

VI Semester

SOFTWARE TESTING LABORATORY			
Course Code	21ISL66	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0:0:2:0	SEE Marks	50
Total Hours of Pedagogy	24	Total Marks	100
Credits	1	Exam Hours	03
Course Objectives: CLO 1. Explain the test cases for any given problem CLO 2. Analyze the requirements for the given problem statement. CLO 3. Design the solution and write test cases for the given problem. CLO 4. Construct control flow graphs for the solution that is implemented. CLO 5. Create appropriate document for the software artifact			
Note: two hours tutorial is suggested for each laboratory sessions.			
Prerequisite			
	<ul style="list-style-type: none"> Students should be familiar with programming languages like C, C++, Java, Python etc. Usage of IDEs like Eclipse, Netbeans and software testing tools should be introduced 		
Sl. No.	<i>PART A – List of problems for which student should develop program and execute in theLaboratory</i>		
1	Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of boundary value testing, derive different test cases, execute these test cases and discuss the test results.		
2	Design, develop, code and run the program in any suitable language to implement the NextDate function. Analyze it from the perspective of equivalence class value testing, derive different test cases, execute these test cases and discuss the test results.		
3	Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of decision table-based testing, derive different test cases, execute these test cases and discuss the test results.		
4	Design and develop a program in a language of your choice to solve the triangle problem defined as follows: Accept three integers which are supposed to be the three sides of a triangle and determine if the three values represent an equilateral triangle, isosceles triangle, scalene triangle, or they do not form a triangle at all. Assume that the upper limit for the size of any side is 10. Derive test cases for your program based on boundary-value analysis, equivalence class partitioning and decision-table approach and execute the test cases and discuss the results.		
5	Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of dataflow testing, derive different test cases, execute these test cases and discuss the test results.		
6	Design, develop, code and run the program in any suitable language to implement the binary search algorithm. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.		
	PART B – Practical Based Learning		
01	Develop a Mini Project with documentation of suitable test-cases and their results to perform automation testing of anyE-commerce or social media web page.		

	<p>Suggested Guidelines:</p> <ul style="list-style-type: none"> ● Create a WebDriver session. ● Navigate to a Web page. ● Locate the web elements on the navigated page. ● Perform an actions on the located elements. ● Assert the performed actions did the correct thing. ● Report the results of the assertions. ● End the session. <p>Each inputs / data feeds (ex: website, username, password, mobile no, product name, etc.,) must be provided through a file linked with code and neither to be entered manually nor to be included in the code Use any software testing tool like selenium, Katalon, etc.,</p>
<p>Course Outcome (Course Skill Set) At the end of the course the student will be able to:</p> <p>CO 1. List out the requirements for the given problem and develop test cases for any given problem .</p> <p>CO 2. Design and implement the solution for given problem and to design flow graph</p> <p>CO 3. Use Eclipse/NetBeans IDE and testing tools to design, develop, debug the Project and create appropriate document for the software artifact.</p> <p>CO 4. Use the appropriate functional testing strategies. Compare the different testing techniques.</p> <p>CO 5. Classify and Compare the problems according to a suitable testing model applying the test coverage metrics.</p>	
<p>Assessment Details (both CIE and SEE)</p> <p>The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each course. The student has to secure not less than 35% (18 Marks out of 50) in the semester-end examination (SEE).</p> <p>Continuous Internal Evaluation (CIE):</p> <p>CIE marks for the practical course is 50 Marks.</p> <p>The split-up of CIE marks for record/ journal and test are in the ratio 60:40.</p> <ul style="list-style-type: none"> ● Each experiment to be evaluated for conduction with observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments designed by the faculty who is handling the laboratory session and is made known to students at the beginning of the practical session. ● Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks. ● Total marks scored by the students are scaled down to 30 marks (60% of maximum marks). ● Weightage to be given for neatness and submission of record/write-up on time. ● Department shall conduct 02 tests for 100 marks, the first test shall be conducted after the 8th week of the semester and the second test shall be conducted after the 14th week of the semester. ● In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce. ● The suitable rubrics can be designed to evaluate each student's performance and learning ability. Rubrics suggested in Annexure-II of Regulation book 	

