z,x!=y}";
  input_class[12] = "{x,y,z : y=z,x!=y}";
  input_class[13] = "{x,y,z : x!=y!=z}";
  input_class[14] = "{x,y,z : x=y+z}";
  input_class[15] = "{x,y,z : x>y+z}";
  input_class[16] = "{x,y,z : y=x+z}";
  input_class[17] = "{x,y,z : y>x+z}";
  input_class[18] = "{x,y,z : z=x+y}";
  input_class[19] = "{x,y,z : z>x+y}";
  output_class[0] = "Equilateral
  Triangle"; output_class[1] = "Isoscles
  Triangle"; output_class[2] = "Scalene
  Triangle"; output_class[3] = "Not a
  Triangle";
}
// To determine test cases in equivalence Testing
void equivalenceTesting(int min[3], int max[3]){
  int expected_output[24], test_cases[24][3], i = 0, x = 0, y = 0, z =
  0; generateClasses();
  cout<< "\n\nInput Classes:";
  for (i = 0; i < 20; i++)
    cout << "\nI" << i << ":\t" <<
  input_class[i]; cout << "\n\nOutput
  Classes:";
  for (i = 0; i < 4; i++)
    cout << "\nO"<< i << ":\t" << output_class[i];
    //for input equivalance class
    for (i = 0; i < 20; i++){
    switch (i)
    {
    case 0:
      x = min[0] - 1;
      y = (min[1] + max[1]) / 2;
      z = (min[2] + max[2]) / 2;
      break
    ; case 1:
      x = max[0] + 1;
      y = (min[1] + max[1]) / 2;
      z = (min[2] + max[2]) / 2;
      break
    ; case 2:
      x = (min[0] + max[0]) / 2;
      y = (min[1] + max[1]) / 2;
      z = (min[2] + max[2]) / 2;
```

```
   break
; case 3:
   y = min[1] - 1;
   x = (min[0] + max[0]) / 2;
   z = (min[2] + max[2]) / 2;
   break
; case 4:
   y = max[1] + 1;
   x = (min[0] + max[0]) / 2;
   z = (min[2] + max[2]) / 2;
   break
; case 5:
   y = (min[1] + max[1]) / 2;
   x = (min[0] + max[0]) / 2;
   z = (min[2] + max[2]) / 2;
   break
; case 6:
   z = min[0] - 1;
   y = (min[1] + max[1]) / 2;
   x = (min[0] + max[0]) / 2;
   break
; case 7:
   z = max[0] + 1;
   y = (min[1] + max[1]) / 2;
   x = (min[0] + max[0]) / 2;
   break
; case 8:
   z = (min[2] + max[2]) / 2;
   y = (min[1] + max[1]) / 2;
   x = (min[0] + max[0]) / 2;
   break
; case 9:
   x = y = z = 60;
   break
; case
10:
   x = y = 60;
   z = (min[2] + max[2]) / 2;
   break
; case
11:
   x = z = 60;
   y = (min[1] + max[1]) / 2;
   break
; case
12:
   y = z = 60;
   z = (min[0] + max[0]) / 2;
   break
; case
13:
      x = 40;
      y = 60;
      z = 80;
       break
        ; case
            14:
       y = (min[1] + max[1]) / 2;
       z = (min[2] + max[2]) /
       2; x = y + z;
       break
   ; case
   15:
       y = (min[1] + max[1]) / 2;
       z = (min[2] + max[2]) /
       2; x = y + z + 1;
       break
   ; case
   16:
       x = (min[0] + max[0]) / 2;
       z = (min[2] + max[2]) /
       2; y = x + z;
       break
   ; case
   17:
       x = (min[0] + max[0]) / 2;
       z = (min[2] + max[2]) /
       2; y = x + z + 1;
       break
   ; case
   18:
       y = (min[1] + max[1]) / 2;
       x = (min[0] + max[0]) /
       2; z = x + y;
       break
   ; case
   19:
       y = (min[1] + max[1]) / 2;
       x = (min[0] + max[0]) /
       2; z = x + y + 1;
       break;
   }
   test_cases[i][0]    =
   x;   test_cases[i][1]
   =                      y;
   test_cases[i][2]    =
   z;
}
//for output equivalance class
for (; i < 24; i++)
{
   switch (i - 20)
   {
   case 0:
       x = y = z = 60;
```

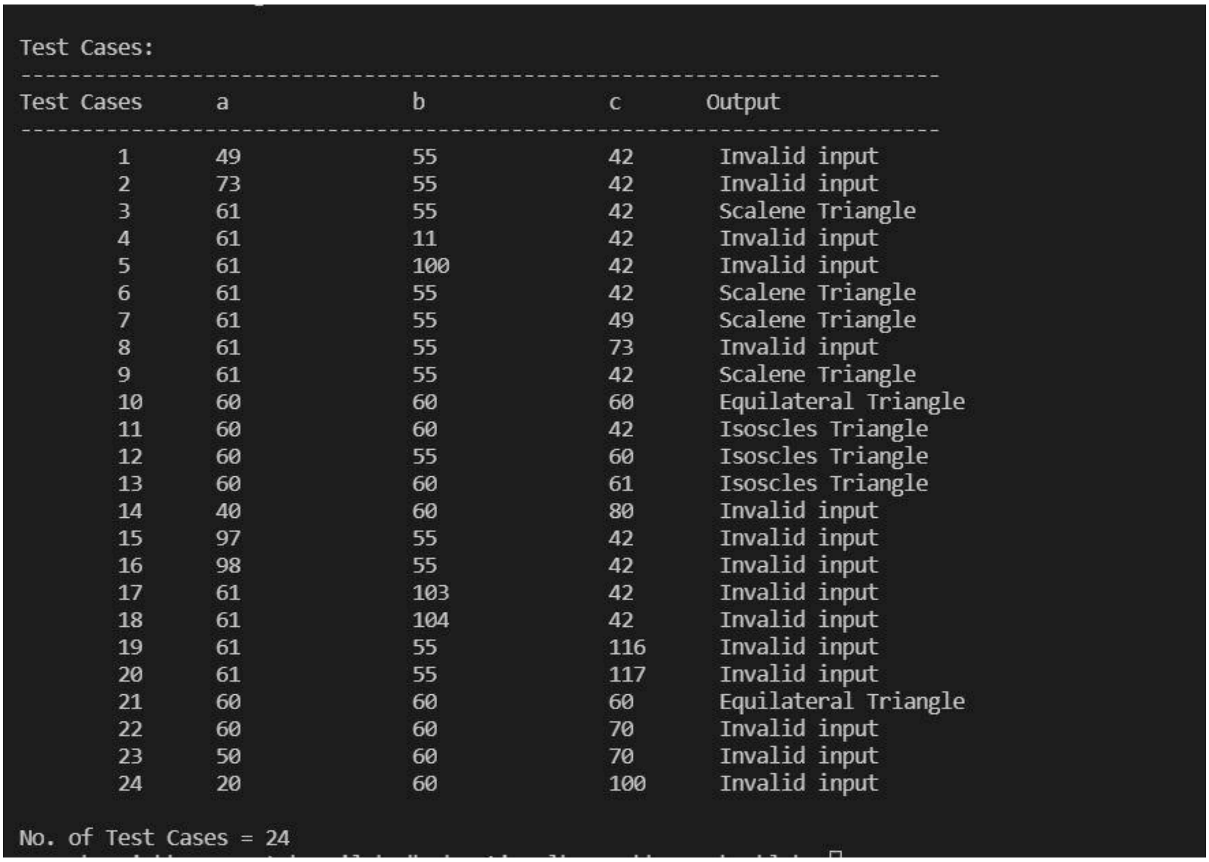
```
    break;
    case 1:
      x = y =
      60; z =
      70;
      break
    ; case 2:
      x = 50;
      y = 60;
      z = 70;
      break
    ; case 3:
      x = 20;
      y = 60;
      z = 100;
      break;
    }
    test_cases[i][0]    =
    x;   test_cases[i][1]
    =                    y;
    test_cases[i][2]    =
    z;
  }
  cout << "\n\nTest Cases:";
  cout<<"\n__________________________________________________";
  //cout<<"\n|\t\t Input\t\t\t| Expected\t\t|";
  cout << "\nTest Cases\ta\t\tb\t\tc\tOutput\t\t\n";
  cout<<"__________________________________________________\n";
  for (int i = 0; i < 24;
    i++){ cout << "\t" << i
    + 1; for (int j = 0; j <
    3; j++){

