



REVA
UNIVERSITY

Bengaluru, India

SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY

SOFTWARE ENGINEERING & TESTING LAB MANUAL

FOR

BTST17F2900

III Year B.Tech in Computer Science and Engineering

(Prepared in –Dec 2016)

NAME	
SRN	
SEM	
SECTION	
BRANCH	
ACADEMIC YEAR	

CONTENTS

SI. NO.	DESCRIPTION	Page No.
I	Course Description	5
II	Course Objectives	5
III	Course Outcomes	5
IV	Guidelines to Students	5
PROGRAMS TO BE PRACTICED		
1	A softball team submit equipment list to their sponsors as cardinals and bluejays. The Quantity of their equipment's (bats, balls and gloves) for respective sponsors as cardinals is 12, 45, and 15 and for Blue jays are 15, 38, and 17. The cost of each equipment (Bat, Balls, and Gloves) are 15, 38 and 17. At the end both the sponsors wants their amount of contributions. Write a program in C language for formulating the above context, and test the same using boundary value analysis. Introspect the causes for its failure and write down the possible reasons for its failure.	6-10
2	A Rifle salespersons in the former Arizona Territory sold rifle locks, stocks and barrels made by a gunsmith in Missouri. Locks cost \$45, Stocks cost \$30, and barrels cost \$25. The salesperson had to sell at-least one complete rifle per month, and production limits were such that the most the salesperson could sell in a month was 70 locks, 80 locks, and 90 barrels. At the end of the month, the salesperson sent a telegram showing -1 lock sold. The Smith then knew the sales for the month were complete and computed the salesperson's commission as follows: 10%` on sales up to \$1000, 15% on the next \$800, and 20% on any sales in excess of \$1800. Write a C program for above context and generate Test cases for same using data flow testing.	11-15
3	Take any ATM system (e.g. SBI Bank ATM system) and study its system specifications and report the various bugs. <ol style="list-style-type: none"> 1. Machine is accepting ATM card. 2. Machine is rejecting expired card. 3. Successful entry of PIN number. 4. Unsuccessful operation due to enter wrong PIN number 3 times. 5. Successful selection of language. 6. Successful selection of account type. 7. Unsuccessful operation due to invalid account type. 8. Successful selection of amount to be withdrawn. 9. Successful withdrawal. 10. Expected message due to amount is greater than day limit. 11. Unsuccessful withdraw operation due to lack of money in ATM. 	16-21

	<p>12. Expected message due to amount to withdraw is greater than possible balance.</p> <p>13. Unsuccessful withdraw operation due to click cancel after insert card.</p> <p>Write a C program to Implement the above context, and generate test cases using decision table approach.</p>	
4	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. Design and develop a program in a language of your choice to solve the triangle problem defined as follows: Derive test cases for your program based on decision-table approach, execute the test cases and discuss the results.	22-26
5	Pick up the nearest phonebook and open it to the first page of names. We're looking to find the first "Smith". Look at the first name. Is it "Smith"? Probably not (it's probably a name that begins with 'A'). Now look at the next name. Is it "Smith"? Probably not. Keep looking at the next name until you find "Smith". You started at the beginning of a sequence and went through each item one by one, in the order they existed in the list, until you found the item you were looking for. Design and develop the code for above context in C language. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.	27-30
6	Suppose you have a list of 10 things and you have two friends and you ask both the friends to order the things in descending order of their liking. You will have two arrays after that, now you count the number of inversion they have and more the number of inversion, the more dissimilar their choices are, less the number of inversions the more similar their choices are. So when you sort an array using merge sort, what you do is you put the array you want to obtain on the top and the array you are starting off with at the bottom and connect all the numbers respectively. The number of intersections you obtain, will be your number of inversions. The maximum number of inversions you can have in an array is $n(n-1)/2$, where (n) is the size of the array and that is when your array is in the reverse order of the order in which you want to arrange it.	31-36
7	<p>Assume that the marks obtained by a student in a test of 100 marks and compute his grade according to the following criteria.</p> <p>Marks ≥ 80 grade=A</p> <p>Marks ≥ 70 & < 80 grade=B</p> <p>Marks ≥ 60 & < 70 grade=C</p> <p>Marks ≥ 50 & < 60 grade=D</p> <p>otherwise grade=F</p> <p>Design a code in C language for the above context. Determine the basis paths and using them derive different test cases, execute these test</p>	37-41

	cases and discuss the test results.	
8	In the game of "twenty questions", your task is to guess the value of a hidden number that is one of the N integers between 0 and N-1. (For simplicity, we will assume that N is a power of two.) Each time that you make a guess, you are told whether your guess is too high or too low. An effective strategy is to maintain an interval that contains the hidden number, guess the number in the middle of the interval, and then use the answer to halve the interval size. Develop a C Code which suits the above context. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.	42-45
9	Given a Binary Number as string, print its 1's and 2's complements. 1's complement of a binary number is another binary number obtained by toggling all bits in it, i.e., transforming the 0 bit to 1 and the 1 bit to 0. Design a C program to Implement above context and test the same using Decision table approach. Derive different test cases, execute these test cases and discuss the test results.	46-51
10	The birth-date for set of people is given in a database, which includes all different categories of birthday holders (leap year, born in different month etc). Design and develop code in C language. Analyze it from the perspective of boundary value testing, derive different test cases, execute these test cases and discuss the test results.	52-56
InformationonVivaquestions		57

I. COURSE DESCRIPTION

Software testing lab underlines fundamental concepts of software testing and develops an expertise of how-to-do for future software testers. The lab focuses on delivering testing Plan, test cases and test suites. Most of the lab sessions will be dedicated to apply techniques and tools to test software units. The lab exercises will use test beds from different programming language. In particular test beds will include student implemented code. This will provide immediate feedback to the students and help him/her to avoid program errors.

II. LAB OBJECTIVES

This course will help students to achieve the following objectives:

1. Apply Different Testing methods like Boundary Value Analysis, Decision Table approach, Equivalence partitioning etc., to test different application.
2. Create test strategies and plans, design test cases, prioritize and execute them,
3. Analyze the performance and suitability of different test approaches like white box testing/ Black Box Testing.
4. Understand the importance of testing in Software development life cycle.

III. COURSE OUTCOMES

At the end of the course Students will know how to:

1. Develop test plans covering different phases of software testing.
2. Develop software testing specifications document.
3. Derive test coverage metrics using McCabe basis path testing.
4. Apply test cases to software test beds.
5. Differentiate the testing techniques and also applicability of different testing techniques.

IV. GUIDELINES TO STUDENTS

Following are the required hardware and software for this lab, which is available in the laboratory.

Hardware: Desktop system or Virtual machine in a cloud with OS installed. Presently in the Lab, Pentium IV Processor having 1GB RAM and 250GB Hard Disk is available. Desktop systems are dual boot having Windows as well as Linux installed on them.

Software: C-compiler

Linux environment: All the programs have been written using 'vi' editor and tested using the gcc compiler in Linux environment. Steps for the successful program completion are:

Open vi editor using the shell command

`$vi <filename.c>`

Windows environment: Programs can be edited, compiled and executed using Turbo C/C++ IDE. All operations like opening a file, saving file, compiling the program and executing the program are listed under different menus inside the IDE. Keyboard shortcuts are also available for the above mentioned operations.

Note: Every lab in the manual includes problem statement, learning outcomes, theoretical description, algorithm, program, program description, expected results, implementation phase, simulation of syntax and logical errors, final program with results, assignment give, and viva question.

Recommendation: It is advised to use Linux environment for executing the programs given as part of this manual.

PROGRAM 1	
1	Problem Statement
1.	A softball team submit equipment list to their sponsors as cardinals and bluejays. The Quantity of their equipments (bats, balls and gloves) for respective sponsors as cardinals is 12, 45, and 15 and for Bluejays are 15, 38, and 17. The cost of each equipment (Bat,Balls,Gloves) are 15, 38 and 17. At the end both the sponsors wants their amount of contributions. Write a program in C language for formulating the above context, and test the same using boundary value analysis. Introspect the causes for its failure and write down the possible reasons for its failure.
2	Student Learning Outcomes
	<p>After successful completion of this lab, the student shall be able to</p> <ol style="list-style-type: none"> Understand the concept and complexity of matrix multiplication. Identify the set of Boundary values required for matrix multiplication. Generates set of feasible test cases.
3	Design of the Program
3.1	Description
	<p>Boundary value analysis is a black box test design technique and it is used to find the errors at boundaries of input domain rather than finding those errors in the center of input.</p> <p>To set up boundary value analysis test cases you first have to determine which boundaries you have at the interface of a software component. The tendency is to relate boundary value analysis more to the so called black box testing, which is strictly checking a software component at its interfaces, without consideration of internal structures of the software.</p>
3.2	Algorithm
	<p>STEP 1: Start</p> <p>STEP 2: Declare variables and initialize necessary variables</p> <p>STEP 3: Enter the element of matrices by row wise using loops</p> <p>STEP 4: Check the number of rows and column of first and second matrices</p> <p>STEP 5: If number of rows of first matrix is equal to the number of columns of second matrix, go to step 2</p> <p>STEP 6: Otherwise, print matrix multiplication is not possible and goes to step 3.</p> <p>STEP 7: Multiply the matrices using nested loops.</p> <p>STEP 8: Print the product in matrix form as console output.</p> <p>STEP 9: Stop</p> <p>Algorithm Description:</p> <ul style="list-style-type: none"> Step 1 is the starting of the algorithm Step 2 is used to declare the variables. Step 3 is used to provide the inputs for the matrices. Step 4, 5 & 6 is used for checking the compatibility of matrices.