- The average of 02 tests is scaled down to **20 marks** (40% of the maximum marks). The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

- SEE marks for the practical course is 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University
- All laboratory experiments are to be included for practical examination.
- (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the internal /external examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.
- General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)
- Students can pick one experiment from the questions lot of PART A with equal choice to all the students in a batch.
- **PART B** : Student should develop a mini project and it should be demonstrated in the laboratory examination (with report and presentation).
- Weightage of marks for **PART A is 60%** and for **PART B is 40%**. General rubrics suggested to be followed for part A and part B.
- Change of experiment is allowed only once (in part A) and marks allotted to the procedure part to be made zero.
- The duration of SEE is 03 hours.

Suggested Learning Resources:

1. Paul C. Jorgensen: Software Testing, A Craftsman's Approach, 3rd Edition, Auerbach Publications, 2008.
2. Herbert Schildt, C:JavaThe Complete Reference,McGraw Hill,7thEdition

Web links and Video Lectures (e-Resources):

- <https://www.javatpoint.com/selenium-tutorial>
- References
- Introduction to Selenium - <https://www.youtube.com/watch?v=FRn5J31eAMw>
- Introduction to programming -https://www.youtube.com/watch?v=2Xa3Y4xz8_s
- Introduction to OOPS - <https://www.youtube.com/watch?v=pBIH24tFRQk>
- Introduction to Java - <https://www.youtube.com/watch?v=mAtkPQO1FcA>
- Eclipse for java - <https://www.youtube.com/watch?v=8cm1x4bC610>

Program 1

Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of boundary value testing, derived different test cases, execute these test cases and discuss the test results.

Assumption price for lock=45.0, stock=30.0 and barrels=25.0 production limit could sell in a month 70 locks,80 stocks and 90 barrels commission on sales = 10 % <= 1000 and 15 % on 1000 to 1800 and 20 % on above 1800.

```
#include<stdio.h>
int main()
{
    int locks, stocks, barrels, tlocks, tstocks, tbarrels;
    float lprice, sprice, bprice, sales, comm;
    int c1,c2,c3,temp;
    lprice=45.0;
    sprice=30.0;
    bprice=25.0;
    tlocks=0;
    tstocks=0;
    tbarrels=0;
    printf("\nEnter the number of locks and to exit the loop enter -1 for locks\n");
    scanf("%d",&locks);
    while(locks!=-1)
    {
        c1=(locks<=0||locks>70);
        printf("Enter the number of stocks and barrels\n");
        scanf("%d%d",&stocks,&barrels);
        c2=(stocks<=0||stocks>80);
        c3=(barrels<=0||barrels>90);
        if(c1)
            printf("value of locks not in the range 1..70 ");
        else
        {
            temp=tlocks+locks;
            if(temp>70)
                printf("new total locks =%d not in the range 1..70 so old ",temp);
            else
                tlocks=temp;
        }
        printf("total locks = %d\n",tlocks);

        if(c2)
```

```
        printf("value of stocks not in the range 1..80 ");
    else

        {

            temp=tstocks+stocks;
            if(temp>80)
                printf("new total stocks =%d not in the range 1..80 so old ",temp);
            else
                tstocks=temp;
        }
    printf("total stocks=%d\n",tstocks);

    if(c3)
        printf("value of barrels not in the range 1..90 ");
    else
        {

            temp=tbarrels+barrels;
            if(temp>90)
                printf("new total barrels =%d not in the range 1..90 so old ",temp);
            else
                tbarrels=temp;
        }
    printf("total barrel=%d",tbarrels);
    printf("\nenter the number of locks and to exit the loop enter -1 for locks\n");
    scanf("%d",&locks);
}
printf("\ntotal locks = %d\ntotal stocks =%d\ntotal barrels =%d\n",tlocks,tstocks,tbarrels);
sales = lprice*tlocks+sprice*tstocks+bprice*tbarrels;
printf("\nthe total sales=%f\n",sales);

if(tlocks>0&&tstocks>0&&tbarrels>0)
{
    if(sales > 1800.0)
    {
        comm=0.10*1000.0;
        comm=comm+0.15*800;
        comm=comm+0.20*(sales-1800.0);
    }
    else if(sales > 1000)
    {
        comm =0.10*1000;
        comm=comm+0.15*(sales-1000);
    }
    else
        comm=0.10*sales;

    printf("the commission is=%f\n",comm);
}
else
```

```
    printf(" Commission cannot be calculated \n");  
    return 0;  
}
```

```
enter the number of locks and to exit the loop enter -1 for locks  
2  
enter the number of stocks and barrels  
3  
4  
total locks = 2  
total stocks=3  
total barrel=4  
enter the number of locks and to exit the loop enter -1 for locks  
-1  
  
total locks = 2  
total stocks =3  
total barrels =4  
  
the total sales=280.000000  
the commission is=28.000000  
-
```

