EXPERIMENT 4

I. Write a program to generate test cases using Robust Case approach.

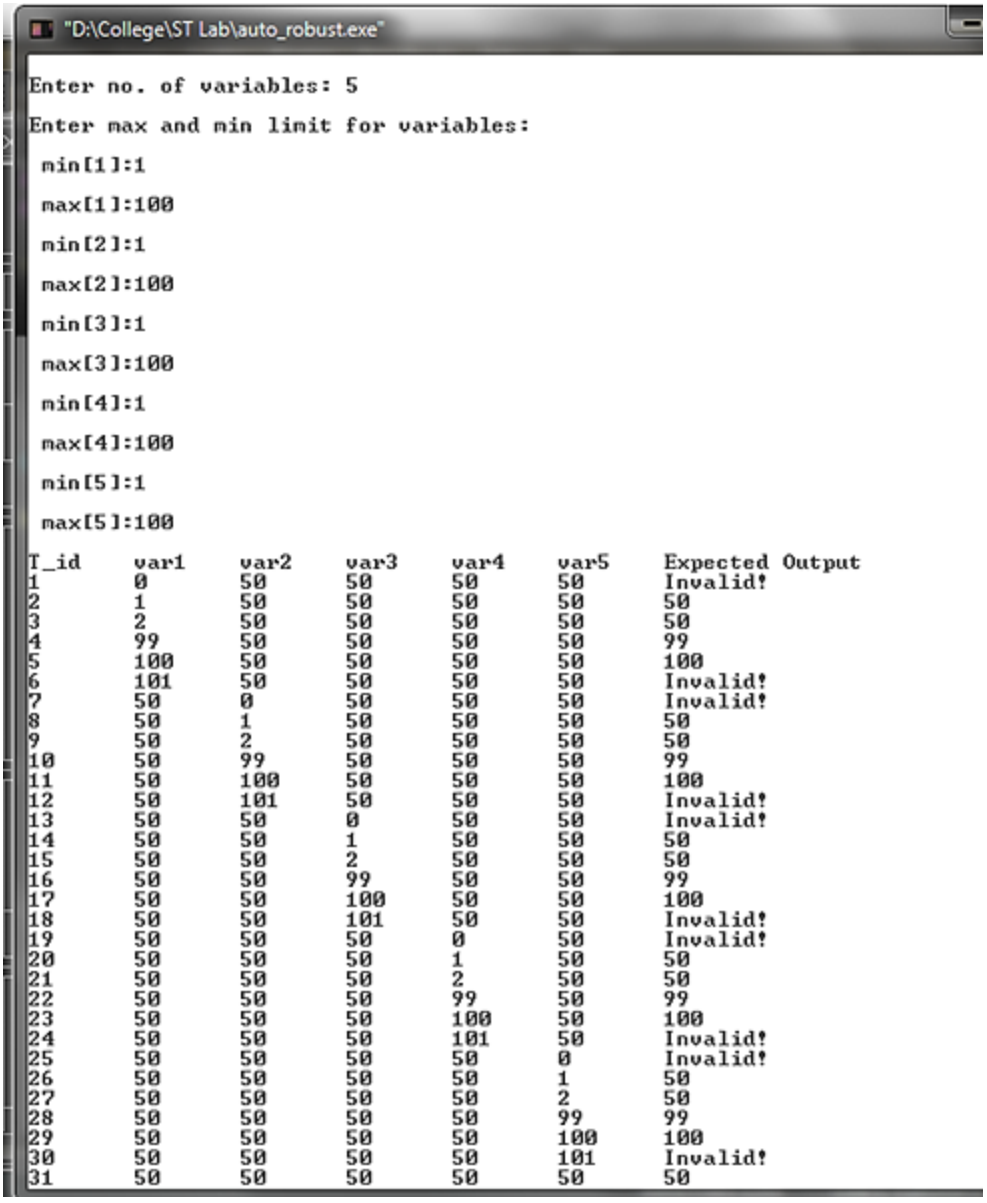
SOURCE CODE

```
#include <stdio.h>
#include <conio.h>
int findmax(int *r, int size) {
	int max =0,i;
	for (i=0; i<size; i++) {
		if( max<r[i] )
		max = r[i];
	}
	return max;
}
int main() {
	int n, i, j, k, id=0, max[32];
	int ub[20], lb[20], t_id=0;
	int row[20], flag=0;
	printf("\nEnter no. of variables: ");
	scanf("%d",&n);
	printf("\nEnter max and min limit for variables: \n");
	for ( i=0; i<n; i++) {
		printf("\n min[%d]:",i+1);
		scanf("%d", &lb[i]);
		printf("\n max[%d]:",i+1);
		scanf("%d", &ub[i]);
	}
	for ( i=-1; i<n; i++) {
		if (i==-1)
			printf("\nT_id\t");
		else
			printf("var%d\t",i+1);
		}
		printf("Expected Output\n");
		for (i=0;i<n;i++) {
			for (j=0;j<6;j++) {
				printf("%d",++t_id);
				for (k=0;k<n;k++) {
					if (k==i) {
						if (j==0) {
							row[k] = lb[i]-1;
							flag=1;
							printf("\t%d",lb[i]-1);
						}

						else if (j==1) {
							row[k] = lb[i];
							printf("\t%d",lb[i]);
						}
```

```
                                        else if (j==2) {
                                                row[k] = lb[i]+1;
                                                printf("\t%d",lb[i]+1);
                                        }
                                        else if (j==3) {
                                                row[k] = ub[i]-1;
                                                printf("\t%d",ub[i]-1);
                                        }
                                        else if (j==4) {
                                                row[k] = ub[i];
                                                printf("\t%d",ub[i]);
                                        }
                                        else {
                                                row[k] = ub[i]+1;
                                                flag = 1;
                                                printf("\t%d",ub[i]+1);
                                        }
                                }
                                else {
                                        row[k] = (ub[k]+lb[k])/2;
                                        printf("\t%d",(ub[k]+lb[k])/2);
                                }
                        }
                        if (flag==1) {
                                printf("\tInvalid!\n");
                                flag=0;
                        }
                        else
                                printf("\t%d\n",findmax(row,n));
                }
        }
        printf("%d",++t_id);
        for (i=0;i<n;i++) {
                row[i] = (ub[i]+lb[i])/2;
                printf("\t%d",(ub[i]+lb[i])/2);
        }
        printf("\t%d\n",findmax(row,n));
        getch();
}
```