- Step 7 is used to calculate the multiplication of matrices based on the provided input.
- Step 8 is displaying the output.
- Step 9 is for stop the process.

Coding using C Language

```

1. #include<stdio.h>
2. void main()
3. {
4.   int a[3][3],b[3][3],c[3][3],i,j,k,m,n,p,q;           //variable declaration
5.   printf("Enter 1st matrix no.of rows & cols");// taking the no of rows & column for first  matrices
6.   scanf("%d%d",&m,&n);
7.   printf(" Enter 2nd matrix no.of rows & cols");    //taking the no of rows & column for second matrices
8.   scanf("%d%d",&p,&q);
9.   printf("\n enter the matrix elements (quantity of the equipments and cost foreach equipment) ");
10.  for(i=0;i<m;i++);
11.  {
12.    for(j=0;j<n;j++);
13.    {
14.      scanf("%d",&a[i][j]);
15.    }
16.  }
17.  printf("\n a matrix is\n");           //displaying 1 Matrix
18.  for(i=0;i<m;i++)
19.  {
20.    for(j=0;j<n;j++)
21.    {
22.      printf("%d\t",a[i][j]);           //displaying 2 Matrix
23.    }
24.    printf("\n");
25.  }
26.  for(j=0;j<q;j++)
27.  {
28.    scanf("%d\t",&b[i][j]);
29.  }
30.  }
31.  printf("\n b matrix is\n");
32.  for(i=0;i<p;i++)
33.  {
34.    for(j=0;j<q;j++)
35.    {
36.      printf("%d\t",b[i][j]);
37.    }
38.    printf("\n");
39.  }
40.  for(i=0;i<m;i++)           //providing matrix multiplication

```

```

41. {
42. for(j=0;j<q;j++)
43. {
44. c[i][j]=0;
45. for(k=0;k<n;k++)
46. {
47. c[i][j]=c[i][j]+a[i][k]*b[k][j]; //calculating matrix multiplication
48. }
49. }
50. }
51. for(i=0;i<m;i++)
52.     for(j=0;j<q;j++)
53. {
54. printf("%d\t",c[i][j])(contribution cost of each sponsors ));    //displaying Matrix multiplication
55. }
56. printf("\n");
57. }

```

3.4 Expected Results

```

Enter 1st matrix no.of rows & cols
3 2
Enter 2nd matrix no.of rows & cols
2 2

Enter the matrix elements (quantity of the equipments and cost for
each equipment)

1 2
3 2
2 1

1 2
2 1

contribution cost of each sponsors
5 4
7 8
4 5

```

3.5 Implementation Phase : Execute the Program

Compile, remove syntax errors, if any, generate object code and make note of the same.

3.6	Simulate the Errors		
3.6.1	Syntax Error		
Sl.No.	Name of the Error	Cause for the Error	Rectification
3.6.2	Logical Error		
Sl.No.	Name of the Error	Cause for the Error	Rectification
1.	Segmentation faults	Inappropriate size of the array.	
4	Final Program and Results		

PROGRAM 2

1 Problem Statement

A Rifle salespersons in the former Arizona Territory sold rifle locks, stocks and barrels made by a gunsmith in Missouri. Locks cost \$45, Stocks cost \$30, and barrels cost \$25. The salesperson had to sell at-least one complete rifle per month, and production limits were such that the most the salesperson could sell in a month was 70 locks, 80 locks, and 90 barrels. At the end of the month, the salesperson sent a telegram showing -1 lock sold. The Smith then knew the sales for the month were complete and computed the salesperson's commission as follows: 10%` on sales up to \$1000, 15% on the next \$800, and 20% on any sales in excess of \$1800. Write a C program for above context and generate Test cases for same using data flow testing.

2 Student Learning Outcomes

After successful completion of this lab, the student's shall be able to

- i. Understand the concept of structural testing and the advantages of the same.
- ii. Identify the Significant of data flow testing, how it differs from basis path testing.
- iii. Generates set of define and usage node.
- iv. Demonstrate how data flow testing gives through/ complete testing of a system.

3 Design of the Program**3.1 Description**

Aims to execute sub-paths from points where each variable is defined to points where it is referenced. These sub-paths are called definition-use pairs, or du-pairs (du-paths, du-chains) Data flow testing is centered on variables ,Data flow testing follows the sequences of events related to a given data item with the objective to detect incorrect sequences It explores the effect of using the value produced by every and each computation.

Variable definition

Occurrences of a variable where a variable is given a new value (assignment, input by the user, input from a file, etc.) Variable DECLARATION is NOT its definition!!!

Variable uses

Occurrences of a variable where a variable is not given a new value (variable DECLARATION is NOT its use)

p-uses (predicate uses)

Occur in the predicate portion of a decision statement such as if-then-else, while-do etc.

c-uses (computation uses)

All others, including variable occurrences in the right hand side of an

assignment statement, or an output statement

du-path:A sub-path from a variable definition to its use.

Test case definitions based on four groups of coverage

– All definitions.

– All c-uses.

– All p-uses.

– **All du-paths.**

3.2 Algorithm

STEP 1: Define lprice=45.0, sprice=30.0, bprice=25.0

STEP2: Input locks

STEP3: while(locks!=-1) 'input device uses -1 to indicate end of data goto STEP 12

STEP4:input (stocks, barrels)

STEP5: compute lsales, ssales, bsales and sales

STEP6: output(" sales:" sales)

STEP7: if (sales > 1800.0) goto STEP 8 else goto STEP 9

STEP8: commission=0.10*1000.0; commission=commission+0.15 * 800.0;

commission = commission + 0.20 * (sales-1800.0)

STEP9: if (sales > 1000.0) goto STEP 10 else goto STEP 11

STEP10: commission=0.10* 1000.0; commission=commission + 0.15 * (sales-1000.0)

STEP11: Output("Commission is \$", commission)

STEP12: exit

Algorithm Description:

- Step 1 is used for variable declaration.
- Step 2 takes the input as lock.
- Step 3 checks whether the quantity of lock is within the range or not.
- Step 4 takes another input as stocks and barrels.
- Step 5 & 6 is used to compute the sales for locks stocks and barrels.
- Step 7, 8, 9 & 10 calculates the commission based on the sales.
- Step 11 displays the commission.
- Step 12 is used to stop the process.

3.3 Coding using C Language

```
1. #include<stdio.h>
2. int main()
3. {
4. int locks, stocks, barrels, tlocks, tstocks, tbarrels;      // variable declaration
5. float lprice,sprice,bprice,lsales,ssales,bsales,sales,comm; ;      // variable declaration
6. lprice=45.0; 8 sprice=30.0; 9 bprice=25.0; 10 tlocks=0; ;      // variable initialization
7. tstocks=0;      // variable initialization
8. tbarrels=0;      // variable initialization
9. printf("\n enter the number of locks and to exit the loop enter -1 for locks\n"); //taking input as locks
```

```
10. scanf("%d", &locks);
11. while(locks!=-1)                                // checking the lock value
12. {
13. printf("enter the number of stocks and barrels\n");    //taking input as stocks and barrels
14. scanf("%d%d",&stocks,&barrels);
15. tlocks=tlocks+locks;                                // calculationg total locks
16. tstocks=tstocks+stocks;                             // calculationg total stocks
17. tbarrels=btarrels+barrels;                          // calculationg total barrels
18. printf("\nenter the number of locks and to exit the loop enter -1 for locks\n");
19. scanf("%d",&locks);
20. }
21. printf("\ntotal locks = %d\n",tlocks);              // Displaying total locks
22. printf("total stocks =%d\n",tstocks);              // Displaying total stocks
23. printf("total barrels =%d\n",tbarrels);            // Displaying total barrels
24. lsales = lprice*tlocks;
25. ssales=sprice*tstocks;
26. bsales=bprice*tbarrels;
27. sales=lsales+ssales+bsales;
28. printf("\nthe total sales=%f\n",sales);            // Displaying total sales
29. if(sales > 1800.0)
30. {
31. comm=0.10*1000.0;
32. comm=comm+0.15*800;
33. comm=comm+0.20*(sales-1800.0);                    //calculating commission if sales is more than 1800
34. }
35. else if(sales > 1000)
36. {
37. comm =0.10*1000;
38. comm=comm+0.15*(sales-1000);                      //calculating commission if sales is more than 1000
39. }
40. else
41. comm=0.10*sales;                                    //calculating commission if sales is Within 1000
42. printf("the commission is=%f\n",comm); //Displaying commission
43. return 0;
44. }
```

3.4	Expected Results
Enter the total no of locks 10 Enter the total no of stocks 10 Enter the total no of barrels 10 The total sales is 1000 The commission is 100.000000 Enter the total no of locks 5 Enter the total no of stocks -1 Enter the total no of barrels 22 Invalid Input Enter the total no of locks 10 Enter the total no of stocks 10 Enter the total no of barrels 20 The total sales is 1250 The commission is 137.500000 Enter the total no of locks 20 Enter the total no of stocks 20 Enter the total no of barrels 20 The total sales is 2000 The commission is 260.000000	
3.5	Implementation Phase : Execute the Program Compile, remove syntax errors, if any, generate object code and make note of the same.
EXECUTION Execute the program and test the test cases in above Tables against program and complete the table with for Actual output column and Status column.	
3.6	Simulate the Errors
3.6.1	Syntax Errors