Test case:

```
#include <stdio.h>
double calculateCommission(double sales) {
    if (sales < 0) {
        return -1;
    } else if (sales < 1000) {
        return sales * 0.05;
    } else if (sales <= 5000) {
        return sales * 0.10;
    } else {
        return sales * 0.15;
    }
}

int main() {
    struct {
        double sales;
        double expected;
    } testCases[] = {
        {500, 25.0},
        {1000, 100.0},
        {3000, 300.0},
        {6000, 900.0},
        {-500, -1},
        {0, 0.0}
    };

    int numTests = sizeof(testCases) / sizeof(testCases[0]);
    int passed = 0;

    for (int i = 0; i < numTests; ++i) {
        double result = calculateCommission(testCases[i].sales);
        if (result == testCases[i].expected) {
            passed++;
            printf("Test case %d passed.\n", i + 1);
        } else {
            printf("Test case %d failed. Expected: %.2f, Got: %.2f\n", i + 1, testCases[i].expected, result);
        }
    }

    printf("\n%d/%d test cases passed.\n", passed, numTests);
    return 0;
}
```

```
C:\TURBOC3\BIN>TC
Test case 1 passed.
Test case 2 passed.
Test case 3 passed.
Test case 4 passed.
Test case 5 passed.
Test case 6 passed.

6/6 test cases passed.
```


Program 2 (this code included test case also)

