VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI

Software Testing Laboratory

(18ISL66)

(As per Visvesvaraya Technological University Syllabus)

Compiled By:

Prof. Pushpalatha K S
Assistant Professor, Dept. of ISE

Prof. Supriya C
Assistant Professor, Dept. of ISE

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
ACHARYA INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi and Govt. of Karnataka),
Acharya Dr. Sarvepalli Radhakrishnan Road, Bangalore-560107.
Ph : 91-080-28396011, 23723466, 28376431
URL: www.acharya.ac.in

2022-23

Disclaimer

Trademark

Edition: 2022 - 23

Document Owner

The primary contact for questions regarding this document is:

Author(s): 1. Prof. Pushpalatha K.S
2. Prof. Supriya C

Department: Information Science & Engineering

Contact Email(s): pushpalatha2391@acharya.ac.in
supriyac2511@acharya.ac.in

Part B

1. 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, execute the test cases and discuss the results.

2. 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.

3. Design, develop, code and run the program in any suitable language to implement the NextDate function. Analyze it from the perspective of boundary value 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 equivalence class partitioning, 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 equivalence class 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 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.

7. 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. Derive test cases for your program based on decision-table approach, execute the test cases and discuss the results.

8. 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

9. 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.

10. 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.

11. Design, develop, code and run the program in any suitable language to implement the quick sort algorithm. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.

12. Design, develop, code and run the program in any suitable language to implement an absolute letter grading procedure, making suitable assumptions. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.

Program 7: Decision Table Approach for Solving Triangle Problem

/* 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 triangle and determine if the three values represent an equilateral triangle, isosceles triangle, scalene triangle, or they do not form a triangle at all. Derive test cases for your program based on decision-table approach, execute the test cases and discuss the results */

```
#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;
}
```

Test Case Name :Decision table for triangle problem

Experiment Number : 7

Test Data : Enter the 3 Integer Value(a , b And c)

Pre-condition : a < b + c , b < a + c and c < a + b

Brief Description : Check whether given value for a equilateral, isosceles , Scalene triangle or can't form a triangle

Input data decision Table

RULES		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
Conditions	C1: a < b + c	F	T	T	T	T	T	T	T	T	T	T
	C2 : b < a + c	-	F	T	T	T	T	T	T	T	T	T
	C3 : c < a + b	-	-	F	T	T	T	T	T	T	T	T
	C4 : a = b	-	-	-	T	T	T	T	F	F	F	F
	C5 : a = c	-	-	-	T	T	F	F	T	T	F	F
	C6 : b = c	-	-	-	T	F	T	F	T	F	T	F
Actions	a1 : Not a triangle	X	X	X								
	a2 : Scalene triangle											X
	a3 : Isosceles triangle							X		X	X	
	a4 : Equilateral triangle				X							
	a5 : Impossible					X	X		X			

Case Id	Description	Input Data			Expected Output	Actual Output	Status	Comments
		a	b	C				
1	Enter the value of a, b and c Such that a is not less than sum of two sides	20	5	5	Message should be displayed can't form a triangle	Cant form a triangle	PASS	R1 (a < b+c condition fails)
2	Enter the value of a, b and c Such that b is not less than sum of two sides and a is less than sum of other two sides	3	15	11	Message should be displayed can't form a triangle			R2
3	Enter the value of a, b and c Such that c is not less than sum of two sides and a and b is less than sum of other two sides	4	5	20	Message should be displayed can't form a triangle			R3
4	Enter the value a, b and c satisfying precondition and a=b, b=c and c=a	5	5	5	Should display the message Equilateral triangle			R4(a=b=c)
5	Enter the value a ,b and c satisfying precondition and a=b and b ≠ c	10	10	9	Should display the message Isosceles triangle			R7(a=b)
6	Enter the value a, b and c satisfying precondition and a ≠b , b ≠ c and c ≠ a	5	6	7	Should display the message Scalene triangle			R11(a!=b!=c)

Program 1 and 4 (Boundary Value and Equivalence Class Analysis Program)