      cout << "\t" << test_cases[i][j] << "\t";
    }
    int expected_output = triangle(test_cases[i], min, max);
    if (expected_output < 0)
      cout << " Invalid
      input\t";
    else
      cout << " " << output_class[expected_output];
    cout << "\t" << endl;
  }
  cout << "\nNo. of Test Cases = " << sizeof(input_class) / sizeof(input_class[0]) +
sizeof(output_class) / sizeof(output_class[0])<<endl;
}

int main()
{
  int min[3],
  max[3];
  srand(time(NULL)
  );
  for (int i = 0; i < 3; i++){
    cout<<"\nEnter min & max value of vertex " << i + 1 << " : ";
```

```
        min[i] = rand() % 100;
        cout << " " << min[i];
        max[i] = rand() % 100;
        while (min[i] >=
        max[i]){
            max[i] = rand() % 100;
        }
        cout << " " << max[i];
    }
    equivalenceTesting(min, max);
    return 0;
}
```

Output:



```
Enter min & max value of vertex 1 :  50 72
Enter min & max value of vertex 2 :  12 99
Enter min & max value of vertex 3 :  18 66

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:     Isoscles Triangle
O2:     Scalene Triangle
O3:     Not a Triangle
```



```
Test Cases:
---------------------------------------------------------------------------
Test Cases      a               b               c       Output
---------------------------------------------------------------------------
        1       49              55              42       Invalid input
        2       73              55              42       Invalid input
        3       61              55              42       Scalene Triangle
        4       61              11              42       Invalid input
        5       61              100             42       Invalid input
        6       61              55              42       Scalene Triangle
        7       61              55              49       Scalene Triangle
        8       61              55              73       Invalid input
        9       61              55              42       Scalene Triangle
        10      60              60              60       Equilateral Triangle
        11      60              60              42       Isoscles Triangle
        12      60              55              60       Isoscles Triangle
        13      60              60              61       Isoscles Triangle
        14      40              60              80       Invalid input
        15      97              55              42       Invalid input
        16      98              55              42       Invalid input
        17      61              103             42       Invalid input
        18      61              104             42       Invalid input
        19      61              55              116      Invalid input
        20      61              55              117      Invalid input
        21      60              60              60       Equilateral Triangle
        22      60              60              70       Invalid input
        23      50              60              70       Invalid input
        24      20              60              100      Invalid input

No. of Test Cases = 24
```

Learning Outcome:

1. Successfully completed Equivalence Class Testing.
2. Equivalence Class Testing in type of the triangle.

Program 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 Decision Table Testing.

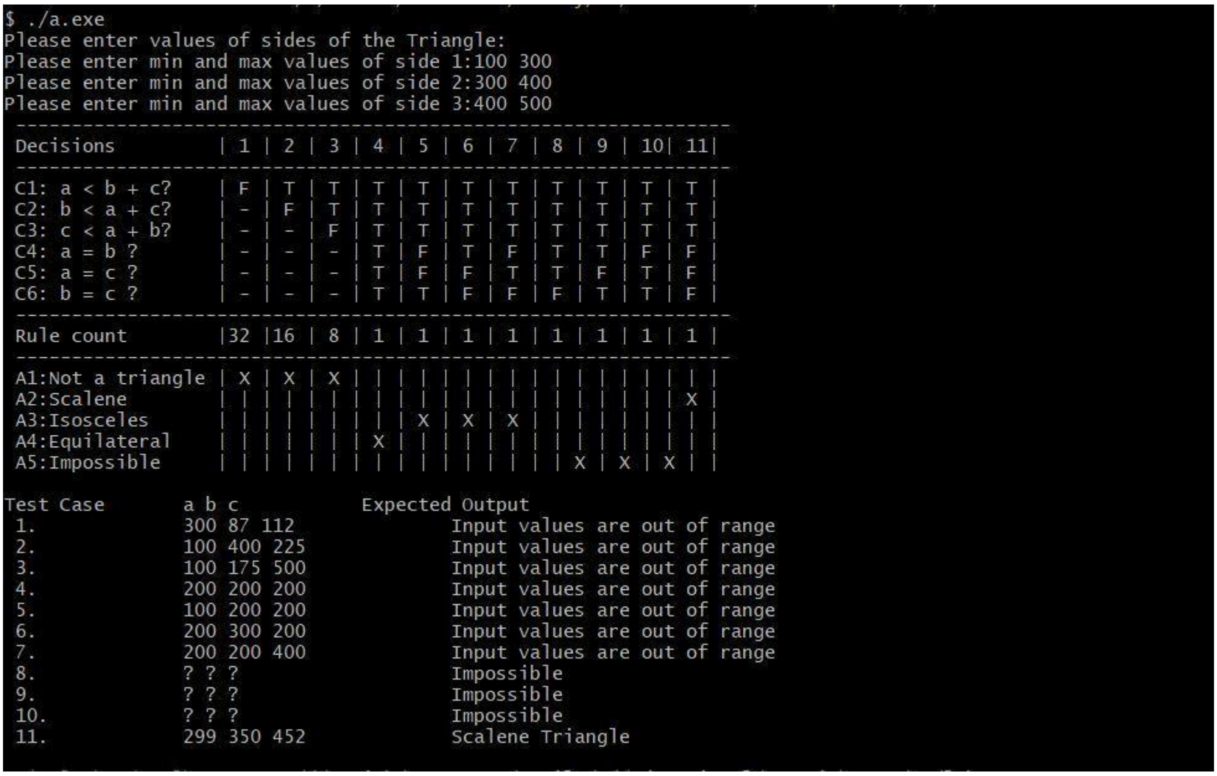
Source Code: Decision Table