Design, develop, code and run the program in any suitable language to implement theNextDate function. Analyze it from the perspective of equivalence class value testing, derive different test cases, execute these test cases and discuss the test results.

```
#include <stdio.h>
#include <conio.h>

#define LEAP_YEAR(year) (((year) % 4 == 0 && (year) % 100 != 0) || ((year) % 400 == 0))
#define DAYS_IN_MONTH(month, year) \
    (((month) == 2) ? (LEAP_YEAR(year) ? 29 : 28) : \
    ((month) == 4 || (month) == 6 || (month) == 9 || (month) == 11) ? 30 : 31)

void theNextDate(int day, int month, int year, int *nextDay, int *nextMonth, int *nextYear) {
    int daysInCurrentMonth = DAYS_IN_MONTH(month, year);

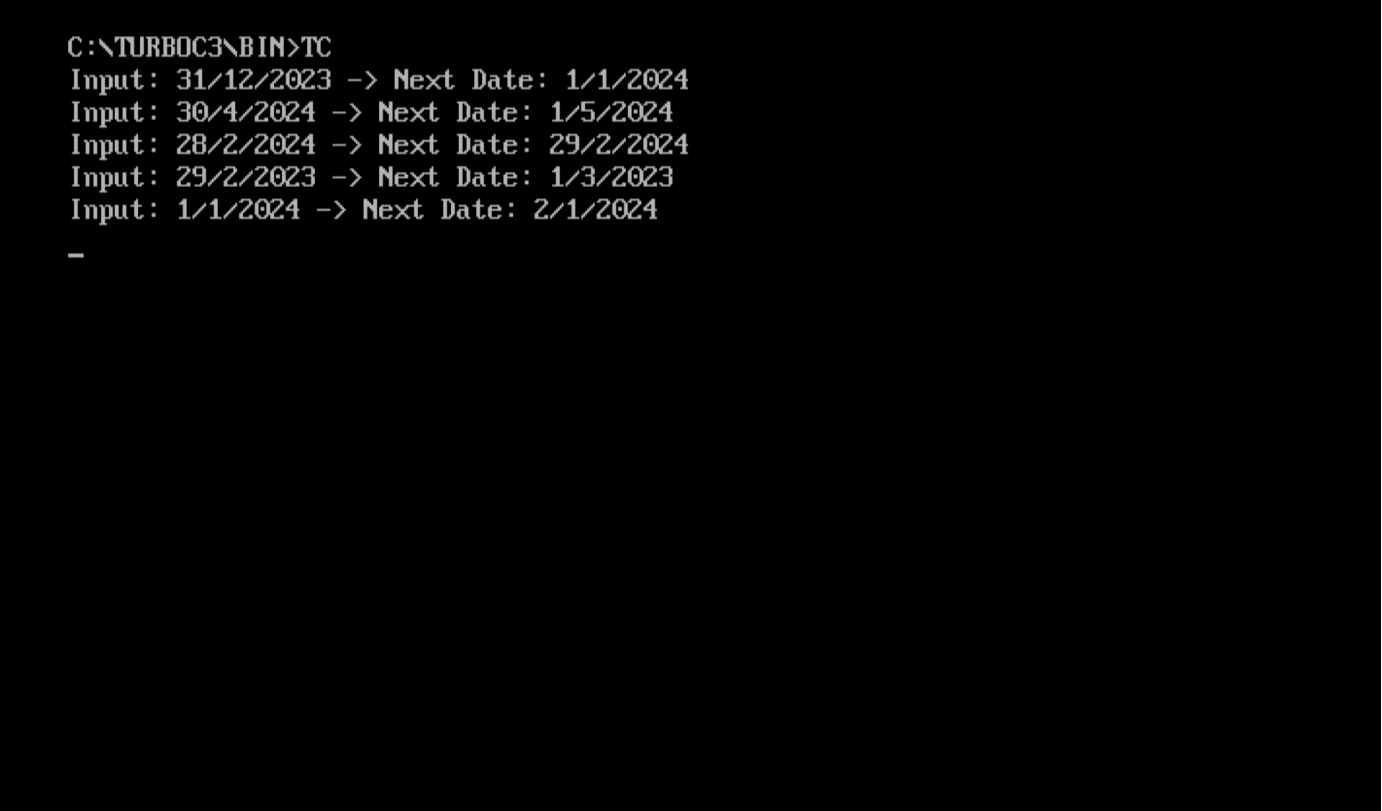
    *nextDay = day + 1;
    *nextMonth = month;
    *nextYear = year;

    if (*nextDay > daysInCurrentMonth) {
        *nextDay = 1;
        *nextMonth = month + 1;
        if (*nextMonth > 12) {
            *nextMonth = 1;
            *nextYear = year + 1;
        }
    }
}

int main() {
    int day, month, year;
    int nextDay, nextMonth, nextYear;

    // Test cases
    int testCases[][3] = {
        {31, 12, 2023}, // Normal end of year
        {30, 4, 2024}, // End of month
        {28, 2, 2024}, // End of February in a leap year
        {29, 2, 2023}, // End of February in a non-leap year
        {1, 1, 2024}, // Start of the year
    };
}
```

```
for (int i = 0; i < sizeof(testCases) / sizeof(testCases[0]); i++) {  
    day = testCases[i][0];  
    month = testCases[i][1];  
    year = testCases[i][2];  
  
    theNextDate(day, month, year, &nextDay, &nextMonth, &nextYear);  
    printf("Input: %d/%d/%d -> Next Date: %d/%d/%d\n", day, month, year, nextDay, nextMonth, nextYear);  
}  
getch();  
return 0;  
}
```



```
C:\TURBOC3\BIN>TC  
Input: 31/12/2023 -> Next Date: 1/1/2024  
Input: 30/4/2024 -> Next Date: 1/5/2024  
Input: 28/2/2024 -> Next Date: 29/2/2024  
Input: 29/2/2023 -> Next Date: 1/3/2023  
Input: 1/1/2024 -> Next Date: 2/1/2024  
-
```