/* 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 triangle and determine if the three values represent an equilateral triangle, isosceles triangle, scalene triangle, or they do not form a triangle at all. Derive test cases for your program based on boundary value analysis, execute the test cases and discuss the results */

```
#include<stdio.h>
int main()
{
        int a,b,c,c1,c2,c3;
        char  istriangle;
        do
        {
                printf("\nenter 3 integers which are sides of triangle\n");
                scanf("%d%d%d",&a,&b,&c);
                printf("\na=%d\tb=%d\tc=%d",a,b,c);
                c1 = a>=1 && a<=10;
                c2= b>=1 && b<=10;
                c3= c>=1 && c<=10;
                if (!c1)
                        printf("\nthe value of a=%d is not the range of permitted value",a);
                 if (!c2)
                        printf("\nthe value of b=%d is not the range of permitted value",b);
                if (!c3)
                        printf("\nthe value of c=%d is not the range of permitted value",c);
        } while(!(c1 && c2 && c3));

        // 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;
}
```

Test Case Name :Boundary Value Analysis for triangle problem

Experiment Number : 1

Test Data : Enter the 3 Integer Value(a , b And c)

Pre-condition : $1 \leq a \leq 10$, $1 \leq b \leq 10$ and $1 \leq c \leq 10$ and $a < b + c$, $b < a + c$ and $c < a + b$

Brief Description : Check whether given value for a Equilateral, Isosceles , Scalene triangle or can't form a triangle

Triangle Problem -Boundary value Test cases for input data

Case Id	Description	Input Data			Expected Output	Actual Output	Status	Comments
		a	B	c				
1	Enter the min value for a , b and c	1	1	1	Should display the message Equilateral triangle	Equilateral	PASS	a=b=c
2	Enter the min value for 2 items and min +1 for any one item1	1	1	2	Message should be displayed can't form a triangle			
3	Enter the min value for 2 items and min +1 for any one item1	1	2	1	Message should be displayed can't form a triangle			
4	Enter the min value for 2 items and min +1 for any one item1	2	1	1	Message should be displayed can't form a triangle			
5	Enter the normal value for 2 items and 1 item is min value	5	5	1	Should display the message Isosceles triangle			
6	Enter the normal value for 2 items and 1 item is min value	5	1	5	Should display the message Isosceles triangle			
7	Enter the normal value for 2 items and 1 item is min value	1	5	5	Should display the message Isosceles triangle			
8	Enter the normal Value for a, b and c	5	5	5	Should display the message Equilateral triangle			

9	Enter the normal value for 2 items and 1 item is max value	5	5	10	Should display the message Not a triangle			
10	Enter the normal value for 2 items and 1 item is max value	5	10	5	Should display the message Not a triangle			
11	Enter the normal value for 2 items and 1 item is max value	10	5	5	Should display the message Not a triangle			
12	Enter the max value for 2 items and max - 1 for any one item	10	10	9	Should display the message Isosceles triangle			
13	Enter the max value for 2 items and max - 1 for any one item	10	9	10	Should display the message Isosceles triangle			
14	Enter the max value for 2 items and max - 1 for any one item	9	10	10	Should display the message Isosceles triangle			
15	Enter the max value for a, b and c	10	10	10	Should display the message Equilateral triangle			

Test Case Name :Equivalence class Analysis for triangle problem
Experiment Number : 4
Test Data : Enter the 3 Integer Value(a , b And c)
Classes: { (1,3),(3,5),(5,7)(7,10) }
Pre-condition : $1 \leq a \leq 10$, $1 \leq b \leq 10$ and $1 \leq c \leq 10$ and $a < b + c$, $b < a + c$ and $c < a + b$
Brief Description : Check whether given value for a Equilateral, Isosceles , Scalene triangle or can't form a triangle

Triangle Problem - Equivalence Class Test cases for input data

Weak Equivalence class Testing								
Case Id	Description	Input Data			Expected Output	Actual Output	Status	Comments
		a	b	C				
1	From class (1,3)	1	1	1	Should display the message Equilateral triangle			
2	From class(3,5)	3	5	3	Should display the message Isosceles triangle			
3	From class(3,5)	3	4	5	Should display the message Scalene triangle			
4	From class(3,5)	4	3	5	Message should be displayed can't form a triangle			