```
Testing #include <iostream>
using namespace std;
int values[7][3];
int a, b, c;

string DecisionTable()
{
  string table = "____________________________________________\n";
  table = table + " Decisions       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10| 11|\n"; table
  = table + "_______________________________________________\n";
  table = table + " C1: a < b + c?                                        | F | T | T
  | T | T | T | T | T | T | T | T |\n"; table = table + " C2: b < a + c?            | - |
  F | T | T | T | T | T | T | T | T | T |\n"; table = table + " C3: c < a + b?
                               | - | - | F | T | T | T | T | T | T | T | T |\n"; table
  = table + " C4: a = b ?          | - | - | - | T | F | T | F | T | T | F | F |\n"; table
  = table + " C5: a = c ?          | - | - | - | T | F | F | T | T | F | T | F |\n"; table
  = table + " C6: b = c ?          | - | - | - | T | T | F | F | F | T | T | F |\n"; table
  = table + "_______________________________________________\n";
  table = table + " Rule count      |32 |16 | 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |\n"; table
  = table + "_______________________________________________\n";
  table = table + " A1:Not a triangle | X | X | X | | | | | | | | | | | | | | | | |\n"; table =
  table + " A2:Scalene              | | | | | | | | | | | | | | | | | | | | | X |\n"; table = table
  + " A3:Isosceles            | | | | | | | | | X | X | X | | | | | | | | |\n"; table =
  table + " A4:Equilateral           | | | | | | | X | | | | | | | | | | | | | | |\n"; table = table
  + " A5:Impossible             | | | | | | | | | | | | | | | | X | X | X | |\n"; return
  table;
}
string TriangleType(int a, int b, int c, int points[3][3])
{
  string res = "Not a Triangle";
  if (a < points[0][0] || a > points[0][1] || b < points[1][0] || b > points[1][1] || c < points[2][0]
|| c > points[2][1])
  {
    res = "Input values are out of range";
  }
  else if (a < b + c && b < a + c && c < a + b)
  {
    if (a == b && b == c)
    {
       res = "Equilateral Triangle";
    }
    else if (a == b || b == c || a == c)
    {
       res = "Isosceles Triangle";
    }
    else
    {
       res = "Scalene Triangle";
    }
  }
  return res;
}
int isTriangle(int index, int points[3][3])
{
  a = points[0][1];
  b = points[1][2];
  c = points[2][2];
  while (a < (b + c))
  {
    b = b /
    2; c = c /
    2;
  }
  values[0][0] = a;
  values[0][1] = b;
  values[0][2] =
  c; index++;
  a = points[0][2];
  b = points[1][1];
  c = points[2][2];
  while (b < (a + c))
  {
    a = a / 2;
    c = c / 2;
  }
  values[1][0] = a;
  values[1][1] = b;
  values[1][2] =
  c; index++;
  a = points[0][2];
  b = points[1][2];
  c = points[2][1];
  while (c < (a + b))
  {
    a = a / 2;
    b = b /
    2;
  }
```

```
 values[2][0] = a;
 values[2][1] = b;
 values[2][2] =
 c; index++;
 return index;
}
int Equilateral(int index, int points[3][3])
{
 a = points[0][3];
 b = points[1][3];
 c = points[2][3];
 int minMax = points[0][0], maximum_minimum = points[0][1], nominal_equation;
 for (int i = 0; i < 3; i++)
 {
  if (minMax >
   points[i][0]) minMax
   = points[i][0];
  if (maximum_minimum >
   points[i][1]) maximum_minimum
   = points[i][1];
 }
 nominal_equation  =  (minMax  +  maximum_minimum)  /
 2; values[index][0] = nominal_equation;
 values[index][1]                          =
 nominal_equation;    values[index][2]
 = nominal_equation; index++;
 return index;
}
int Isosceles(int index, int points[3][3])
{
 a = points[0][3];
 b = points[1][3];
 c = points[2][3];
 int minMax = points[0][0], maximum_minimum = points[0][1], nominal_equation;
 for (int i = 0; i < 3; i++)
 {
  if (minMax >
   points[i][0]) minMax
   = points[i][0];
  if (maximum_minimum >
   points[i][1]) maximum_minimum
   = points[i][1];
 }
 nominal_equation  =  (minMax  +  maximum_minimum)  /
 2; values[index][0] = points[0][0];
 values[index][1]                          =
 nominal_equation;    values[index][2]
 = nominal_equation; index++;
 values[index][0] =
 nominal_equation; values[index][1]
 = points[1][0]; values[index][2] =
 nominal_equation;
```

```
    index++;
    values[index][0] =
    nominal_equation; values[index][1]
    = nominal_equation;
    values[index][2] = points[2][0];
    index++;
    return index;
}
int impossible(int index)
{
    for (int i = 0; i < 3; i++)
    {
        cout << " " << index + i << ".\t\t? ? ?\t\t\t"<< "Impossible" << endl;
    }
    index = index + 3;
    return index;
}
void TestCases(int points[3][3])
{
    cout << "Test Case\ta b c\t\tExpected Output" << endl;
    int index = 0;
    index = isTriangle(index, points);
    index = Equilateral(index,
    points); index = Isosceles(index,
    points); index = 1;
    for (int i = 0; i < 7; i++)
    {
        a = values[i][0];
        b = values[i][1];
        c = values[i][2];
        cout << " " << index << ".\t\t" << a << " " << b << " " << c << "\t\t"<< TriangleType(a, b,
c, points) << endl;
        index++;
    }
    index = impossible(index);
    cout << " " << index<< ".\t\t" << points[0][1] - 1 << " " << points[1][2] << " " <<
    points[2][2]
+ 2 << "\t\t" << TriangleType(points[0][1] - 1, points[1][2], points[2][2] + 2, points) << endl;
}
int main()
{
    int points[3][3]; // 0-min, 1- max, 2- nominal
    cout << "Please enter values of sides of the Triangle:" << endl;
    for (int i = 0; i < 3; i++)
    {
        cout << "Range of side i.e Min and Max " << i + 1 <<
        ":"; cin >> points[i][0] >> points[i][1];
        points[i][2] = (points[i][0] + points[i][1]) / 2;
        points[i][3] = points[i][1] - points[i][0];
    }
    cout << DecisionTable() <<
    endl; TestCases(points);
    return 0;
}
```

Output:



```
$ ./a.exe
Please enter values of sides of the Triangle:
Please enter min and max values of side 1:100 300
Please enter min and max values of side 2:300 400
Please enter min and max values of side 3:400 500
 ----------------------------------------------------------------
 Decisions         | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10| 11|
 ----------------------------------------------------------------
 C1: a < b + c?    | F | T | T | T | T | T | T | T | T | T | T |
 C2: b < a + c?    | - | F | T | T | T | T | T | T | T | T | T |
 C3: c < a + b?    | - | - | F | T | T | T | T | T | T | T | T |
 C4: a = b ?       | - | - | - | T | F | T | F | T | T | F | F |
 C5: a = c ?       | - | - | - | T | F | F | T | T | F | T | F |
 C6: b = c ?       | - | - | - | T | T | F | F | F | T | T | F |
 ----------------------------------------------------------------
 Rule count        |32 |16 | 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
 ----------------------------------------------------------------
 A1:Not a triangle | X | X | X | | | | | | | | | | | | | | | | |
 A2:Scalene        | | | | | | | | | | | | | | | | | | | | | X |
 A3:Isosceles      | | | | | | | | | X | X | X | | | | | | | | |
 A4:Equilateral    | | | | | | | X | | | | | | | | | | | | | | |
 A5:Impossible     | | | | | | | | | | | | | | | | X | X | X | |