Program 3

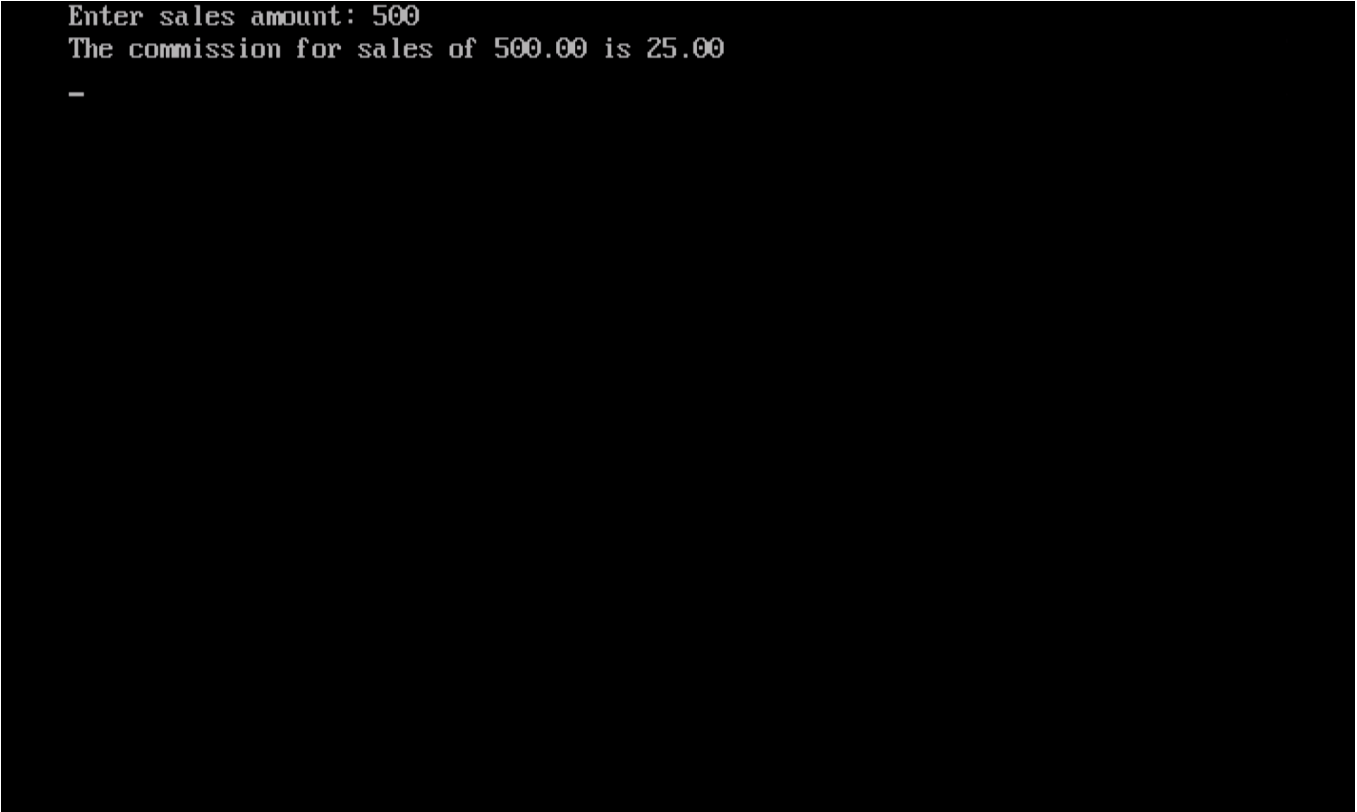
Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of decision table-based testing, derive different test cases, execute these test cases and discuss the test results.

```
#include <stdio.h>

double calculateCommission(double sales)
{
    double commission = 0.0;
    if (sales < 0) {
        printf("Sales amount cannot be negative.\n");
        return -1;
    }
    if (sales <= 1000) {
        commission = sales * 0.05;
    } else if (sales <= 5000) {
        commission = 1000 * 0.05 + (sales - 1000) * 0.10;
    } else {
        commission = 1000 * 0.05 + 4000 * 0.10 + (sales - 5000) * 0.15;
    }
    return commission;
}

int main()
{
    double sales;
    printf("Enter sales amount: ");
    scanf("%lf", &sales);
    double commission = calculateCommission(sales);
    if (commission >= 0)
    {
        printf("The commission for sales of %.2lf is %.2lf\n", sales, commission);
    }
}
```

```
return 0;  
}
```



```
Enter sales amount: 500  
The commission for sales of 500.00 is 25.00  
-
```