Sl.No.	Name of the Error	Cause for the Error	Rectification
1.	Compound statement missing	The compiler encountered an expression statement without a semicolon following it.	Use Semicolon properly wherever required.
2.	Undeclared Variable Name	Variable is not declared properly.	Variable needs to be declared before usage.
3.	Unmatched Parentheses	'}' expected	

3.6.2 Logical Error

4 Final Program and Results

SNAPSHOTS:

1. Snapshot for Total sales and commission when total sales are within 1000 and Invalid input

```

root@localhost:~
File Edit View Terminal Tabs Help
[root@localhost ~]# cc cfinal.c
[root@localhost ~]# ./a.out
Enter the total number of locks
10
Enter the total number of stocks
10
Enter the total number of barrelss
10
The total sales is 1000
The commission is 100.000000
[root@localhost ~]# cc cfinal.c
[root@localhost ~]# ./a.out
Enter the total number of locks
5
Enter the total number of stocks
-1
Enter the total number of barrelss
22
invalid input
[root@localhost ~]#

```

Enter the total no of locks

2.Invalid Input and Total sales and commission when total sales are within 1000

3. Snapshot for for Total sales and commission when total sales are within 1800 and to find out the total commission 360

PROGRAM 3

1	Problem Statement
	<p>Take any ATM system (e.g. SBI Bank ATM system) and study its system specifications and report the various bugs.</p> <ol style="list-style-type: none"> 14. Machine is accepting ATM card. 15. Machine is rejecting expired card. 16. Successful entry of PIN number. 17. Unsuccessful operation due to enter wrong PIN number 3 times. 18. Successful selection of language. 19. Successful selection of account type. 20. Unsuccessful operation due to invalid account type. 21. Successful selection of amount to be withdrawn. 22. Successful withdrawal. 23. Expected message due to amount is greater than day limit. 24. Unsuccessful withdraw operation due to lack of money in ATM. 25. Expected message due to amount to withdraw is greater than possible balance. 26. Unsuccessful withdraw operation due to click cancel after insert card. <p>Write a C program to Implement the above context, and generate test cases using decision table approach.</p>
2	Student Learning Outcomes
	<p>After successful completion of this lab, the student's shall be able to:</p> <ul style="list-style-type: none"> • Analyze the functionality of ATM machine's features. • Understand the working of Decision table approach. • Generate different test cases with pass and fail results.
3	Design of the Program
3.1	Description
	<ol style="list-style-type: none"> 1. Checking mandatory input parameters 2. Checking optional input parameters 3. Check whether able to create account entity. 4. Check whether you are able to deposit an amount in the newly created account (and thus updating the balance) 5. Check whether you are able to withdraw an amount in the newly created account (after deposit) (and thus updating the balance) 6. Check whether company name and its pan number and other details are provided in case of salary account 7. Check whether primary account number is provided in case of secondary account

8. Check whether company details are provided in cases of company's current account
9. Check whether proofs for joint account is provided in case of joint account
10. Check whether you are able deposit an account in the name of either of the person in a joint account.
11. Check whether you are able withdraws an account in the name of either of the person in a joint account.
12. Check whether you are able to maintain zero balance in salary account.
13. Check whether you are not able to maintain zero balance (or mini balance) in non-salary account.

3.2 Algorithm

Step 1: Start

Step 2: User accesses his account using Debit card through ATM machine with help of PIN.

Step 3: ATM machine reads this card and check it with Bank server.

Step 4: ATM waits to enter the transactions request.

Step 5: User may use ATM now and transact

Step 6: Stop

Algorithm description:

Step 1 is starting the execution of the program.

Step 2 takes the inputs as user's pin no

Step 3 verifies the pin no with the bank server for valid authentication.

Step 4 takes the transaction details like amount, type of account etc.

Step 5 user gets the money/ mini statements as requested.

Step 6 stops the process.

.

3.3 Coding using C Language :

```

1. #include <stdio.h>
2. unsigned long amount=1000, deposit, withdraw;      //variable declaration
3. int choice, pin, k;                                //variable declaration
4. char transaction='y';                              //variable declaration
5. void main()
6. {
7.     while(pin!=1111)                                //checking for the validity of the pin no
8.     {
9.         printf("enter your secret pin number"); //asking for valid pin no
10.        scanf("%d", &pin);
11.        if(pin!=1111)
12.        printf("please enter valid password");
13.    }

```

```
14. do
15. {
16. printf("*****welcome to ATM service*****\n");
17. printf("1.Check Balance \n");      //displaying the options
18. printf("2.Withdraw Cash \n");
19. printf("Deposit Cash \n");
20. printf("Go Back to Main Menu \n");
21. printf("enter your choice:");
22. scanf("%d", &choice);
23. switch(choice)
24. {
25. case 1:printf("\n your balance in RS: %lu",amount);    //requesting for the amount
26. break;
27. case 2: printf("Enter the amount to withdraw: ");      //amount is getting entered by user
28. scanf("%lu",withdraw);
29. if(withdraw%100!=0)                                     // checking for the denomination of 100
30. {
31. printf("\n please enter the amount in multiples of 100");
32. }
33. else if(withdraw>(amount-500))                          // checking for sufficient balance
34. {
35. printf("\n Insufficient Balance");
36. }
37. else
38. {
39.          amount=amount-withdraw;
40. printf("\n\n please collect cash");                    // withdrawn is successful
41. printf("\n your current balance is %lu", amount);
42. }
43. break;
44. case 3:printf("\n Enter the amount to be deposit");    // request for money deposite
45. scanf("%lu",&deposit);
46. amount=amount+deposit;
47. printf("Your Balance is %lu",amount);                  // displaying the current balance
48. break;
49. case 4:printf("\n thank U Using ATM" );
50. break;
51. default:printf ("Invalid Choice");
52. }
53.          printf("\n\n Do U Wish To Have Another transaction?(Y/N):\n "); // asking for further
                                                    transaction
54. fflush(stdin);
55. scanf("%c", &transaction);
56. if(transaction== 'n' || transaction== 'N')
57. k=1;
58. }while(!k);
```

```
59. printf("\n\n Thanks For Using Out ATM Service");
60. }
```

3.4 Expected Results

Enter the pin no:
1122
Enter your secret no:
2434
*****welcome to ATM service*****
1.Check Balance
2.Withdraw Cash
3.Deposite Cash
4.Go Back to Main Menu
Enter your choice: 1
Your balance is 11000.00
Enter the amount to withdraw: 500
Please collect the cash:5000
Your current balance is 6000

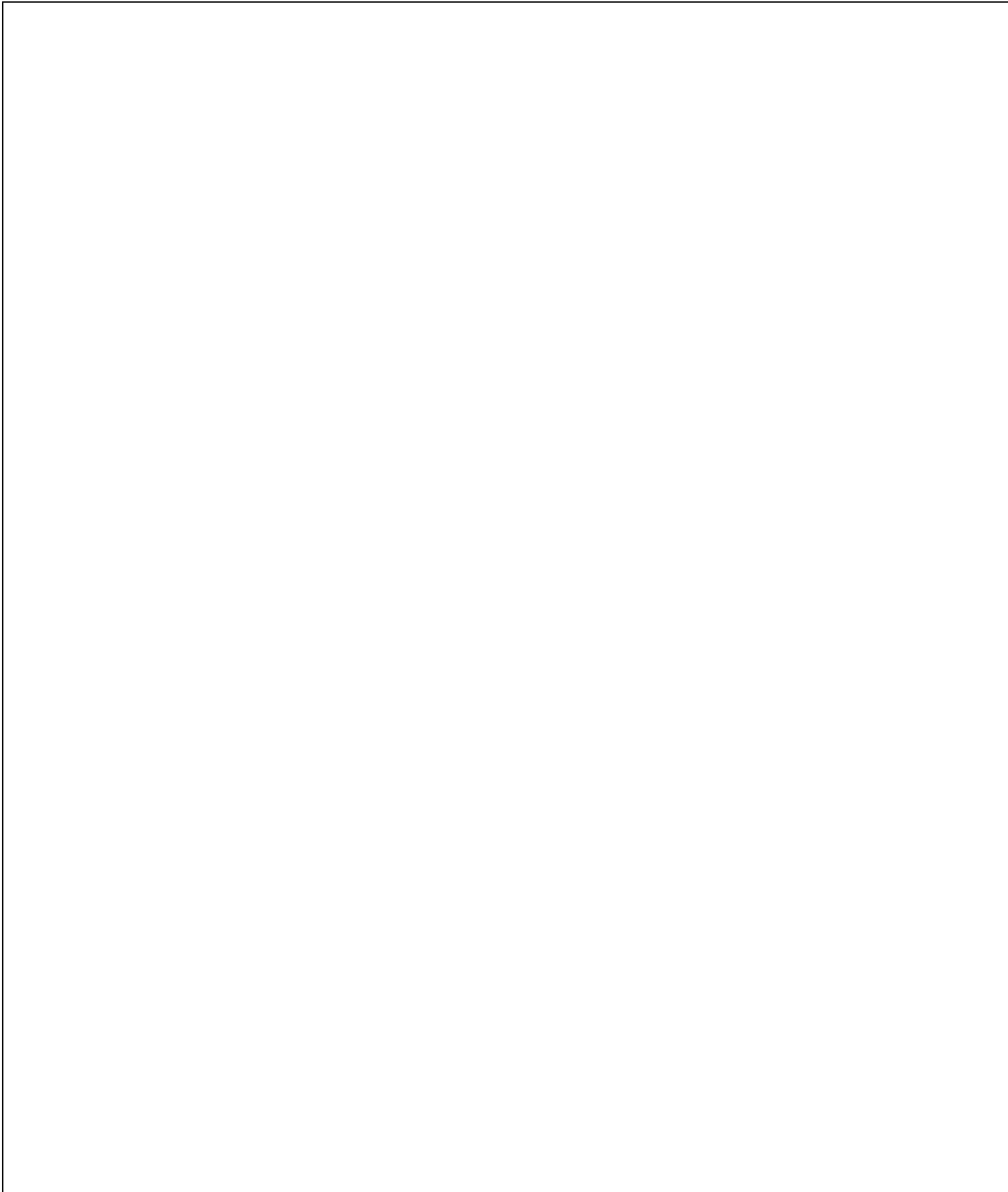
3.5 Implementation Phase: Execute the Program Compile, remove syntax errors, if any, generate object code and make note of the same.