Weak Robust Equivalence Class Testing								
5	Enter one invalid input and two valid value for a , b and c	-1	5	5	Should display value of a is not in the range of permitted values			
6	Enter one invalid input and two valid value for a , b and c	5	-1	5	Should display value of a is not in the range of permitted values			
7	Enter one invalid input and two valid value for a , b and c	5	5	-1	Should display value of a is not in the range of permitted values			
8	Enter one invalid input and two valid value for a , b and c	11	5	5	Should display value of a is not in the range of permitted values			

9	Enter one invalid input and two valid value for a , b and c	5	11	5	Should display value of a is not in the range of permitted values			
10	Enter one invalid input and two valid value for a , b and c	5	5	11	Should display value of a is not in the range of permitted values			
Strong Robust Equivalence class Testing								
11	Enter one invalid input and two valid value for a , b and c	-1	5	5	Should display value of a is not in the range of permitted values			
12	Enter one invalid input and two valid value for a , b and c	5	-1	5	Should display value of a is not in the range of permitted values			
13	Enter one invalid input and two valid value for a , b and c	5	5	-1	Should display value of a is not in the range of permitted values			
14	Enter two invalid input and two valid value for a , b and c	-1	-1	5	Should display value of a is not in the range of permitted values			
					Should display value of b is not in the range of permitted values			
14	Enter two invalid input and two valid value for a , b and c	5	-1	-1	Should display value of b is not in the range of permitted values			
					Should display value of c is not in the range of permitted values			
14	Enter two invalid input and two valid value for a , b and c	-1	5	-1	Should display value of a is not in the range of permitted values			
					Should display value of c is not in the range of permitted values			
15	Enter all invalid inputs	-1	-1	-1	Should display value of a is not in the range of permitted values			
					Should display value of b is not in the range of permitted values			
					Should display value of c is not in the range of permitted values			

```
1  //Program 9: (Dataflow Testing for commission calculation)
2  #include<stdio.h>
3  int main()
4  {
5          int locks, stocks, barrels, tlocks, tstocks, tbarrels;
6          float lprice,sprice,bprice,lsales,ssales,bsales,sales,comm;
7          lprice=45.0;
8          sprice=30.0;
9          bprice=25.0;
10         tlocks=0;
11         tstocks=0;
12         tbarrels=0;
13         printf("\nenter the number of locks and to exit the loop enter -1 for locks\n");
           scanf("%d", &locks);
14         while(locks!=-1) {
15                 printf("enter the number of stocks and barrels\n");
                   scanf("%d%d",&stocks,&barrels);
16                 tlocks=tlocks+locks;
17                 tstocks=tstocks+stocks;
18                 tbarrels=btarrels+barrels;
19                 printf("\nenter the number of locks and to exit the loop enter -1 for
                   locks\n"); scanf("%d",&locks);
20         }
21         printf("\ntotal locks = %d\",tlocks);
22         printf("total stocks =%d\n",tstocks);
23         printf("total barrels =%d\n",tbarrels);

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);
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);
             }
34         else if(sales > 1000)
35         {
36                 comm =0.10*1000;
37                 comm=comm+0.15*(sales-1000);
           }
38         else
39                 comm=0.10*sales;
40         printf("the commission is=%f\n",comm);
41         return 0;
42 }
```

Test Case Name : Data Flow Testing for Commission Program

Experiment No : 9

Precondition : Enter -1 for locks to exit from input loop

Brief Description : Enter the locks, stocks and barrels > 0

Define /Use nodes for variables in the commission problem

Variable name	Defined at node	Used at Node
Lprice	7	24
Sprice	8	25
Bprice	9	26
Tlocks	10,16	16,21,24
Tstocks	11,17	17,22,25
Tbarrels	12,18	18,23,26
Locks	13,19	14,16
Stocks	15	17
Barrels	15	18
Lsales	24	27
Ssales	25	27
Bsales	26	27
Sales	27	28,29,33,34,37,39
Comm	31,32,33,36,37,39	32,33,37,40