Test Case       a b c          Expected Output
 1.             300 87 112             Input values are out of range
 2.             100 400 225            Input values are out of range
 3.             100 175 500            Input values are out of range
 4.             200 200 200            Input values are out of range
 5.             100 200 200            Input values are out of range
 6.             200 300 200            Input values are out of range
 7.             200 200 400            Input values are out of range
 8.             ? ? ?                  Impossible
 9.             ? ? ?                  Impossible
 10.            ? ? ?                  Impossible
 11.            299 350 452            Scalene Triangle
```

Learning Outcome:

1. Successfully completed Decision Table Testing.
2. Decision Table Testing in type of the triangle.

Program 7

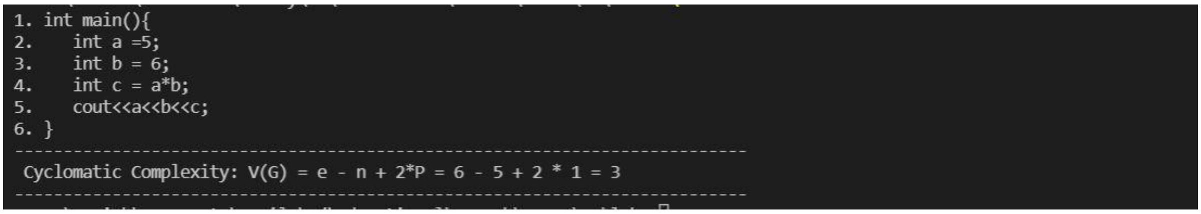
Aim: Write a program to calculate Cyclomatic complexity of a given program.

Source Code: Cyclomatic complexity

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

int calculate_complexity(int e, int n, int P)
{
    int V = e - n + 2 * P;
    return V;
}
int main()
{
    ifstream program_file; string
    read_sinlge_line;
    program_file.open("maximum_programs.txt");
    if (program_file.is_open())
    {
        int e = 0, n = 0, p = 1;
        bool source_code_present; size_t
        function_entry_point;
        while (getline(program_file, read_sinlge_line))
        {
            if (read_sinlge_line.empty())
                continue;
            if (!source_code_present)
            {
                function_entry_point =
read_sinlge_line.find("int main()");
                if (function_entry_point != string::npos)
                {
                                           sourc
                                           e_cod
                                           e_pre
endl;                                      sent =
                                           true;
                                           e++;
                                           cout << e << ". " <<
                                           read_sinlge_line <<

                                       }
                                   }
            else
            {
                if (read_sinlge_line.find_first_of("//") != string::npos)
                    continue;
```

```
                                        e++;
                                        n++;
                                        cout << e << ". " << read_sinlge_line << endl;
                                }
                        }
                        program_file.close(); cout <<
"____________________________________________________________________________________________
____________________" << endl;
                        cout << " Cyclomatic Complexity: V(G) = e - n + 2*P = " << e << " - " << n
<< " + 2 * "
                                << p << " = " << calculate_complexity(e, n, p) <<
endl;
                        cout <<
"____________________________________________________________________________________________
____________________" << endl;
                }
                else
                {
                        cout << "Unable to open file";
                }
                return 0;
}
```

Output:



```
1. int main(){
2.    int a =5;
3.    int b = 6;
4.    int c = a*b;
5.    cout<<a<<b<<c;
6. }
--------------------------------------------------------------------------
 Cyclomatic Complexity: V(G) = e - n + 2*P = 6 - 5 + 2 * 1 = 3
--------------------------------------------------------------------------
```

Learning Outcome:

1. Successfully completed Cyclomatic complexity.
2. Cyclomatic complexity of a program.

Program 8

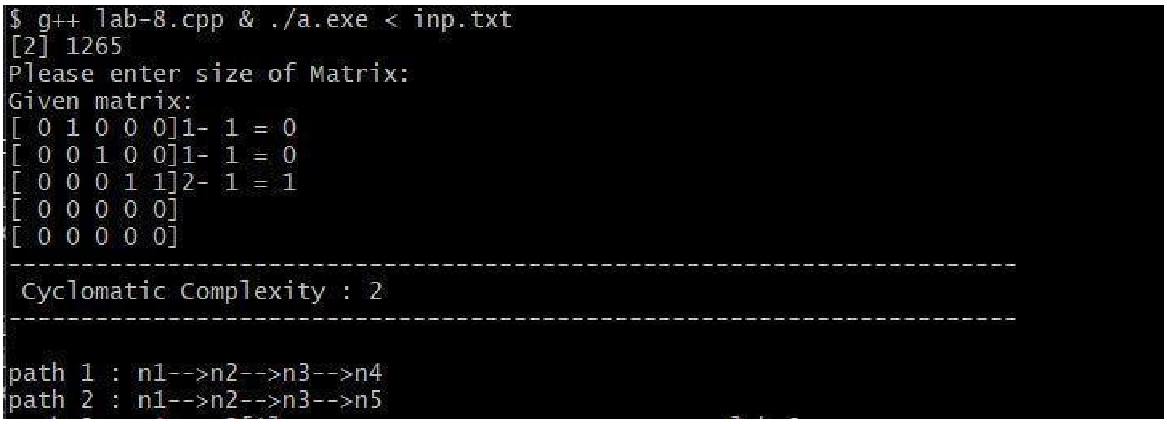
Aim: Write a program to input graph matrix and perform DD path testing.