OUTPUT:



```
"D:\College\ST Lab\auto_robust.exe"

Enter no. of variables: 5
Enter max and min limit for variables:
 min[1]:1
 max[1]:100
 min[2]:1
 max[2]:100
 min[3]:1
 max[3]:100
 min[4]:1
 max[4]:100
 min[5]:1
 max[5]:100
T_id    var1    var2    var3    var4    var5    Expected Output
1       0       50      50      50      50      Invalid!
2       1       50      50      50      50      50
3       2       50      50      50      50      50
4       99      50      50      50      50      99
5       100     50      50      50      50      100
6       101     50      50      50      50      Invalid!
7       50      0       50      50      50      Invalid!
8       50      1       50      50      50      50
9       50      2       50      50      50      50
10      50      99      50      50      50      99
11      50      100     50      50      50      100
12      50      101     50      50      50      Invalid!
13      50      50      0       50      50      Invalid!
14      50      50      1       50      50      50
15      50      50      2       50      50      50
16      50      50      99      50      50      99
17      50      50      100     50      50      100
18      50      50      101     50      50      Invalid!
19      50      50      50      0       50      Invalid!
20      50      50      50      1       50      50
21      50      50      50      2       50      50
22      50      50      50      99      50      99
23      50      50      50      100     50      100
24      50      50      50      101     50      Invalid!
25      50      50      50      50      0       Invalid!
26      50      50      50      50      1       50
27      50      50      50      50      2       50
28      50      50      50      50      99      99
29      50      50      50      50      100     100
30      50      50      50      50      101     Invalid!
31      50      50      50      50      50      50
```