+

Selected De fine/Use Paths for Com mission problem

Test case id	Description	Variables Path (Beginning, End nodes)	Du Paths	Definition clear?	Comments
1	Check for lock price variable DEF(lprice,7) and USE(lprice,24)	(7 , 24)	<7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24>	Yes	
2	Check for Stock price variable DEF(sprice,8) and USE(sprice,25)	(8 , 25)	<8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25>	Yes	
3	Check for barrel price variable DEF(bprice,9) and USE(bprice,26)	(9 , 26)	<9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26>	Yes	
4	Check for total locks variable DEF((tlocks,10)and DEF(tlocks,16)) and 3 usage node(USE(tlocks,16),USE(tlocks,21), USE(tlocks,24))	(10 , 16)	<10-11-12-13-14-15-16>	Yes	
		(10 , 21)	<10-11-12-13-14-15-16-17-18-19-20-14-21>	No	
		(10 , 24)	<10-11-12-13-14-15-16-17-18-19-20-14-21-22-23-24>	No	
		(16 , 16)	<16-16>	Yes	
		(16 , 21)	<16-17-18-19-14-21>	No	
		(16 , 24)	<16-17-18-19-20-14-21-22-23-24>	No	
5	Check for total stocks variable DEF((tstocks,11) and DEF(tstocks,17)) and 3 usage node(USE(tstocks,17),USE(tstocks,22), USE(tstocks,25))	(11 , 17)	<11-12-13-14-15-16-17>	Yes	
		(11 , 22)	<11-12-13-14-15-16-17-18-19-20-21-14-21>	No	
		(11, 25)	<11-12-13-14-15-16-17-18-19-20-21-14-21-23-24-25>	No	
		(17 , 17)	<17-17>	Yes	
		(17 , 22)	<17-18-19-20-14-21-22>	No	

		(17 , 25)	<17-18-19-20-14-21-22-23-24-25>	No	
6	check for locks variable (DEF(locks,13), DEF(locks,19) and USE(locks,14),USE(locks,16)	(13 , 14)	<13-14>	Yes	Begin the loop
		(13 , 16)	<13-14-15-16>	Yes	
		(19 , 14)	<19-20-14>	Yes	
		(19 , 16)	<19-20-14-15-16>	Yes	Repeat the loop
7	Check for stocks variable (DEF(stocks,15) and USE(stocks,17)	(15 , 17)	<15-16-17>	Yes	
8	Check for sales DEF(sales, 27) and USE(Sales, 28), USE(Sales , 29), USE(Sales,33) , USE(Sales , 34) , USE(Sales,37) , USE(Sales , 39)	(27 ,28)	<27-28>	Yes	
		(27 , 29)	<27-28-29>	Yes	
		(27 , 33)	<27-28-29-30-31-32-33>	Yes	
		(27 , 34)	<27-28-29-34>	Yes	
		(27 , 37)	<27-28-29-34-35-36-37>	Yes	
		(27 , 39)	<27-28-29-34-38-39>	Yes	
9	Check for Commission variable DEF(comm, 31,32,33) , DEF(comm,34,35) and DEF(comm,39) and USE(comm,40)	((31,32,33),40)	<31-32-33-40>	Yes	
		((34 , 35) , 40)	<34-35-40>	Yes	
		((39 , 40)	<39 - 40>	Yes	

Program 2,5 and8 (Boundary, Equivalence and Decision Test Case for Commission Problem)

/* Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of boundary value, derive 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;
}
```

Test Case Name : Boundary Value for Commission Problem

Experiment Number : 2

Test data : price Rs for lock - 45.0 , stock - 30.0 and barrel - 25.0

sales = total lock * lock price + total stock * stock price + total barrel * barrel price

commission : 10% up to sales Rs 1000 , 15 % of the next Rs 800 and 20 % on any sales in excess of 1800

Pre-condition : lock = -1 to exit and 1< =lock < = 70 , 1<=stock <=80 and 1<=barrel<=90

Brief Description : The salesperson had to sell at least one complete rifle per month.

CHECKING BOUNDARY VALUE FOR LOCKS, STOCKS AND BARRELS AND COMMISSION

Commission Problem Output Boundary Value Analysis Cases

Case Id	Description	Input Data			Expected Output		Actual output		Status	
		Total Locks	Total Stocks	Total Barrels	Sales	Comm-ission	Sales	Comm -ission		Comment
1	Enter the min value for locks, stocks and barrels	1	1	1	100	10				output minimum
2	Enter the min value for 2 items and min +1 for any one item	1	1	2	125	12.5				output minimum +
3		1	2	1	130	13				output minimum +
4		2	1	1	145	14.5				output minimum +
5	Enter the value sales approximately mid value between 100 to 1000	5	5	5	500	50				Midpoint
6	Enter the values to calculate the commission for sales nearly less than 1000	10	10	9	975	97.5				Border point -
7		10	9	10	970	97				Border point -
8		9	10	10	955	95.5				Border point -
9	Enter the values sales exactly equal to 1000	10	10	10	1000	100				Border point
10	Enter the values to calculate the commission for sales nearly greater than 1000	10	10	11	1025	103.75				Border point +
11		10	11	10	1030	104.5				Border point +
12		11	10	10	1045	106.75				Border point +
13	Enter the value sales approximately mid value between 1000 to 1800	14	14	14	1400	160				Midpoint

14	Enter the values to calculate the commission for sales nearly less than 1800	18	18	17	1775	216.25				Border point -
15		18	17	18	1770	215.5				Border point -
16		17	18	18	1755	213.25				Border point -
17	Enter the values sales exactly equal to 1800	18	18	18	1800	220				Border point
18	Enter the values to calculate the commission for sales nearly greater than 1800	18	18	19	1825	225				Border point +
19		18	19	18	1830	226				Border point +
20		19	18	18	1845	229				Border point +
21	Enter the values normal value for lock, stock and barrel	48	48	48	4800	820				Midpoint
22	Enter the max value for 2 items and max - 1 for any one item	70	80	89	7775	1415				Output maximum -
23		70	79	90	7770	1414				Output maximum -
24		69	80	90	7755	1411				Output maximum -
25	Enter the max value for locks, stocks and barrels	70	80	90	7800	1420				Output maximum

Output Special Value Test Cases

Case Id	Description	Input Data			Expected Output		Actual output			
		Total Locks	Total Stocks	Total Barrels	Sales	Commission	Sales	Commission	Status	Comment
1	Enter the random values such that to calculate commission for sales nearly less than 1000	11	10	8	995	99.5				Border point -
2	Enter the random values such that to calculate commission for sales nearly greater than 1000	10	11	9	1005	100.75				Border point +
3	Enter the random values such that to calculate commission for sales nearly less than 1800	18	17	19	1795	219.25				Border point -
4	Enter the random values such that to calculate commission for sales nearly greater than 1800	18	19	17	1805	221				Border point +

Test Case Name :Equivalence Class for Commission Problem

Experiment Number : 5

Test data : price Rs for lock - 45.0 , stock - 30.0 and barrel - 25.0

sales = total lock * lock price + total stock * stock price + total barrel * barrel price

commission : 10% up to sales Rs 1000 , 15 % of the next Rs 800 and 20 % on any sales in excess of 1800