3.6 Simulate the Errors

3.6.1 Syntax Errors

Sl No	Name of the Error	Cause for the Error	Rectification
1.	Undeclared variables	Variable that hasn't been declared.	

2.	Missing Semicolon	Statement is not exitedproperly	
3.6.2	Logical Errors		
SI No	Name of the Error	Cause for the Error	Rectification
1.	Bad mathematical calculation.		
4.	Final Program and Results		



PROGRAM 4

1	Problem Statement
1. 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. Design and develop a program in a language of your choice to solve the triangle problem defined as follows: Derive test cases for your program based on decision-table approach, execute the test cases and discuss the results.	
2	Student Learning Outcomes
After successful completion of this lab, the student's shall be able to: <ul style="list-style-type: none"> • Evaluate the types of triangle based on given input. • Understand the working of decision table approach. • Derive set of test cases based on decision table approach. 	
3	Design of the Program
3.1	Description
<p>REQUIREMENTS:</p> <p>R1. The system should accept 3 positive integer numbers (a, b, c) which represents 3 sides of the triangle. Based on the input it should determine if a triangle can be formed or not.</p> <p>R2. If the requirement R1 is satisfied then the system should determine the type of the triangle, which can be</p> <ul style="list-style-type: none"> • Equilateral (i.e. all the three sides are equal) • Isosceles (i.e Two sides are equal) • Scalene (i.e All the three sides are unequal) <p>else suitable error message should be displayed. Here we assume that user gives three positive integer numbers as input.</p> <p>Form the given requirements we can draw the following conditions:</p> <p>C1: $a < b + c$?</p> <p>C2: $b < a + c$?</p> <p>C3: $c < a + b$?</p> <p>C4: $a = b$?</p> <p>C5: $a = c$?</p> <p>C6: $b = c$?</p> <p>According to the property of the triangle, if any one of the three conditions C1, C2 and C3 are not satisfied then triangle cannot be constructed. So only when C1, C2 and C3 are true the triangle can be formed, then depending on conditions C4, C5 and C6 we can decide what type of triangle will be formed. (i.e requirement R2).</p>	
3.2	Algorithm

Step 1: Input a, b & c i.e three integer values which represent three sides of the triangle.

Step 2: if (a < (b + c)) and (b < (a + c)) and (c < (a + b)) then do step 3

Else print not a triangle. do step 6.

Step 3: if (a=b) and (b=c) then

Print triangle formed is equilateral. do step 6.

Step 4: if (a ≠ b) and (a ≠ c) and (b ≠ c) then

Print triangle formed is scalene. do step 6.

Step 5: Print triangle formed is Isosceles.

Step 6: stop

Algorithm Description:

Step 1 declares all the variables as the sides of the triangle.

Step 2 checks the possibility of formation of the triangle.

Step 3 displays the equilateral triangle

Step 4 displays the scalene triangle

Step 5 displays the Isosceles triangle

Step 6 stop the process.

3.3 Coding using C Language

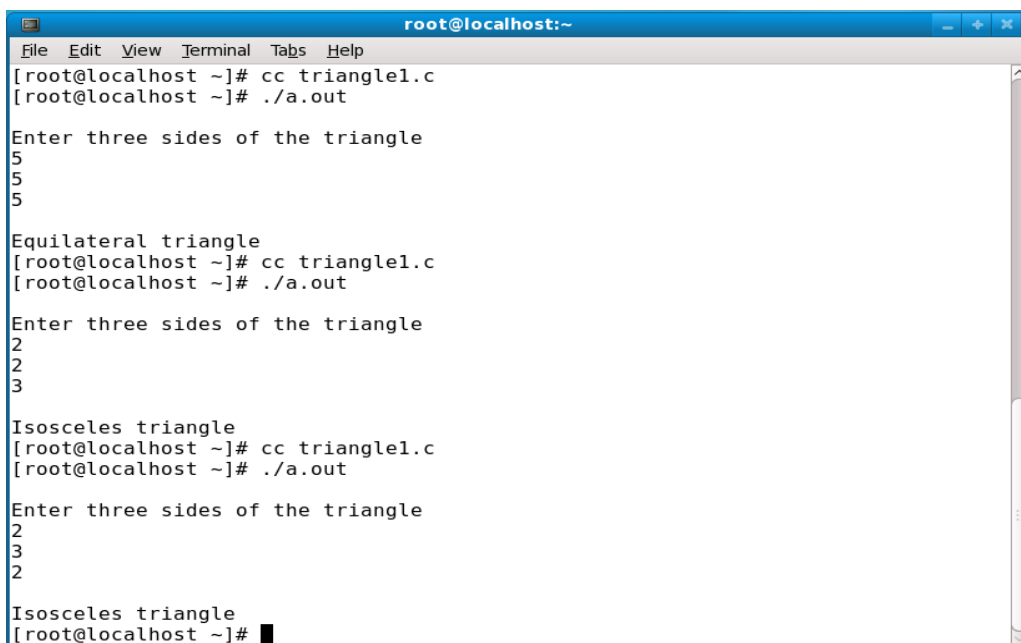
Program Description:

```
1. #include<stdio.h>
2. int main()
3. {
4.   int a, b, c;
5.   printf("Enter three sides of the triangle");    //taking the input as the three sides f a triangle
6.   scanf("%d%d%d", &a, &b, &c);
7.   if((a<b+c)&&(b<a+c)&&(c<a+b))                  //checking the validity of a triangle
8.   {
9.     if((a==b)&&(b==c))
10.    {
11.      printf("Equilateral triangle");              //displaying the equilateral triangle
12.    }
13.   else if((a!=b)&&(a!=c)&&(b!=c))                //displaying the scalene triangle
14.   {
15.     printf("Scalene triangle");
16.   }
17.   else
18.     printf("Isosceles triangle");                  //displaying the isoceles triangle
19.   }
20.   else
21.   {
22.     printf("triangle cannot be formed");
23.   }
24.   return 0;
```

25. }

3.4 Expected Results

Enter three sides of the triangle
5
5
5
Equilateral triangle
Enter three sides of the triangle
2
2
3
Isosceles triangle
Enter three sides of the triangle
5
6
8
Scalene triangle
Enter three sides of the triangle
1
2
5
Triangle can't be formed



```
root@localhost:~  
File Edit View Terminal Tabs Help  
[root@localhost ~]# cc triangle1.c  
[root@localhost ~]# ./a.out  
  
Enter three sides of the triangle  
5  
5  
5  
Equilateral triangle  
[root@localhost ~]# cc triangle1.c  
[root@localhost ~]# ./a.out  
  
Enter three sides of the triangle  
2  
2  
3  
Isosceles triangle  
[root@localhost ~]# cc triangle1.c  
[root@localhost ~]# ./a.out  
  
Enter three sides of the triangle  
2  
3  
2  
Isosceles triangle  
[root@localhost ~]#
```

3.5 Implementation Phase : Execute the Program

Compile, remove syntax errors, if any, generate object code and make note of the same.

Technique Used: Decision Table Approach

Decision Table-Based Testing has been around since the early 1960's; it is used to depict complex logical relationships between input data. A Decision Table is the method used to build a complete set of test cases without using the internal structure of the program in question. In order to create test cases we use a table to contain the input and output values of a program.