Source Code: DD path testing

```
#include <iostream>
#include <map> #include
<iterator> #include <list>
using namespace std;
// using adjacency list representation
class Graph
{
        int V; // No. of vertices
        int index;
        // adjacency lists list<int>
        *adj;
        // A recursive function used by dfs_traversal
        void helper(int v, bool visited[]);

public:
        Graph(int V);
        // function to add an edge to graph
        void add_edge(int v, int w);
        // dfs_traversal traversal of the vertices
        // reachable from v
        void dfs_traversal(int v);
};
Graph::Graph(int V)
{
        this -> V = V;
        adj = new list<int>[V]; index = 0;
}
void Graph::add_edge(int v, int w)
{
        adj[v].push_back(w); // Add w to v's list.
}

void Graph::helper(int v, bool visited[])
{
        // Mark the current node as visited and
        // print it
        // Recur for all the vertices adjacent
        // to this vertex
        list<int>::iterator i; i =
        adj[v].begin();
        if (i == adj[v].end())
        {
                visited[v] = true; helper(0,
                visited);
        }
        if (v == 0)
        {
                int count = 0;
                for (int i = 0; i < V; i++)
                        if (visited[i] == false) count++;
                if (count > 1)
                {
                        index++;
                        cout << "\npath " << index << " : n" << v + 1;
                }
        }
        for (; i != adj[v].end(); ++i)
        {
                if (!visited[*i])
                {
                        cout << "-->n" << *i + 1; helper(*i,
                        visited);
                }
                visited[v] = true;
        }
        // if (!visited[*i])
        //      helper(*i, visited);
}

void Graph::dfs_traversal(int v)
{
        // Mark all the vertices as not visited
        bool *visited = new bool[V]; for (int i =
        0; i < V; i++)
                visited[i] = false;
        // Call the recursive helper function
        // to print dfs_traversal traversal helper(v, visited);
}
int main()
{
        cout << "Please enter size of matrix2D:" << endl;
        int m; cin >>
        m;
        int matrix2D[m][m]; Graph
        g(m);
        // Enter t
        for (int i = 0; i < m; i++)
        {
                for (int j = 0; j < m; j++)
                {
```

```
                                // cout<<"please enter "<<i+1<<" th row "
    <<j+1<<"th column value: "<<endl;
                            cin >> matrix2D[i][j];
                }
        }
        cout << "Given matrix2D:" << endl;
        int complexity = 1;
        int value;
        for (int i = 0; i < m; i++)
        {
                cout << "["; value
                = 0;
                for (int j = 0; j < m; j++)
                {
                        cout << " " << matrix2D[i][j];
                        if (matrix2D[i][j] == 1)
                        {
                                value = value + matrix2D[i][j]; g.add_edge(i,
                                j);
                        }
                }
                if (value > 0)
                {
                                                                complexity = complexity + value - 1;
                                                                cout << "]" << value << "- 1 = " <<
endl;                                                           (value - 1) <<

                                                        }
                else
                        cout << "]" << endl;
        }
        cout <<
"____________________________________________________________________________
---------" << endl;
        cout << " Cyclomatic Complexity : " << complexity << endl; cout <<
"____________________________________________________________________________
---------" << endl;
        g.dfs_traversal(0);
        return 0;
}
```

Output:



```
$ g++ lab-8.cpp & ./a.exe < inp.txt
[2] 1265
Please enter size of Matrix:
Given matrix:
[ 0 1 0 0 0]1- 1 = 0
[ 0 0 1 0 0]1- 1 = 0
[ 0 0 0 1 1]2- 1 = 1
[ 0 0 0 0 0]
[ 0 0 0 0 0]
----------------------------------------------------------------------
 Cyclomatic Complexity : 2
----------------------------------------------------------------------

path 1 : n1-->n2-->n3-->n4
path 2 : n1-->n2-->n3-->n5
```

Learning Outcome:

1. Successfully completed DD path testing.
2. DD path testing in type of the matri

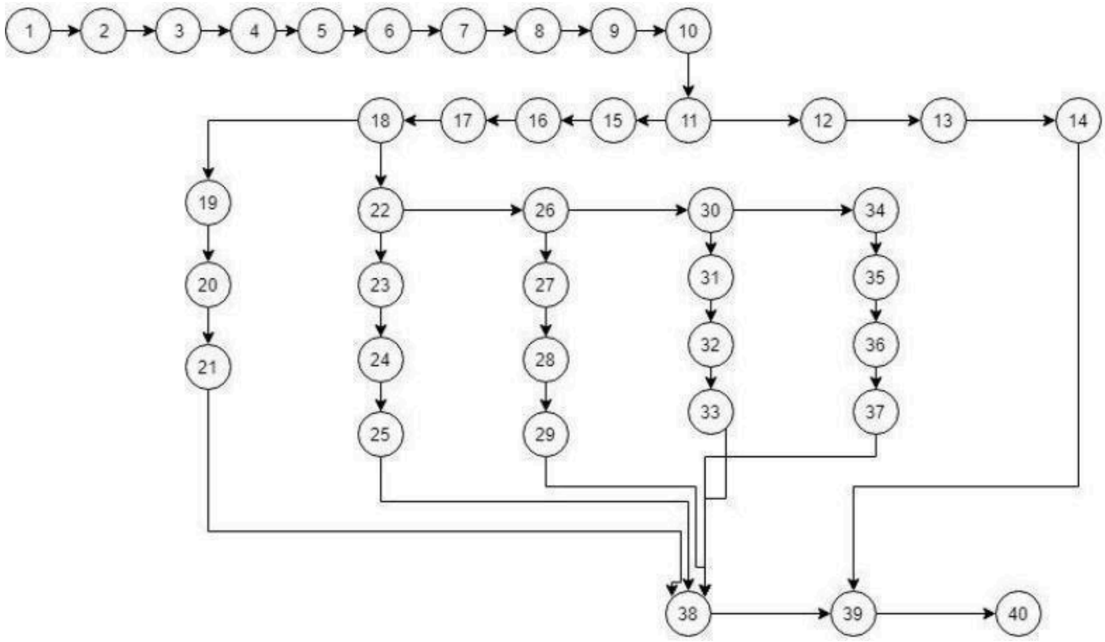
Program 9

Aim: Write a program to find the type of the division of student on the basis user marks input and generate test cases to test the program using Control Flow Testing.

Source Code: Control Flow Testing

```
#include <iostream> using
namespace std; int main()
{
        int subject1, subject2, subject3, average;
        cout << "Enter marks of 3 subjects (between 0-100)" << endl;
        cout << "Enter mark of first subject: "; cin >> subject1;
        cout << "Enter mark of second subject: "; cin >> subject2;
        cout << "Enter mark of third subject: "; cin >> subject3;
        if (subject1 > 100 || subject1 < 0 || subject2 > 100 || subject2 < 0 || subject3 > 100 ||
subject3 < 0)
        {
                cout << "Invalid Marks";
        }
        else
        {
                average = (subject1 + subject2 +                subject3)       /   3;
                if (average < 40)
                {
                        cout << "Fail";
                }
                else if (average >=40 && average                < 50)
                {
                        cout << "Third Division";
                }
                else if (average >=50 && average                < 60)
                {
                        cout << "Second Division";
                }
                else if (average >=60 && average                < 75)
                {
                        cout << "First Division";
                }
                else
                {
                        cout << "First Division with Distinction";
                }
        }
        return 0;
}
```