Pre-condition : lock = -1 to exit and 1< =lock < = 70 , 1<=stock <=80 and 1<=barrel<=90

Brief Description : The salesperson had to sell at least one complete rifle per month.

Checking boundary value for locks, stocks and barrels and commission

Valid Classes

L1 ={LOCKS :1 <=LOCKS<=70}

L2 ={Locks=-1}(occurs if locks=-1 is used to control input iteration)

L3 ={stocks : 1<=stocks<=80}

L4= {barrels :1<=barrels<=90}

Invalid Classes

L3 ={locks: locks=0 OR locks<-1}

L4 ={locks: locks> 70}

S2 ={stocks : stocks<1}

S3 ={stocks : stocks >80}

B2 ={barrels : barrels <1}

B3 =barrels : barrels >90}

Commission Problem Output Equivalence Class Testing

(Weak & Strong Normal Equivalence Class)

Case Id	Description	Input Data			Expected Output		Actual output		Status	Comment
		Total Locks	Total Stocks	Total Barrels	Sales	Commission	Sales	Commission		
1	Enter the value within the range for lock, stocks and barrels	35	40	45	3900	640				

Weak Robustness Equivalence Class

Case	Description	Input Data	Expected Output	Actual output	Status	Comment

Id		Locks	Stocks	Barrels				
WR1	Enter the value locks = -1	-1	40	45	Terminates the input loop and proceed to calculate sales and commission (if Sales > 0)			
WR2	Enter the value less than -1 or equal to **zero** for locks and other valid inputs	0	40	45	Value of Locks not in the range 1..70			
WR3	Enter the value greater than 70 for locks and other valid inputs	71	40	45	Value of Locks not in the range 1..70			
WR4	Enter the value less than or equal than 0 for stocks and other valid inputs	35	0	45	Value of stocks not in the range 1..80			
WR5	Enter the value greater than 80 for stocks and other valid inputs	35	81	45	Value of stocks not in the range 1..80			
WR6	Enter the value less than or equal 0 for barrels and other valid inputs	35	40	0	Value of Barrels not in the range 1..90			
WR7	Enter the value greater than 90 for barrels and other valid inputs	35	40	91	Value of Barrels not in the range 1..90			

Strong Robustness Equivalence Class

Case Id	Description	Input Data			Expected Output	Actual output	Status	Comment
		Locks	Stocks	Barrels				
SR1	Enter the value less than -1 for locks and other valid inputs	-2	40	45	Value of Locks not in the range 1..70			
SR2	Enter the value less than or equal than 0 for stocks and other valid inputs	35	-1	45	Value of stocks not in the range 1..80			
SR3	Enter the value less than or equal 0 for barrels and other valid inputs	35	40	-2	Value of Barrels not in the range 1..90			
SR4	Enter the locks and stocks less than or equal to 0 and other valid inputs	-2	-1	45	Value of Locks not in the range 1..70			
					Value of stocks not in the range 1..80			
SR5	Enter the locks and barrel less than or equal to 0 and other valid inputs	-2	40	-1	Value of Locks not in the range 1..70			
					Value of Barrels not in the range 1..90			
SR6	Enter the stocks and barrel less than or equal to 0 and other valid inputs	35	-1	-1	Value of stocks not in the range 1..80			
					Value of Barrels not in the range 1..90			

SR7	Enter the stocks and barrel less than or equal to 0 and other valid inputs	-2	-2	-2	Value of Locks not in the range 1..70			
					Value of stocks not in the range 1..80			
					Value of Barrels not in the range 1..90			

Some addition equivalence Boundary checking

Case Id	Description	Input Data			Expected Output		Actual output			
		Total Locks	Total Stocks	Total Barrels	Sales	Commission	Sales	Commission	Status	Comment
OR1	Enter the value for lock, stocks and barrels where 0 < Sales < 1000	5	5	5	500	50				
OR2	Enter the value for lock, stocks and barrels where 1000 < Sales < 1800	15	15	15	1500	175				
OR3	Enter the value for lock, stocks and barrels where Sales < 1800	25	25	25	2500	360				

Test Case Name :Decision Table for Commission Problem

Experiment Number : 8