3.6 Simulate the Errors**3.6.1 Syntax Errors**

Sl no	Name of Error	Cause for the Error	Rectification
1.	Compound statement missing	The compiler encountered an expression statement without a semicolon following it.	Use Semicolon properly wherever required.
2.	Undeclared Variable Name	Variable is not declared properly.	Variable needs to be declared before usage.
3.	Unmatched Parentheses	'}' expected	

3.6.2 Logical Errors

Sl no	Name of Error	Cause for the Error	Rectification
1.	Usage of condition for evaluating triangle types	<p>Instead of using</p> $a < b + c \&\& (b < a + c) \&\& (c < a + b)$ <p>The following condition is used.</p> $a < = b + c \&\& (b < = a + c) \&\& (c < = a + b)$	

4	Final Program and Results
----------	----------------------------------

PROGRAM 5

1	Problem Statement
<p>1. Pickup the nearest phonebook and open it to the first page of names. We're looking to find the first "Smith". Look at the first name. Is it "Smith"? Probably not (it's probably a name that begins with 'A'). Now look at the next name. Is it "Smith"? Probably not. Keep looking at the next name until you find "Smith". You started at the beginning of a sequence and went through each item one by one, in the order they existed in the list, until you found the item you were looking for. Design and develop the code for above context in C language. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.</p>	
2	Student Learning Outcomes
<p>After successful completion of this lab, the student shall be able to</p> <ul style="list-style-type: none"> • Understand the concept of linear search. • Generates set of feasible test cases. 	
3	Design of the Program
3.1	Description
<p>Linear Search: A linear search searches an element or value from an array till the desired element or value is not found and it searches in a sequence order. It compares the element with all the other elements given in the list and if the element is matched it returns the position. Linear Search is applied on the unsorted or unordered list when there are fewer elements in a list.</p>	
3.2	Algorithm
<p>Linear Search (Array XYZ, Value id)</p> <p>Step 1: Set i to 0 and n to the number of elements in array</p> <p>Step 2: IF $i > n$ then go to step 7</p> <p>Step 3: IF $XYZ[i] = id$ then go to step 6</p> <p>Step 4: Set i to $i + 1$</p> <p>Step 5: Go to Step 2</p> <p>Step 6: Print Element id Found at index i and go to step 8</p> <p>Step 7: Print element not found</p> <p>Step 8: Exit</p> <p>Algoritihm Description</p> <ul style="list-style-type: none"> • Line 1 includes header files using preprocessor directives • Line 4 declares different variables and array 	

- Lines 5 and 6 read the number of employees in the array
- Lines 7 to 9 read the employee ids
- Lines 10 and 11 read the id to be searched
- Lines 12 to 20 search for the employee id in the array and prints its location if found
- Lines 21 to 22 print “employee is not present in array” if id does not exist in the array

3.3 Coding using C Language

```

1. #include <stdio.h>
2. #include<string.h>
3. int main()
4. {
5.   char name[100][50], search[30];
6.   int c, n;
7.   printf("Enter the number of elements in array\n");
8.   scanf("%d",&n);
9.   printf("Enter %d names\n", n);
10.  for (c = 0; c < n; c++)
11.    scanf("%s", name[c]);
12.  printf("Enter the name to search\n");
13.  scanf("%s", search);
14.  for (c = 0; c < n; c++)
15.  {
16.    if ((strcmp(name[c],search)==0)) /* if required element found */
17.    {
18.      printf("%s is present at location %d.\n", search, c+1);
19.      break;
20.    }
21.  }
22.  if (c == n)
23.    printf("%s is not present in phonebook.\n", search);
24.  return 0;
25. }

```

3.4 Expected Results

```

Enter the number of elements in array
4
Enter 4 names
Smith
John
Ron
Jack

```

Enter the name to search
 Smith
 Smith is present at location 1.

Enter the number of elements in array
 2
 Enter 4 names
 John
 Ron
 Enter the name to search
 Smith
 Smith is not present in phonebook.

3.5 **Implementation Phase : Execute the Program**
Compile, remove syntax errors, if any, generate object code and make note of the same.

3.6 **Simulate the Errors**

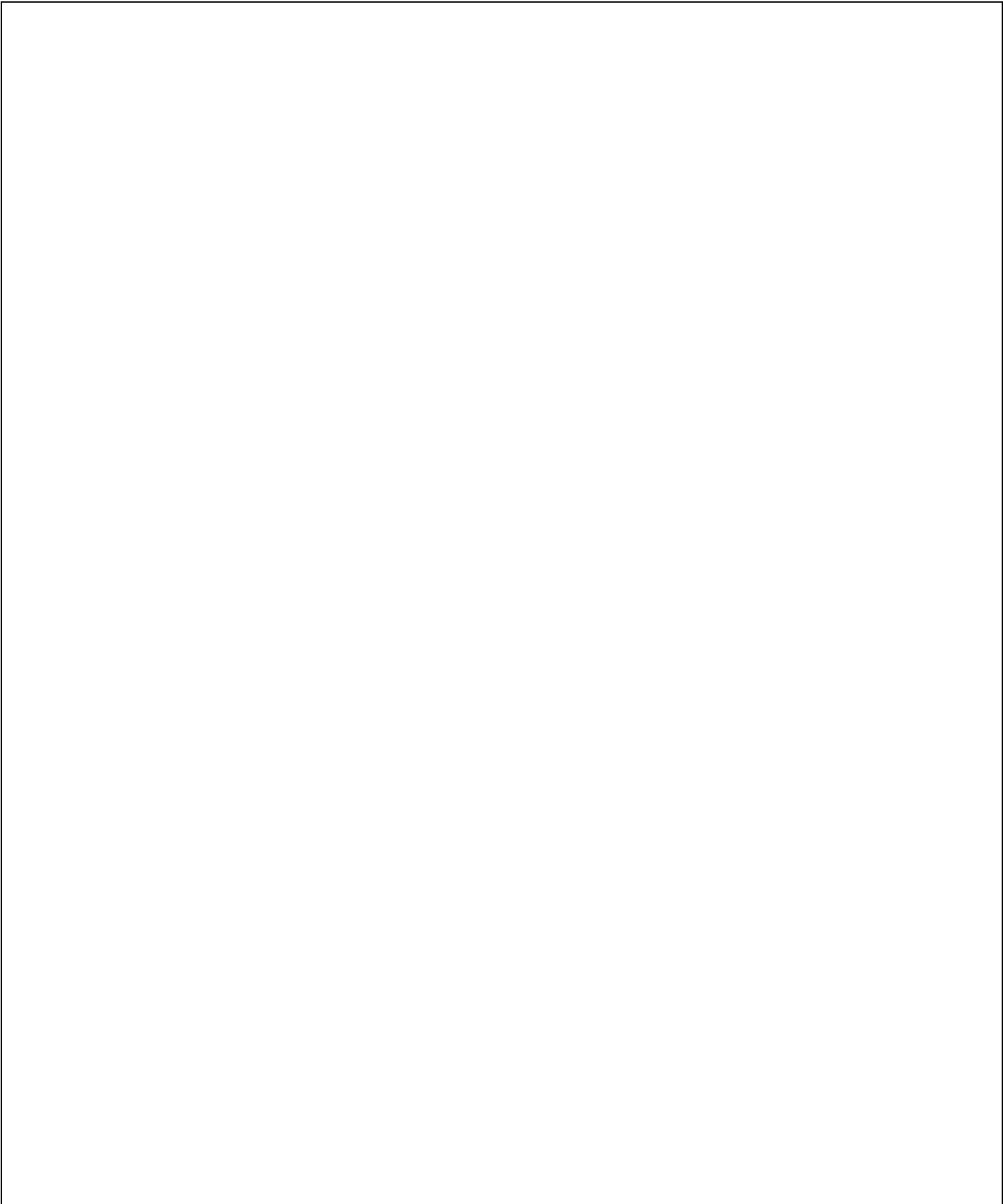
3.6.1 **Syntax Errors**

Sl no	Name of Error	Cause for the Error	Rectification

3.6.2 **Logical Errors**

Sl No	Name of Error	Cause for the Error	Rectification

4 **Final Program and Results**



PROGRAM 6

1	Problem Statement
	<p>Suppose you have a list of 10 things and you have two friends and you ask both the friends to order the things in descending order of their liking. You will have two arrays after that, now you count the number of inversion they have and more the number of inversion, the more dissimilar their choices are, less the number of inversions the more similar their choices are. So when you sort an array using merge sort, what you do is you put the array you want to obtain on the top and the array you are starting off with at the bottom and connect all the numbers respectively. The number of intersections you obtain, will be your number of inversions. The maximum number of inversions you can have in an array is $n(n-1)/2$, where (n) is the size of the array and that is when your array is in the reverse order of the order in which you want to arrange it.</p>
2	Student Learning Outcomes
	<p>After successful completion of this lab, the student will able to</p> <ul style="list-style-type: none"> • Understand the concept of divide and conquer and its application merge sort . • Generates set of feasible test cases.
3	Design of the Program
3.1	<p>Description</p> <p>Merge sort is based on the divide-and-conquer paradigm. Its worst-case running time has a lower order of growth than insertion sort. Since we are dealing with subproblems, we state each subproblem as sorting a subarray $A[p \dots r]$. Initially, $p = 1$ and $r = n$, but these values change as we recurse through subproblems.</p> <p>To sort $A[p \dots r]$:</p> <p>1. Divide Step</p> <p>If a given array A has zero or one element, simply return; it is already sorted. Otherwise, split $A[p \dots r]$ into two subarrays $A[p \dots q]$ and $A[q + 1 \dots r]$, each containing about half of the elements of $A[p \dots r]$. That is, q is the halfway point of $A[p \dots r]$.</p> <p>2. Conquer Step</p> <p>Conquer by recursively sorting the two subarrays $A[p \dots q]$ and $A[q + 1 \dots r]$.</p> <p>3. Combine Step</p> <p>Combine the elements back in $A[p \dots r]$ by merging the two sorted subarrays $A[p \dots q]$ and $A[q + 1 \dots r]$ into a sorted sequence. To accomplish this step, we will define a procedure MERGE (A, p, q, r).</p>

3.2 Algorithm

To sort the entire sequence $A[1 \dots n]$, make the initial call to the procedure MERGE-SORT ($A, 1, n$).