Control Flow Graph



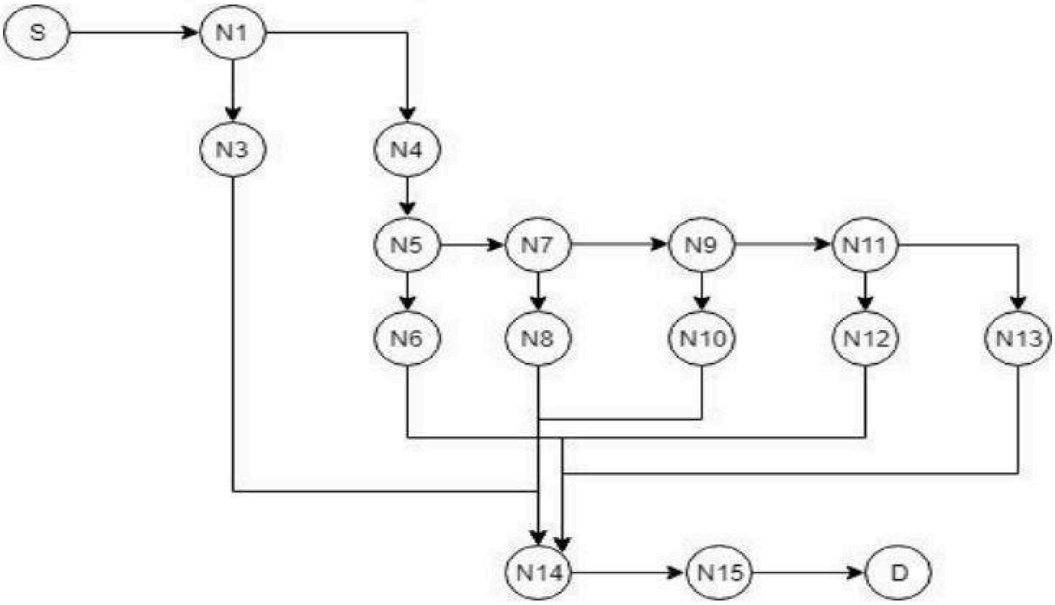
Test ID	Mark1	Mark2	Mark3	Expected Output	Paths
1.	75	80	85	First Division with Distinction	1-10, 11, 15-17, 18, 22, 26, 30, 34-37, 38, 39,40
2.	68	68	68	First Division	1-10, 11, 15-17, 18, 22, 26, 30-33, 38, 39,40
3.	55	55	55	Second Division	1-10, 11, 15-17, 18, 22, 26-29, 38, 39,40
4.	45	45	45	Third Division	1-10, 11, 15-17, 18, 22-25, 38, 39, 40
5.	25	25	25	Fail	1-10, 11, 15-17, 18-21, 38, 39,40
6.	-1	50	50	Invalid marks	1-10, 11-14, 38, 39,40

Mapping of program graph nodes to DD graph nodes

Program graph nodes	DD path graph corresponding node	Remarks
1	S	Source node
2-10	N1	Sequential nodes
11	N2	Decision node, if true goto 12,else goto 15

12-14	N3	Sequential nodes
15-17	N4	Sequential nodes
18	N5	Decision node, if true goto 19, else goto 22
19-21	N6	Sequential nodes
22	N7	Decision node, if true goto 23, else goto 26
23-25	N8	Sequential node
26	N9	Decision node, if true goto 27, else goto 30
27-29	N10	Sequential node
30	N11	Decision node, if true goto 31, else goto 34
31-33	N12	Sequential nodes
34-37	N13	Sequential nodes
38	N14	Junction node, five edges 21, 25, 29, 33 and 37 are terminated here.
39	N15	Junction node, two edges 14 and 38 are terminated here
40	D	Destination node

DD path graph of the program



Learning Outcome:

1. Successfully completed Control Flow Testing.
2. Control Flow Testing to find division of student.

Program 10

Aim: Write a program to find the type of the division of student on the basis user marks input and generate test cases to test the program using Mutation Testing.

Source Code: Mutation Testing

```
#include <iostream> using
namespace std; int main()
{
        int subject1, subject2, subject3, average;
        cout << "Enter marks of 3 subjects (between 0-100)" << endl;
        cout << "Enter mark of first subject: "; cin >> subject1;
        cout << "Enter mark of second subject: "; cin >> subject2;
        cout << "Enter mark of third subject: "; cin >> subject3;
        if (subject1 > 100 || subject1 < 0 || subject2 > 100 || subject2 < 0 || subject3 > 100 ||
subject3 < 0)
        {
                cout << "Invalid Marks";
        }
        else
        {
                average = (subject1 + subject2 +                  subject3) / 3;
                if (average < 40)
                {
                        cout << "Fail";
                }
                else if (average >=40 && average                  < 50)
                {
                        cout << "Third Division";
                }
                else if (average >=50 && average                  < 60)
                {
                        cout << "Second Division";
                }
                else if (average >=60 && average                  < 75)
                {
                        cout << "First Division";
                }
                else
                {
                        cout << "First Division with                  Distinction";
```

```
            }
        }
        return 0;
}
```

Test cases

Test ID	Mark1	Mark2	Mark3	Expected Output
1.	75	80	85	First Division with Distinction
2.	68	68	68	First Division
3.	55	55	55	Second Division
4.	45	45	45	Third Division
5.	25	25	25	Fail
6.	-1	50	50	Invalid marks

Mutated statements

Mutant No.	Line No.	Original Line	Modified Line
M1.	13	mark1 > 100	mark1 < 100
M2.	17	(mark1 + mark2 + mark3)	(mark1 + mark2 + 50)
M3.	18	avg < 40	avg >= 40
M4.	21	avg>=40 && avg<50	avg>=40 || avg<50
M5.	24	avg>=50 && avg<60	avg>=50&& avg>60
M6.	27	Avg>=60 && avg<75	Avg>60 && avg<75

Actual output of mutant M1

Test ID	Mark1	Mark2	Mark3	Expected Output
1.	75	80	85	***Invalid marks**
2.	68	68	68	***Invalid marks**
3.	55	55	55	***Invalid marks**
4.	45	45	45	***Invalid marks**
5.	25	25	25	***Invalid marks**
6.	-1	50	50	Invalid marks

Actual output of mutant M2

Test ID	Mark1	Mark2	Mark3	Expected Output
1.	75	80	85	***First Division**
2.	68	68	68	First Division
3.	55	55	55	Second Division
4.	45	45	45	Third Division
5.	25	25	25	Fail
6.	-1	50	50	Invalid marks

Actual output of mutant M3

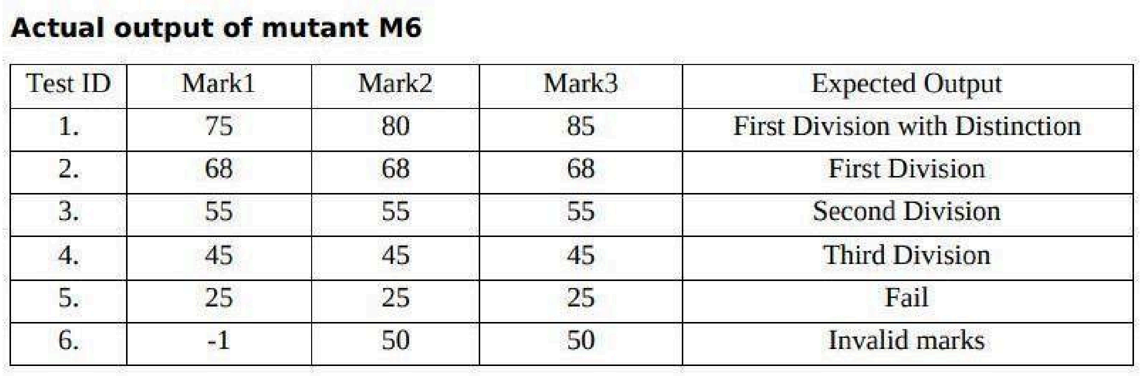
Test ID	Mark1	Mark2	Mark3	Expected Output
1.	75	80	85	* **Fail**
2.	68	68	68	* **Fail**
3.	55	55	55	* **Fail**
4.	45	45	45	* **Fail**
5.	25	25	25	***First Division with Distinction**
6.	-1	50	50	Invalid marks

Actual output of mutant M4

Test ID	Mark1	Mark2	Mark3	Expected Output
1.	75	80	85	* **Third Division**
2.	68	68	68	***Third Division**
3.	55	55	55	* **Third Division**
4.	45	45	45	Third Division
5.	25	25	25	Fail
6.	-1	50	50	Invalid marks