Test data : price Rs for lock - 45.0 , stock - 30.0 and barrel - 25.0

sales = total lock * lock price + total stock * stock price + total barrel * barrel price

commission : 10% up to sales Rs 1000 , 15 % of the next Rs 800 and 20 % on any sales in excess of 1800

Pre-condition : lock = -1 to exit and 1< =lock < = 70 , 1<=stock <=80 and 1<=barrel<=90

Brief Description : The salesperson had to sell at least one complete rifle per month.

Input data decision Table

RULES		R1	R2	R 3	R4	R5	R 6	R7	R8	R10
Conditions	C1: Locks = -1	T	F	F	F	F	F	F	F	F
	C2 : 1 ≤ Locks ≤ 70	-	T	T	F	T	F	F	F	T
	C3 : 1 ≤ Stocks ≤ 80	-	T	F	T	F	T	F	F	T
	C4 : 1 ≤ Barrels ≤ 90	-	F	T	T	F	F	T	F	T
Actions	a1 : Terminate the input loop	X								
	a2 : Invalid locks input				X		X	X	X	
	a3 : Invalid stocks input			X		X		X	X	
	a4 : Invalid barrels input		X			X	X		X	
	a5 : Calculate total locks, stocks and barrels		X	X	X	X	X	X		X
	a5 : Calculate Sales	X								
	a6: proceed to commission decision table	X								

Commission calculation Decision Table (Precondition : lock = -1)

RULES		R1	R2	R3	R4
Condition	C1 : tlocks>0 && tstocks>0 && tbarrels>0	T	T	T	F
	C1 : Sales > 0 AND Sales ≤ 1000	T	F	F	
	C2 : Sales > 1001 AND sales ≤ 1800		T	F	
	C3 : sales ≥1801			T	
Actions	A1 : Cannot calculate the commission				X
	A2 : comm= 10%*sales	X			
	A3 : comm = 10%*1000 + (sales-1000)*15%		X		
	A4 : comm = 10%*1000 + 15% * 800 + (sales-1800)*20%			X	

Precondition : Initial Value Total Locks= 0 , Total Stocks=0 and Total Barrels=0

Precondition Limit :Total locks, stocks and barrels should not exceed the limit 70,80 and 90 respectively

NIR-Not in Range
SAEO-Same as Expected Output

Commission Problem -Decision Table Test cases for input data

Case Id	Description	Input Data			Output					Actual Output	Status	Comments
		Locks	Stocks	Barrels	Terminate i/p loop	Calculate commission	Total locks	Total stocks	Total Barrels			
1	Enter the value of Locks= -1	-1			yes	0				SAEO	PASS	
2	Enter the valid input for lock and stack and invalid for barrels	20	30	-5		No	20	30	NIR	SAEO	PASS	
3	Enter the valid input for lock and barrels and invalid for stocks	15	-2	45								
4	Enter the valid input for lock and barrels and invalid for stocks	-4	15	16								
5	Enter the valid input for lock and invalid value for stocks and barrels	15	80	100								
6	Enter the valid input for stocks and invalid value for locks and barrels	88	20	99								
7	Enter the valid input for barrels and invalid value for locks and stocks	100	200	25								

Commission Problem -Decision Table Test cases for commission calculation

L-locks
S-Stocks
B-Barrels
Precondition : Locks = -1

Case Id	Description	Input Data			Expected Output				Actual Output	Status	Comments
		L	S	B	Expected Sales	Actual Sales	Expected Commission	Actual Commission			
1	Check the value of sales=0	0	0	0	0	0	0	0	SAEO	PASS	
2	if sales value with in these range(Sales > 0 AND Sales ≤ 1000)	10	9	10							
3	if sales value with in these range(Sales > 1000 AND Sales ≤ 1800)	15	15	15							
4	if sales value with in these range(Sales > 1800	20	30	40							

Program 10 (Binary Search - Path Testing)

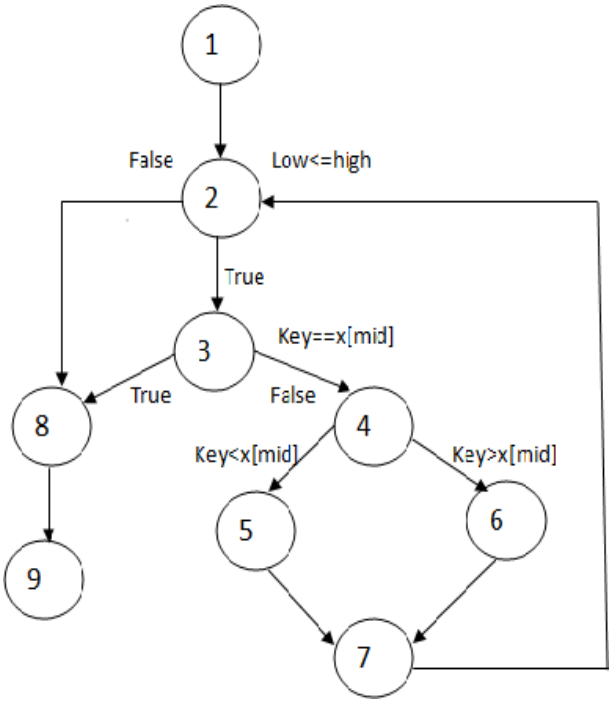
/* Design, develop a code and run the program in any suitable language to implement the binary search algorithm. Determine the basis paths and using them derive different test cases execute these test cases and discuss the test results */

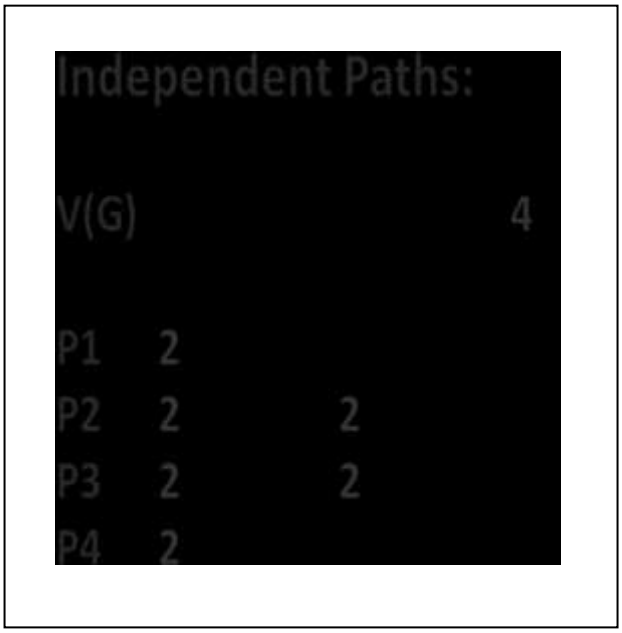
```
#include<stdio.h>
int binsrc(int x[],int low,int high,int key)
{
        int mid;
        while(low<=high)
        {
                mid=(low+high)/2;
                if(key==x[mid])
                        return mid;
                if(key<x[mid])
                        high=mid-1;
                else
                        low=mid+1;
        }
        return -1;
}