MERGE-SORT (A, p, r)

1. IF $p < r$ // Check for base case
2. THEN $q = \text{FLOOR}[(p + r)/2]$ // Divide step
3. MERGE (A, p, q) // Conquer step.
4. MERGE ($A, q + 1, r$) // Conquer step.
5. MERGE (A, p, q, r) // Conquer step.

MERGE (A, p, q, r)

1. $n_1 \leftarrow q - p + 1$
2. $n_2 \leftarrow r - q$
3. Create arrays $L[1 \dots n_1 + 1]$ and $R[1 \dots n_2 + 1]$
4. FOR $i \leftarrow 1$ TO n_1
5. DO $L[i] \leftarrow A[p + i - 1]$
6. FOR $j \leftarrow 1$ TO n_2
7. DO $R[j] \leftarrow A[q + j]$
8. $L[n_1 + 1] \leftarrow \infty$
9. $R[n_2 + 1] \leftarrow \infty$
10. $i \leftarrow 1$
11. $j \leftarrow 1$
12. FOR $k \leftarrow p$ TO r
13. DO IF $L[i] \leq R[j]$
14. THEN $A[k] \leftarrow L[i]$
15. $i \leftarrow i + 1$
16. ELSE $A[k] \leftarrow R[j]$
17. $j \leftarrow j + 1$

Algorithm Description:**MERGE-SORT**

Step 1: Divide the Array A into two parts and go on calling merge procedure

Step 2: Conquer the divided array into sorted array

MERGE

Step 1: The following is the input and output of the MERGE procedure.

Step 2: INPUT: Array A and indices p, q, r such that $p \leq q \leq r$ and subarray $A[p \dots q]$ is sorted and subarray $A[q + 1 \dots r]$ is sorted. By restrictions on p, q, r , neither subarray is empty.

Step 3: OUTPUT: The two subarrays are merged into a single sorted subarray in $A[p \dots r]$.

3.3 Coding using C language

```
1. #include <stdio.h>
2. #include<stdlib.h>

3. int _mergeSort(int arr[], int temp[], int left, int right);
4. int merge(int arr[], int temp[], int left, int mid, int right);

5. /* This function sorts the input array and returns the
6. number of inversions in the array */
7. int mergeSort(int arr[], int array_size)
8. {
9.     int *temp = (int *)malloc(sizeof(int)*array_size);
10.    return _mergeSort(arr, temp, 0, array_size - 1);
11. }

12. /* An auxiliary recursive function that sorts the input array and
13. returns the number of inversions in the array. */
14. int _mergeSort(int arr[], int temp[], int left, int right)
15. {
16.     int mid, inv_count = 0;
17.     if (right > left)
18.     {
19.         /* Divide the array into two parts and call _mergeSortAndCountInv()
20.         for each of the parts */
21.         mid = (right + left)/2;

22.         /* Inversion count will be sum of inversions in left-part, right-part
23.         and number of inversions in merging */
24.         inv_count = _mergeSort(arr, temp, left, mid);
25.         inv_count += _mergeSort(arr, temp, mid+1, right);

26.         /*Merge the two parts*/
27.         inv_count += merge(arr, temp, left, mid+1, right);
28.     }
29.     return inv_count;
30. }

31. /* This funt merges two sorted arrays and returns inversion count in
32. the arrays.*/
33. int merge(int arr[], int temp[], int left, int mid, int right)
34. {
35.     int i, j, k;
36.     int inv_count = 0;
```

```
37. i = left; /* i is index for left subarray*/
38. j = mid; /* i is index for right subarray*/
39. k = left; /* i is index for resultant merged subarray*/
40. while ((i <= mid - 1) && (j <= right))
41. {
42. if (arr[i] <= arr[j])
43. {
44. temp[k++] = arr[i++];
45. }
46. else
47. {
48. temp[k++] = arr[j++];
49. inv_count = inv_count + (mid - i);
50. }
51. }

52. /* Copy the remaining elements of left subarray
53. (if there are any) to temp*/
54. while (i <= mid - 1)
55. temp[k++] = arr[i++];
56. /* Copy the remaining elements of right subarray
57. (if there are any) to temp*/
58. while (j <= right)
59. temp[k++] = arr[j++];
60. /*Copy back the merged elements to original array*/
61. for (i=left; i <= right; i++)
62. arr[i] = temp[i];
63. return inv_count;
64. }
65. /* Driver progra to test above functions */
66. int main(int argv, char** args)
67. {
68. int c,m,arr[10] ; //={1, 20, 6, 4, 5};
69. printf("Input number of things \n");
70. scanf("%d", &m);
71. printf("Input value for number of things %d \n",m);
72. for (c = 0; c < m; c++)
73. {
74. scanf("%d", &arr[c]);
75. }
76. printf(" Number of inversions are %d \n", mergeSort(arr, 5));
77. getchar();
78. return 0;
79. }
```

3.4	Expected Results		
<div>Input number of things</div> <div>10</div> <div>Input value for number of things 10</div> <div>9</div> <div>8</div> <div>7</div> <div>6</div> <div>5</div> <div>4</div> <div>3</div> <div>2</div> <div>1</div> <div>0</div> <div>Number of inversions are 10</div>			
3.5	Implementation Phase : Execute the Program Compile, remove syntax errors, if any, generate object code and make note of the same.		
3.6	Simulate the Errors		
3.6.1	Syntax Errors		
SI No	Name of the Error	Cause for the Error	Rectification

3.6.2	Logical Errors		
SI No	Name of the Error	Cause for the Error	Rectification
4	Final Program and Results		

PROGRAM 7

1 Problem Statement

Assume that the marks obtained by a student in a test of 100 marks and compute his grade according to the following criteria.

Marks \geq 80 grade=A

Marks \geq 70 & $<$ 80 grade=B

Marks \geq 60 & $<$ 70 grade=C

Marks \geq 50 & $<$ 60 grade=D

otherwise grade=F

Design a code in C language for the above context. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.

2 Student Learning Outcomes

After successful completion of this lab, the student shall be able to

-

3 Design of the Program**3.1 Description**

The system should accept marks of 6 subjects, each marks in the range 1 to 100.

i.e., for example, $1 \leq \text{marks} \leq 100$

$1 \leq \text{kannada} \leq 100$

$1 \leq \text{maths} \leq 100$ etc.

If R1 is satisfied average of marks scored and percentage of the same and depending on percentage display the grade.

3.2 Algorithm

Step 1: start

Step 2: take the input as marks of different subjects like kanada, english, hindi, maths, science, sst.

Step 3: calculate the average marks.

Step 4: check the grade based on user input.

Step 5: display the grade.

Step 5: stop

Algorithm Description:

Step 1 is the beginning of the program

Step 2 is for taking the inputs.

Step 3 for calculating the average marks

Step 4 checks the input.

Step 5 displays the geade

Step 6 stop the process.

3.3 Coding Using C

1. `#include<stdio.h>`

2. `main()`

3. `{`

```
4. float kan,eng,hindi,maths,science, sst,avmar;
5. printf("Letter Grading:\n");
6. printf("SSLC Marks Grading:\n");
7. printf("Enter the marks for Kannada:");
8. scanf("%f",&kan);
9. printf("enter the marks for English:");
10. scanf("%f",&eng);
11. printf("enter the marks for Hindi:");
12. scanf("%f",&hindi);
13. printf("enter the marks for Maths");
14. scanf("%f",&maths);
15. scanf("%f",&sst);
16. avmar=(kan+eng+hindi+maths+science+sst)/6.00;
17. printf("the average marks are=%f\n",avmar);
18. if((avmar<50)&&(avmar>0))
19. printf("fail");
20. else
21. if((avmar>=50)&&(avmar<60))
22. printf("Grade D");
23. else
24. if((avmar>=60)&&(avmar<70))
25. printf("Grade C");
26. else
27. if((avmar>=70)&&(avmar<80))
28. printf("Grade B");
29. else
30. if((avmar>80))
31. printf("Grade A");
32.
33. printf("enter the marks for Science:");
34. scanf("%f",&science);
35. printf("enter the marks for Social Science:");
36. }
```

3.4 Expected Results

Letter grading
SSLC marks grading:
Enter the marks for kannada:30
Enter the marks for English: 30
Enter the marks for Hindi: 30
Enter the marks for maths: 35
Enter the marks for science: 35
Enter the marks for social science: 35
The average marks are=32.500000
Grade fail

Letter grading
SSLC marks grading:
Enter the marks for kannada:50
Enter the marks for English: 50
Enter the marks for Hindi: 50
Enter the marks for maths: 5
Enter the marks for science: 55
Enter the marks for social science: 55
The average marks are=52.500000
Grade 'D'

Letter grading
SSLC marks grading:
Enter the marks for kannada:65
Enter the marks for English: 65
Enter the marks for Hindi: 65
Enter the marks for maths: 65
Enter the marks for science: 65
Enter the marks for social science: 65
The average marks are=62.40002
Grade 'C'

Letter grading
SSLC marks grading:
Enter the marks for kannada:75
Enter the marks for English: 75
Enter the marks for Hindi: 75
Enter the marks for maths: 75
Enter the marks for science: 65
Enter the marks for social science: 65
The average marks are=71.666666
Grade 'B'

Letter grading
SSLC marks grading:
Enter the marks for kannada:85
Enter the marks for English: 85
Enter the marks for Hindi: 75
Enter the marks for maths: 75

Enter the marks for science: 85
 Enter the marks for social science: 85
 The average marks are=81.666666
 Grade 'A'

3.5 Implementation Phase: Compile the Program and note the syntax errors/warning if occurred during each compilation. Remove those syntax errors/warnings for generation of executable file.