Test case :

```
#include <assert.h>
#include <stdio.h>
#include <conio.h>

double calculateCommission(double salesAmount, double commissionRate) {
    return salesAmount * commissionRate;
}

int main() {
    // Test case 1: Sales amount = 1000, Commission rate = 5%

    double salesAmount = 1000;
    double commissionRate = 0.05;
    double expectedCommission = 50;
    assert(calculateCommission(salesAmount, commissionRate) == expectedCommission);

    // Test case 2: Sales amount = 2000, Commission rate = 10%

    salesAmount = 2000;
    commissionRate = 0.10;
    expectedCommission = 200;
    assert(calculateCommission(salesAmount, commissionRate) == expectedCommission);

    // Test case 3: Sales amount = 500, Commission rate = 3%

    salesAmount = 500;
    commissionRate = 0.03;
    expectedCommission = 15;
    assert(calculateCommission(salesAmount, commissionRate) == expectedCommission);

    // Test case 4: Sales amount = 0, Commission rate = 5%

    salesAmount = 0;
    commissionRate = 0.05;
    expectedCommission = 0;
    assert(calculateCommission(salesAmount, commissionRate) == expectedCommission);

    printf("All test cases passed!\n");
    getch();
    return 0;
}
```



```
All test cases passed!
```

```
=== Code Execution Successful ===|
```

Program 4

A] BOUNDARY VALUE ANALYSIS :

Design and develop a program in a language of your choice to solve the triangle problem defined as follows: Accept three integers which are supposed to be the three sides of a triangle and determine if the three values represent an equilateral triangle, isosceles triangle, scalene triangle, or they do not form a triangle at all. Assume that the upper limit for the size of any side is 10. Derive test cases for your program based on boundary value analysis, equivalence class partitioning and decision-table approach and execute the test cases and discuss the results.

```
#include<stdio.h>
#include<conio.h>
int main( )
{
    int a,b,c,c1,c2,c3;
    do
    {
        printf("enter the sides of triangle\n");
        scanf("%d%d%d",&a,&b,&c);
        c1=((a>=1) && (a<=10));
        c2=((b>=1) && (b<=10));
        c3=((c>=1) && (c<=10));
        if(!c1)
            printf("value of a is out of range");
        if(!c2)
            printf("value of b is out of range");
        if(!c3)
            printf("value of c is out of range");
    }while(!c1 || !c2 || !c3);
    if((a+b)>c && (b+c)>a && (c+a)>b)
    {
```

```
if(a==b && b==c)
    printf("Triangle is equilateral\n");

else if(a!=b && b!=c && c!=a)
    printf("Triangle is scalene\n");
else
    printf("Triangle is isosceles\n");
}
else
    printf("Triangle cannot be formed \n");
getch();
return 0;
}
```

```
enter the sides of triangle
3
3
3
Triangle is equilateral
-
```

```
enter the sides of triangle
1
2
3
Triangle cannot be formed
-
```

```
enter the sides of triangle
3
4
5
Triangle is scalene
-
```

Test case:

```
#include <stdio.h>
#include <conio.h>
int isValidTriangle(int a, int b, int c)
{
    return (a + b > c) && (a + c > b) && (b + c > a);
}

void runTestCases()
{
    struct {
        int side1, side2, side3;
        int expectedResult;
    } testCases[] = {
        {3, 4, 5, 1}, // Valid triangle (right triangle)
        {1, 2, 3, 0}, // Not a valid triangle (degenerate case)
        {5, 12, 13, 1}, // Valid triangle (right triangle)
        {10, 2, 8, 0}, // Not a valid triangle
        {7, 10, 5, 1} // Valid triangle
    };

    int numTests = sizeof(testCases) / sizeof(testCases[0]);
    for (int i = 0; i < numTests; i++)
    {
        int result = isValidTriangle(testCases[i].side1, testCases[i].side2, testCases[i].side3);

        printf("Test Case %d: Sides = %d, %d, %d\n", i + 1, testCases[i].side1, testCases[i].side2,
testCases[i].side3);

        printf("Expected Result = %d\n", testCases[i].expectedResult);

        printf("Function Result = %d\n", result);
        printf("Test %s\n", (result == testCases[i].expectedResult) ? "Passed" : "Failed");
    }
}

int main() {
    runTestCases();
    getch();
    return 0;
}
```



```
Expected Result = 1  
Function Result = 1  
Test Passed
```

```
Test Case 2: Sides = 1, 2, 3  
Expected Result = 0  
Function Result = 0  
Test Passed
```

```
Test Case 3: Sides = 5, 12, 13  
Expected Result = 1  
Function Result = 1  
Test Passed
```

```
Test Case 4: Sides = 10, 2, 8  
Expected Result = 0  
Function Result = 0  
Test Passed
```

```
Test Case 5: Sides = 7, 10, 5  
Expected Result = 1  
Function Result = 1  
Test Passed
```