Actual output of mutant M5

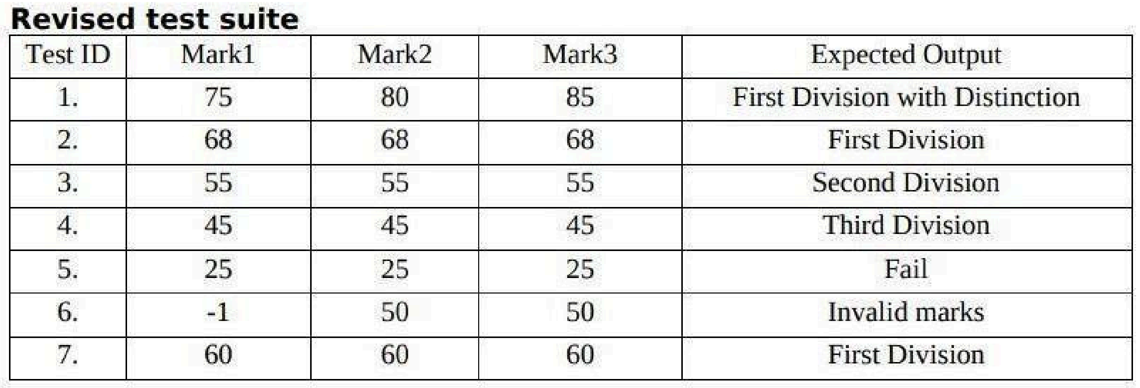
Test ID	Mark1	Mark2	Mark3	Expected Output
1.	75	80	85	***Second Division**
2.	68	68	68	***Second Division**
3.	55	55	55	***First Division with Distinction**
4.	45	45	45	Third Division
5.	25	25	25	Fail
6.	-1	50	50	Invalid marks



Actual output of mutant M6

Test ID	Mark1	Mark2	Mark3	Expected Output
1.	75	80	85	First Division with Distinction
2.	68	68	68	First Division
3.	55	55	55	Second Division
4.	45	45	45	Third Division
5.	25	25	25	Fail
6.	-1	50	50	Invalid marks

$$\text{Mutation Score} = \frac{\text{Number of mutants killed}}{\text{Total number of mutants}} = \frac{5}{6} = 0.8$$



Revised test suite

Test ID	Mark1	Mark2	Mark3	Expected Output
1.	75	80	85	First Division with Distinction
2.	68	68	68	First Division
3.	55	55	55	Second Division
4.	45	45	45	Third Division
5.	25	25	25	Fail
6.	-1	50	50	Invalid marks
7.	60	60	60	First Division

Learning Outcome:

1. Successfully completed Mutation Testing.
2. Mutation Testing to find division of student.

Program 11

Aim: Write a program to generate test cases to test the program using Slice Based Testing.

Source Code: Slice Based Testing

```
#include <fstream>
#include <iostream>
#include <string.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
using namespace
std;

void showslice(char array[], int point[], int k){
  cout << "\n______________________________\n";
  int cur = 1, opening = 0, i, braces = 0, temp;
  char current_line[1000];
  int len = strlen(array);
  int j = 0;
  for (i = 0; i < len - 3; i++)
  {
    while (point[j] != cur)
    {
      if (array[i] == '\n')
      {
        cur++;
      }
      else
      {
        i++;
      }
    }
    cout << array[i];
    if (array[i] ==
      '{') braces++;
    else if (array[i] ==
      '}') braces--;
    else if (array[i] == '\n')
```

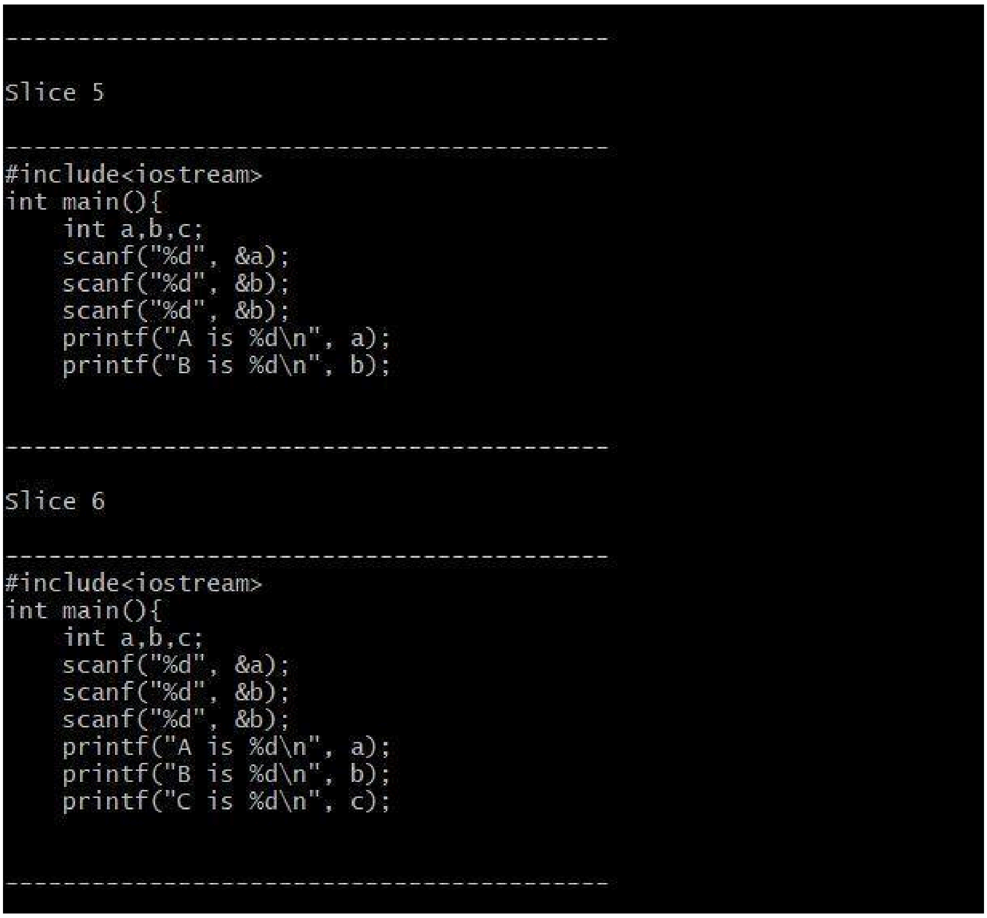
```
{
  cur++
  ; j++;
  if (j > k)
  {
    break;
```

```
            }
        }
    }
    for (; braces > 0 && i < len - 3; i++)
    {
        temp = 0;
        while (array[i] != '\n')
        {
            current_line[temp++] = array[i];
            if (array[i] == '}')
            {
                opening =
                1; braces--;
                i++;
                break;
            }
            i++;
        }
        current_line[temp] = '\n';
        if (opening == 1)
        {
            temp = 0;
            while (current_line[temp] != '\n')
            {
                cout << current_line[temp];
                temp++;
            }
        }
        opening = 0;
    }
    cout << "\n\n______________________________\n";
}
int main()
{
    fstream testFile;
    testFile.open("maximum_programs.txt", ios::in);
    int i = 0, lines = 0, len = 0, slice = 1, opening =
    0; char array[1000];
    int point[1000];
    if (testFile.is_open())
    {
        while (testFile)
        {
            array[i] =
            testFile.get(); i++;
        }
        len = strlen(array);
```