3.6 Simulate the Errors/warnings

3.6.1 Syntax Errors/warnings

Sl.No	Error/Warning Simulated	Reflection of the Error/Warning on Compilation/Output
Semicolon missing	The compiler encountered an expression statement without a semicolon following it.	
Undeclared Variable Name	Variable is not declared properly.	
Unmatched Parentheses)' expected	

3.6.2 Logical Errors

Sl.No	Error/Warning Simulated	Reflection of the Error/Warning on Compilation/Output

4 Final Program and Results

PROGRAM 8**1 Problem Statement**

In the game of "twenty questions", your task is to guess the value of a hidden number that is one of the N integers between 0 and $N-1$. (For simplicity, we will assume that N is a power of two.) Each time that you make a guess, you are told whether your guess is too high or too low. An effective strategy is to maintain an interval that contains the hidden number, guess the number in the middle of the interval, and then use the answer to halve the interval size. Develop a C Code which suits the above context. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.

2 Student Learning Outcomes

After successful completion of this lab, the student shall be able to

- Understand the concept of binary search.
- Understand why binary search is better than linear search
- Generates set of feasible test cases.

3	Design of the Program
3.1	Description Binary Search: Binary Search is applied on the sorted array or list . Binary search first compares the value with the elements in the middle position of the array. If the value is matched, then returns the position. If the value is less than the middle element, then it must lie in the lower half of the array and if it's greater than the element then it must lie in the upper half of the array. We repeat this procedure on the lower (or upper) half of the array. Binary Search is useful when there are large numbers of elements in an array.
3.2	Algorithm Binary Search(Array XYZ, Value id) Step 1: Set n to the number of elements in array, $low \leftarrow 0$ and $high \leftarrow n-1$ Step 2: Set $mid \leftarrow \lfloor (low+high)/2 \rfloor$ Step 3: While $low \leq high$, Repeat Step 4 to 7 Step 4: IF $id = XYZ[mid]$ Print Element id Found at index mi Step 5: ELSE IF $id < XYZ[mid]$ Set $high \leftarrow mid-1$ Step 6: ELSE Set $low \leftarrow mid+1$ Step 7: Set $mid \leftarrow \lfloor (low+high)/2 \rfloor$ Step 8: Exit Algorithm Description <ul style="list-style-type: none"> • Step 1 includes n number of elements, set $low = 0$ and $high = n-1$ • Step 2 evaluates mid value from low and high value • Step 3 it repeats step 4 to 7 till $low \leq high$ • Step 4 to 7 : compares mid value with key value, if found returns else search in left or right part.
3.3	Coding using C language <pre> 1. #include <stdio.h> 2. int main() 3. { 4. int c, first, last, middle, n, search, array[100]; 5. printf("Enter number of elements\n"); 6. scanf("%d",&n); 7. printf("Enter %d integers\n", n); 8. for (c = 0; c < n; c++) 9. scanf("%d",&array[c]); 10. printf("Enter the guess value\n"); 11. scanf("%d", &search); 12. first = 0; 13. last = n - 1; 14. middle = (first+last)/2; </pre>

```
15. while (first <= last)
16. {
17. if (array[middle] < search)
18. first = middle + 1;
19. else if (array[middle] == search)
20. {
21. printf("guess %d found at location %d.\n", search, middle+1);
22. break;
23. }
24. else
25. last = middle - 1;
26. middle = (first + last)/2;
27. }
28. if (first > last)
29. printf("guess Not found! %d is not present in the list.\n", search);
30. return 0;
31. }
```

3.4 Expected Results

Enter the number of elements

7

Enter 7 integers

45

56

67

78

89

90

98

Enter the guess value

78

Guess 78 found at location 4

3.5	Implementation Phase: Compile the Program and note the syntax errors/warning if occurred during each compilation. Remove those syntax errors/warnings for generation of executable file.																
3.6	Simulate the Errors/warnings																
3.6.1	Syntax Errors/warnings																
<table><tr><td>Sl.No</td><td>Error/Warning Simulated</td><td>Reflection of the Error/Warning on Compilation/Output</td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>			Sl.No	Error/Warning Simulated	Reflection of the Error/Warning on Compilation/Output												
Sl.No	Error/Warning Simulated	Reflection of the Error/Warning on Compilation/Output															
3.6.2	Logical Errors																
<table><tr><td>Sl.No</td><td>Error/Warning Simulated</td><td>Reflection of the Error/Warning on Compilation/Output</td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>			Sl.No	Error/Warning Simulated	Reflection of the Error/Warning on Compilation/Output												
Sl.No	Error/Warning Simulated	Reflection of the Error/Warning on Compilation/Output															
4	Final Program and Results																

PROGRAM 9**1 Problem Statement**

Given a Binary Number as string, print its 1's and 2's complements. 1's complement of a binary number is another binary number obtained by toggling all bits in it, i.e., transforming the 0 bit to 1 and the 1 bit to 0. Design a C program to Implement above context and test the same using Decision table approach. Derive different test cases, execute these test cases and discuss the test results.

2 Student Learning Outcomes

After successful completion of this lab, the student shall be able to:

- Understand the concept of 1's and 2's complement of a binary number.
- Generates set of feasible test cases.

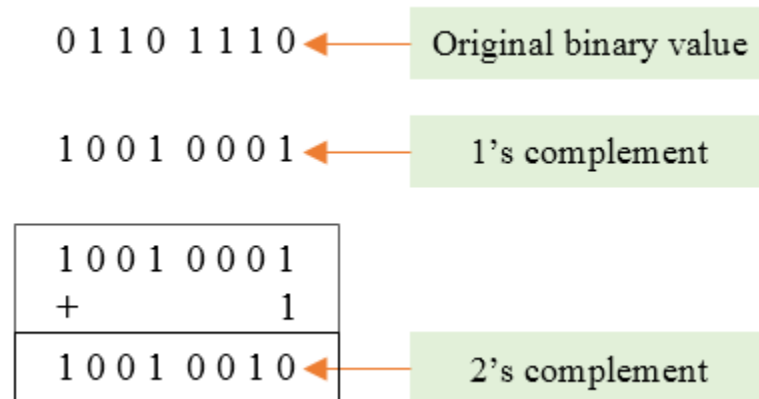
3 Design of the Program

3.1 Description

Two's complement

Wikipedia states that the two's complement of an N-bit number is defined as the complement with respect to 2^N ; in other words, it is the result of subtracting the number from 2^N , which in binary is one followed by N zeroes.

Or in simple words two's complement can be also defined as sum of 1's complement of the binary number and 1.

**3.2 Algorithm:**

Main

Step 1: Start

Step 2: declare the subprogram "complement(char *a)"

Step 3: initialize the variable i

Step 4: read the binary number

Step 5: perform the loop operation. if it is true then follows. if not goto step 7

i) for(i = 0; a[i] != '\0'; i++)

ii) if(a[i] != '0' && a[i] != '1') then displayed the number is not valid. enter the correct number.

iii) Exit the loop

Step 6: call sub program 'complement(a)'

Step 7: Stop

Sub Program:

Step 1: initialize the variable l, c = 0, b[160]

Step 2: l = strlen(a)

Step 3: perform the loop operation. if it is true then follows. if not goto

i) for(i = l - 1; i >= 0; i--)

ii) if(a[i] == '0') then b[i] = '1' else

iii) b[i] = '0'

Step 4: for(i = l - 1; i >= 0; i--) is true

i) if(i == l - 1) then

ii) if(b[i] == '0') then b[i] = '1' else

iii) b[i] = '0', c = 1 if not goto step 5

Step 5: if(c == 1 && b[i] == '0') istrue then
 i)b[i] = '1', c=0 if not goto Step 6
 Step 6: if(c == 1 && b[i] == '1') then b[i] = '0', c = 1
 Step 7: displayed b[l] = '\0'
 Step 8: print b and return to main program

Algorithm Description

Step 1: Start

Step 2: declare the subprogram "complement(char *a)"

Step 3: initialize the variable l ,read the binary numberperform the loop operation. if it istrue then follows. if not stop

Sub Program:

Step 1: perform the loop operation. if it istrue then follows. if not goto

Step 2: for(i = l - 1; i >= 0; i--) istrue

 i) if(i == l - 1) then

 ii) if(b[i] == '0') then b[i] = '1' else

 iii)b[i] = '0', c = 1 if not goto step 3

Step 3: if(c == 1 && b[i] == '0') istrue then

 i)b[i] = '1', c=0 if not goto Step 6

Step 4: if(c == 1 && b[i] == '1') then b[i] = '0', c = 1

Step 5: displayed b[l] = '\0'