Program 4

B] DECISION TABLE APPROACH FOR SOLVING TRIANGLE PROBLEM

```
#include<stdio.h>
int main()
{
    int a,b,c;
    char istriangle;
    printf("enter 3 integers which are sides of triangle\n");
    scanf("%d%d%d",&a,&b,&c);
    printf("a=%d\t,b=%d\t,c=%d",a,b,c);

    // to check is it a triangle or not

    if( a<b+c && b<a+c && c<a+b )
        istriangle='y';
    else
        istriangle='n';

    // to check which type of triangle

    if (istriangle=='y')
        if ((a==b) && (b==c))
            printf("equilateral triangle\n");
        else if ((a!=b) && (a!=c) && (b!=c))
            printf("scalene triangle\n");
        else
            printf("isosceles triangle\n");
    else
        printf("Not a triangle\n");
    return 0;
}
```

```
enter 3 integers which are sides of triangle
4
4
4
a=4      ,b=4      ,c=4equilateral triangle
-
```

```
enter 3 integers which are sides of triangle
2
3
4
a=2      ,b=3      ,c=4scalene triangle
-
```

Program 5 (this code included test case also)


Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of dataflow testing, derive different test cases, execute these test cases and discuss the test results.

```
#include <stdio.h>

double calculateCommission(double sales)
{
    double commission = 0.0;
    if (sales < 0) {
        printf("Sales amount cannot be negative.\n");
        return -1;
    }
    if (sales <= 1000) {
        commission = sales * 0.05;
    } else if (sales <= 5000) {
        commission = 1000 * 0.05 + (sales - 1000) * 0.10;
    } else {
        commission = 1000 * 0.05 + 4000 * 0.10 + (sales - 5000) * 0.15;
    }
    return commission;
}

void runTestCase(double sales, double expectedCommission)
{
    double commission = calculateCommission(sales);
    if (commission == expectedCommission) {
        printf("Test case passed for sales amount %.2lf\n", sales);
    }
    else
```

```
{  
    printf("Test case failed for sales amount %.2lf: expected %.2lf, got %.2lf\n", sales, expectedCommission,  
commission);  
}  
}  
  
void runAllTestCases()  
{  
    runTestCase(500, 25.0);  
    runTestCase(1500, 100.0);  
    runTestCase(6000, 600.0);  
    runTestCase(-500, -1);  
}  
  
int main()  
{  
    runAllTestCases();  
    return 0;  
}
```



Sales amount cannot be negative.
Test case passed for sales amount -500.00

Program 6(this code included test case also)

Design, develop, code and run the program in any suitable language to implement the binary search algorithm. Determine the basis paths and using them derive different

Test cases, execute these test cases and discuss the test results.

```
#include <stdio.h>
#include <conio.h>
int binarySearch(int arr[], int size, int target)
{
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == target) {
            return mid; // Target found
        } else if (arr[mid] < target) {
            left = mid + 1; // Search in the right half
        } else {
            right = mid - 1; // Search in the left half
        }
    }

    return -1; // Target not found
}

void testBinarySearch()
{
    int arr1[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int size1 = sizeof(arr1) / sizeof(arr1[0]);

    // Test case 1: Target is present in the array
    int target1 = 5;
    int result1 = binarySearch(arr1, size1, target1);
    printf("Test Case 1: ");
    printf("Target %d found at index %d\n", target1, result1);

    // Test case 2: Target is not present in the array
    int target2 = 11;
    int result2 = binarySearch(arr1, size1, target2);
    printf("Test Case 2: ");
    printf("Target %d not found in array, result = %d\n", target2, result2);
}
```