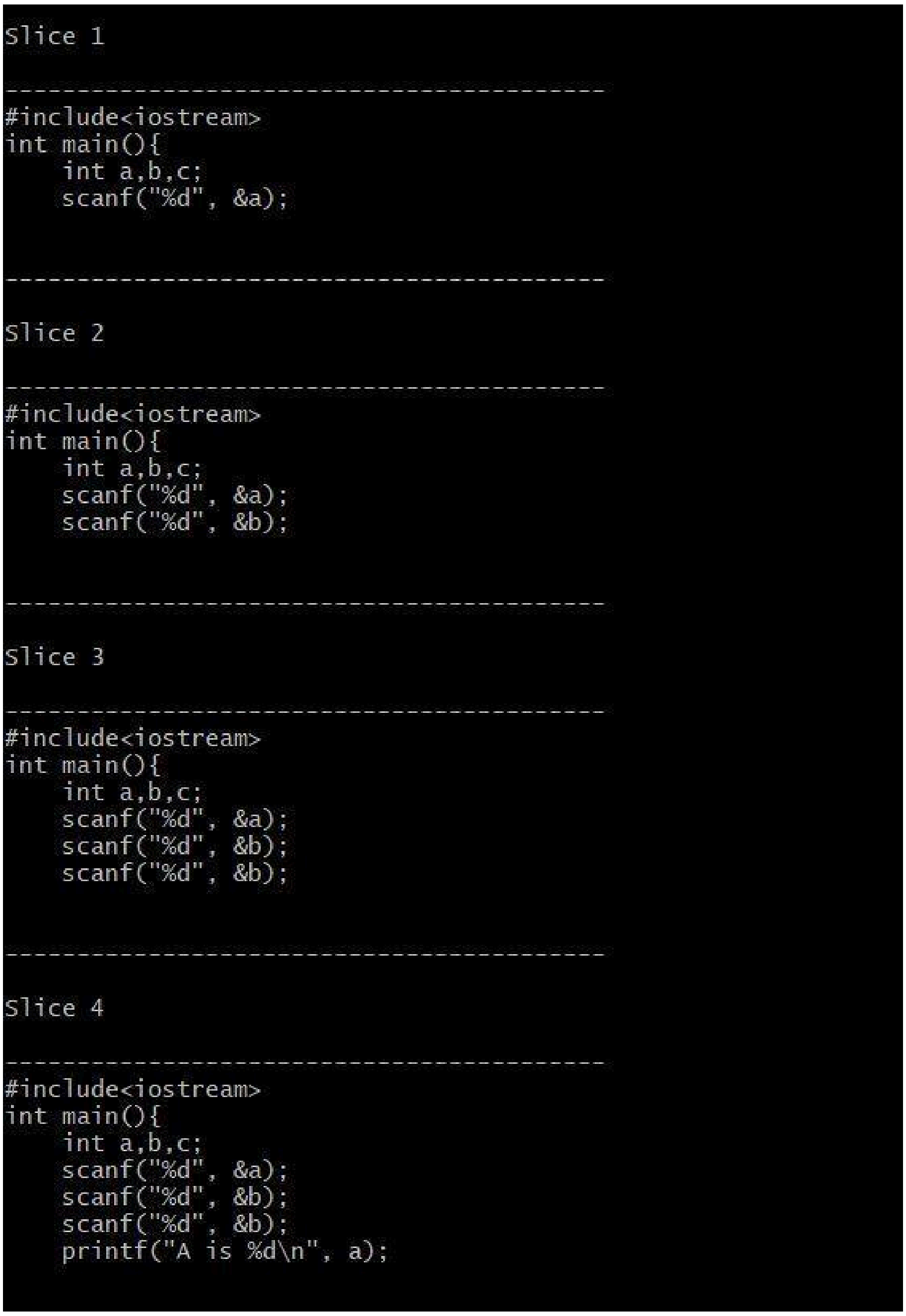
```
}
opening = 0;
int cur = 1;
int k = 0;
for (i = 0; i < len - 3; i++)
{
  if (array[i] == '\n')
  {
    if (!opening)
    {
      point[k] =
      cur; k++;
    }
    cur++;
    opening = 0;
    continue;
  }
  else if (array[i] == 's')
  {
    if (array[i + 1] == 'c')
    {
      if (array[i + 2] == 'a')
      {
        if (array[i + 3] == 'n')
        {
          if (array[i + 4] == 'f')
          {
            if (array[i + 5] == '(')
            {
              point[k] = cur;
              cout << "\nSlice " << slice++ <<
              endl; showslice(array, point, k);
              k++;
              opening = 1;
            }
          }
        }
      }
    }
  }
  else if (array[i] == 'p')
  {
    if (array[i + 1] == 'r')
    {
      if (array[i + 2] == 'i')
      {
        if (array[i + 3] == 'n')
```

```
          {
            if (array[i + 4] == 't')
            {
              if (array[i + 5] == 'f')
              {
                int temp = 1;
                while (array[i + temp] != ',' && array[i + temp] !=
                  '\n') temp++;
                if (array[i + temp] ==
                  ','){ point[k] = cur;
                  cout << "\nSlice " << slice++ <<
                  endl; showslice(array, point, k);
                  k++;
                  opening = 1;
                }
                else{
                  cur++;
                  opening =
                  0;
                }
              }
            }
          }
        }
      }
    }
  }
  testFile.close();
  return 0;
}
```

Output:



```
------------------------------------------

Slice 5

------------------------------------------
#include<iostream>
int main(){
    int a,b,c;
    scanf("%d", &a);
    scanf("%d", &b);
    scanf("%d", &b);
    printf("A is %d\n", a);
    printf("B is %d\n", b);


------------------------------------------

Slice 6

------------------------------------------
#include<iostream>
int main(){
    int a,b,c;
    scanf("%d", &a);
    scanf("%d", &b);
    scanf("%d", &b);
    printf("A is %d\n", a);
    printf("B is %d\n", b);
    printf("C is %d\n", c);


------------------------------------------
```



```
Slice 1

------------------------------------------
#include<iostream>
int main(){
    int a,b,c;
    scanf("%d", &a);


------------------------------------------

Slice 2

------------------------------------------
#include<iostream>
int main(){
    int a,b,c;
    scanf("%d", &a);
    scanf("%d", &b);


------------------------------------------

Slice 3

------------------------------------------
#include<iostream>
int main(){
    int a,b,c;
    scanf("%d", &a);
    scanf("%d", &b);
    scanf("%d", &b);


------------------------------------------

Slice 4

------------------------------------------
#include<iostream>
int main(){
    int a,b,c;
    scanf("%d", &a);
    scanf("%d", &b);
    scanf("%d", &b);
    printf("A is %d\n", a);
```

Learning Outcome:

1. Successfully completed Slice Based Testing.
2. Slice Based Testing for code input using file.