Step 6: print b and return to main program

3.3 Coding using C language

```
1. #include <stdio.h>
2. #include<stdlib.h>
3. #include<string.h>

4. void complement (char *a);
5. void main()
6. {
7. char a[16];
8. int i;

9. printf("Enter the binary number");
10. scanf("%s",a);
11. for(i=0;a[i]!='\0'; i++)
12. {
13. if (a[i]!='0' && a[i]!='1')
14. {
15. printf("The number entered is not a binary number. Enter the correct number");
16. exit(0);
17. }
18. }
19. complement(a);
```



```
20. }
21. void complement (char *a)
22. {
23. int l, i, c=0;
24. char b[16];
25. l=strlen(a);
26. for (i=l-1; i>=0; i--)
27. {
28. if (a[i]=='0')
29. b[i]='1';
30. else
31. b[i]='0';
32. }
33. printf("The 1's complement is %s\n", b);

34. for(i=l-1; i>=0; i--)
35. {
36. if(i==l-1)
37. {
38. if (b[i]=='0')
39. b[i]='1';
40. else
41. {
42. b[i]='0';
43. c=1;
44. }
45. }
46. else
47. {
48. if(c==1 && b[i]=='0')
49. {
50. b[i]='1';
51. c=0;
52. }
53. else if (c==1 && b[i]=='1')
54. {
55. b[i]='0';
56. c=1;
57. }
58. }
59. }
60. b[l]='\0';
61. printf("The 2's complement is %s", b);
62. }
```

3.4 Expected Results

	<p>Enter the binary number</p> <p>100101</p> <p>The 1's compliment of binary number is</p> <p>011010</p> <p>The 2's compliment of binary number is</p> <p>011011</p>															
3.5	Implementation Phase: Compile the Program and note the syntax errors/warning if occurred during each compilation. Remove those syntax errors/warnings for generation of executable file.															
3.6	Simulate the Errors/warnings															
3.6.1	Syntax Errors/warnings															
<table border="1" style="width: 100%; border-collapse: collapse; margin-top: 20px;"> <thead> <tr> <th style="width: 10%;">SI.No</th> <th style="width: 40%;">Error/Warning Simulated</th> <th style="width: 50%;">Reflection of the Error/Warning on Compilation/Output</th> </tr> </thead> <tbody> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> </tbody> </table>		SI.No	Error/Warning Simulated	Reflection of the Error/Warning on Compilation/Output												
SI.No	Error/Warning Simulated	Reflection of the Error/Warning on Compilation/Output														

3.6.2 Logical Errors

Sl.No	Error/Warning Simulated	Reflection of the Error/Warning on Compilation/Output

4 Final Program and Results

PROGRAM 10**1 Problem Statement**

The birth-date for set of people is given in a database, which includes all different categories of birthday holders (leap year, born in different month etc). Design and develop code in C language. Analyze it from the perspective of boundary value testing, derive different test cases, execute these test cases and discuss the test results.

2 Student Learning Outcomes

After successful completion of this lab, the student shall be able to:

- Generate the leap year.
- Calculate the next day of the current date given as input.
- Know the boundary value analysis.
- Understand how equivalence class is used to generates the boundary.

3 Design of the Program

3.1	Description
<p>The constraints are</p> <p>C1: $1 \leq \text{month} \leq 12$</p> <p>C2: $1 \leq \text{day} \leq 31$</p> <p>C3: $1812 \leq \text{year} \leq 2012$.</p> <p>If any one condition out of C1, C2 or C3 fails, then this function produces an output "value of month not in the range 1...12".</p> <p>Since many combinations of dates can exist, hence we can simply display one message for this function: "Invalid Input Date".</p> <p>A very common and popular problem occurs if the year is a leap year. We have taken into consideration that there are 31 days in a month. But what happens if a month has 30 days or even 29 or 28 days?</p> <p>A year is called as a leap year if it is divisible by 4, unless it is a century year.</p> <p>Century years are leap years only if they are multiples of 400. So, 1992, 1996 and 2000 are leap years while 1900 is not a leap year.</p>	
3.2	Algorithm
<p>STEP 1: Input date in format DD.MM.YYYY</p> <p>STEP 2: if MM is 01, 03, 05, 07, 08, 10 do STEP 3 else STEP 6</p> <p>STEP 3: if DD < 31 then do STEP 4 else if DD=31 do STEP 5 else output(Invalid Date);</p> <p>STEP 4: tomorrowday=DD+1 goto STEP 18</p> <p>STEP 5: tomorrowday=1; tomorrowmonth=month + 1 goto STEP 18</p> <p>STEP 6: if MM is 04, 06, 09, 11 do STEP 7</p> <p>STEP 7: if DD<30 then do STEP 4 else if DD=30 do STEP 5 else output(Invalid Date);</p> <p>STEP 8: if MM is 12</p> <p>STEP 9: if DD<31 then STEP 4 else STEP 10</p> <p>STEP 10: tomorrowday=1, tomorrowmonth=1, tomorrowyear=YYYY+1; goto STEP 18</p> <p>STEP 11: if MM is 2</p> <p>STEP 12: if DD<28 do STEP 4 else do STEP 13</p> <p>STEP 13: if DD=28 & YYYY is a leap do STEP 14 else STEP 15</p> <p>STEP 14: tomorrowday=29 goto STEP 18</p> <p>STEP 15: tomorrowday=1, tomorrowmonth=3, goto STEP 18;</p> <p>STEP 16: if DD=29 then do STEP 15 else STEP 17</p> <p>STEP 17: output("Cannot have feb", DD); STEP 19</p> <p>STEP 18: output(tomorrowday, tomorrowmonth, tomorrowyear);</p> <p>STEP 19: exit</p> <p>Algorithm Description:</p> <p>Step 1 takes the input in the format</p> <p>Step 2 checks the specific months</p> <p>Step 3 if month is 01, 03, 05, 07, 08, 10 then checks the date</p> <p>Step 4 if date is lesser than 31 then increment the date by 1.</p> <p>Step 5 if months belongs to 04, 06, 09, 11, then checks for date again</p> <p>Step 6 if date is less than 30 then calculate tomorrow's date.</p>	

Step 8 & 11 is used for checking the months.
Step 12, 13, 14, 15, 16, 17, & 18 is used to check the leap years.
Step 19 is used to exit the process.

3.3 Coding Using C

```
1. #include<stdio.h>
2. main( )
3. {
4.  int month[12]={31,28,31,30,31,30,31,31,30,31,30,31};
5.  int d,m,y,nd,nm,ny,ndays;
6.  printf("enter the date,month,year");
7.  scanf("%d%d%d",&d,&m,&y);
8.  ndays=month[m-1];
9.  if(y<=1812 && y>2012)
10. {
11.  printf("Invalid Input Year");
12.  exit(0);
13. }
14. if(d<=0 || d>ndays)
15. {
16.  printf("Invalid Input Day");
17.  exit(0);
18. }
19. if(m<1 && m>12)
20. {
21.  printf("Invalid Input Month");
22.  exit(0);
23. }
24. if(m==2)
25. {
26.  if(y%100==0)
27.  {
28.  if(y%400==0)
29.  ndays=29;
30.  }
31.  else
32.  if(y%4==0)
33.  ndays=29;
34.  }
35.  nd=d+1;
36.  nm=m;
37.  ny=y;
38.  if(nd>ndays)
39.  {
40.  nd=1;
41.  nm++;
```

```
42. }
43. if(nm>12)
44. {
45. nm=1;
46. ny++;
47. }
48. printf("\n Given date is %d:%d:%d",d,m,y);
49. printf("\n Next day's date is %d:%d:%d",nd,nm,ny);
50. }
```

3.4 Expected Results

Enter the date, month, year
00
03
2000
Invalid Input day
Enter the date, month, year
30
03
2000
Given date is: 30:3:2000
The next date is:31:3:2000
Enter the date, month, year
15
11
2000
Given date is: 15:11:2000
The next date is:16:11:200
Enter the date, month, year
31
12
1998
Given date is: 31:12:1998
The next date is:1:1:1999

3.5 Implementation Phase: Compile the Program and note the syntax errors/warning if occurred during each compilation. Remove those syntax errors/warnings for generation of executable file.

3.6	Simulate the Errors/warnings		
3.6.1	Syntax Errors/warnings		
Sl no	Name of Error	Cause for the Error	Rectification
1.	Declaration error	Array variable not declared	Before usage of array elements it needs to be declared.
2.	Missing semicolon	Statement is not exited properly	Each statement needs to be exited properly.
3.6.2	Logical Errors		
Sl no	Name of Error	Cause for the Error	Rectification
1.	Postincrement usage	To change the value of month, year, date , post increment is not used appropriately	Proper usage of post-increment.
2.	Missing Array elements	Out of 12 months in a year, only 11 months are declared.	All the months needs to specify.
4	Final Program and Results		

--	--

VIVA QUESTIONS

1. What is the MAIN benefit of designing tests early in the life cycle?
2. When is used Decision table testing?
3. What is beta testing?
4. What is the difference between Testing Techniques and Testing Tools?
5. What is component testing?
6. What is an equivalence partition (also known as an equivalence class)?
7. Testing activity which is performed to expose defects in the interfaces and in the interaction between integrated components is?
8. What are the Structure-based (white-box) testing techniques?
9. When "Regression Testing" should be performed?
10. What is negative and positive testing?
11. What is the purpose of a test completion criterion?
12. When should testing be stopped?
13. What are semi-random test cases?
14. Why does the boundary value analysis provide good test cases?
15. What is maintenance testing?
16. Why is incremental integration preferred over "big bang" integration?
17. Why we split testing into distinct stages?
18. What studies data flow analysis?
19. What is Alpha testing?
20. What is a failure?