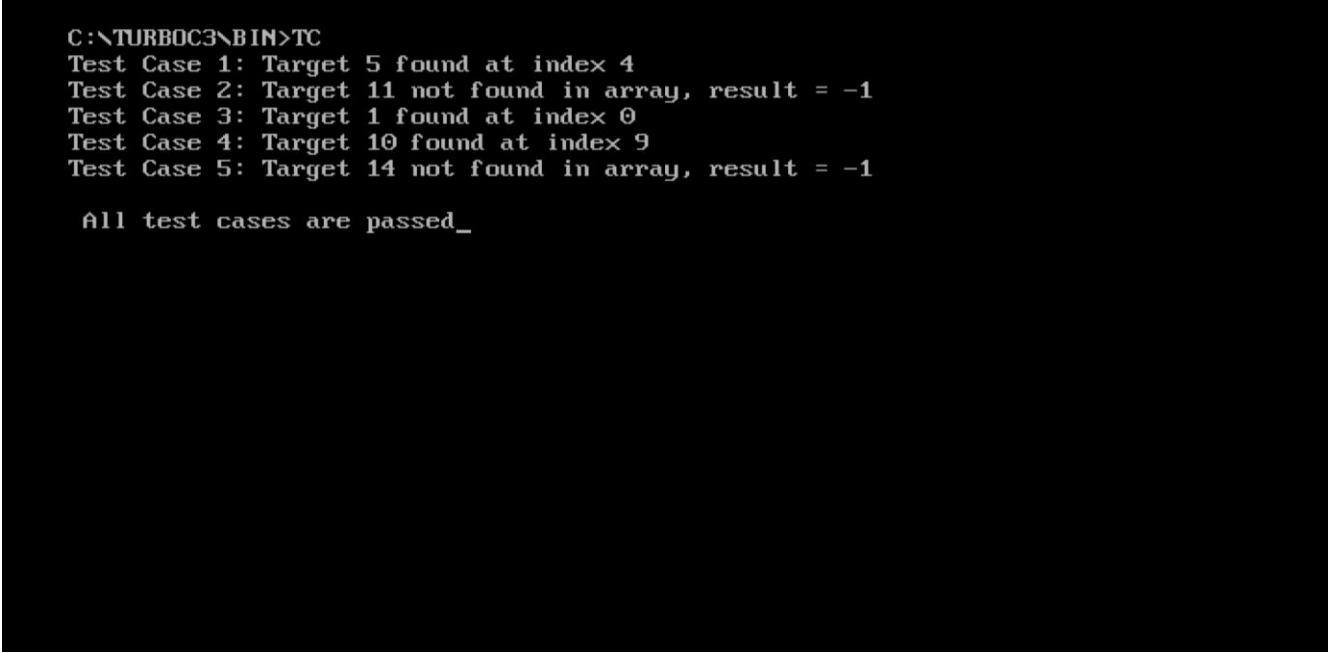
```
// Test case 3: Target is the first element
int target3 = 1;
int result3 = binarySearch(arr1, size1, target3);
printf("Test Case 3: ");
printf("Target %d found at index %d\n", target3, result3);

// Test case 4: Target is the last element
int target4 = 10;
int result4 = binarySearch(arr1, size1, target4);
printf("Test Case 4: ");
printf("Target %d found at index %d\n", target4, result4);

// Test case 5: Empty array
int target5 = 14;
int result5 = binarySearch(arr1, size1, target5);
printf("Test Case 5: ");
printf("Target %d not found in array, result = %d\n", target5, result5);
printf("\n All test cases are passed");

}

int main() {
    testBinarySearch();
    getch();
    return 0;
}
```



```
C:\TURBOC3\BIN>TC
Test Case 1: Target 5 found at index 4
Test Case 2: Target 11 not found in array, result = -1
Test Case 3: Target 1 found at index 0
Test Case 4: Target 10 found at index 9
Test Case 5: Target 14 not found in array, result = -1

All test cases are passed_
```