int main()
{
        int x[20],key,i,n,succ;
        printf("Enter the n value");
        scanf("%d",&n);
        if(n>0)
        {
        printf("enter the elements in ascending order\n");
                for(i=0;i<n;i++)
                scanf("%d",&x[i]);

        printf("enter the key element to be searched\n");
              scanf("%d",&key);
                succ=binsrc(x,0,n-1,key);
                if(succ>=0)
                        printf("Element found in position = %d\n",succ+1);
                else
                        printf("Element not found \n");
        }
        else
                printf("Number of element should be greater than zero\n");
        return 0;
}
```

Binary Search function with line number

```
int binsrc(int x[],int low,int high,int key)                    
{                                                               
        int mid;                                                1
        while(low<=high)                                        2
        {                                                       
                mid=(low+high)/2;                               
                if(key==x[mid])                                 3
                        return mid;                             8
                if(key<x[mid])                                  4
                        high=mid-1;                             5
                Else                                            
                        low=mid+1;                              6
        }                                                       7
        return -1;                                              8
}                                                               9
```

Program Graph – for Binary Search





Test Cases – Binary Search

Paths	Inputs		Expected Output	Remarks
	X[]	Key		
P1: 1-2-3-8-9	{10,20,30,40,50}	30	Success	Key ∈ X[] and Key==X[mid]
P2: 1-2-3-4-5-7-2	{10,20,30,40,50}	20	Repeat and Success	Key < X[mid] Search 1st Half
P3: 1-2-3-4-6-7-2	{10,20,30,40,50}	40	Repeat and Success	Key > X[mid] Search 2nd Half
P4: 1-2-8-9	{10,20,30,40,50}	60 OR 05	Repeat and Failure	Key ∉ X[]
P4: 1-2-8-9	Empty	Any Key	Failure	Empty List

Program 11(Quick Sort-Path Testing)

/*Design, develop, code and run the program in any suitable language to implement the Quick-Sort Algorithm. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.*/

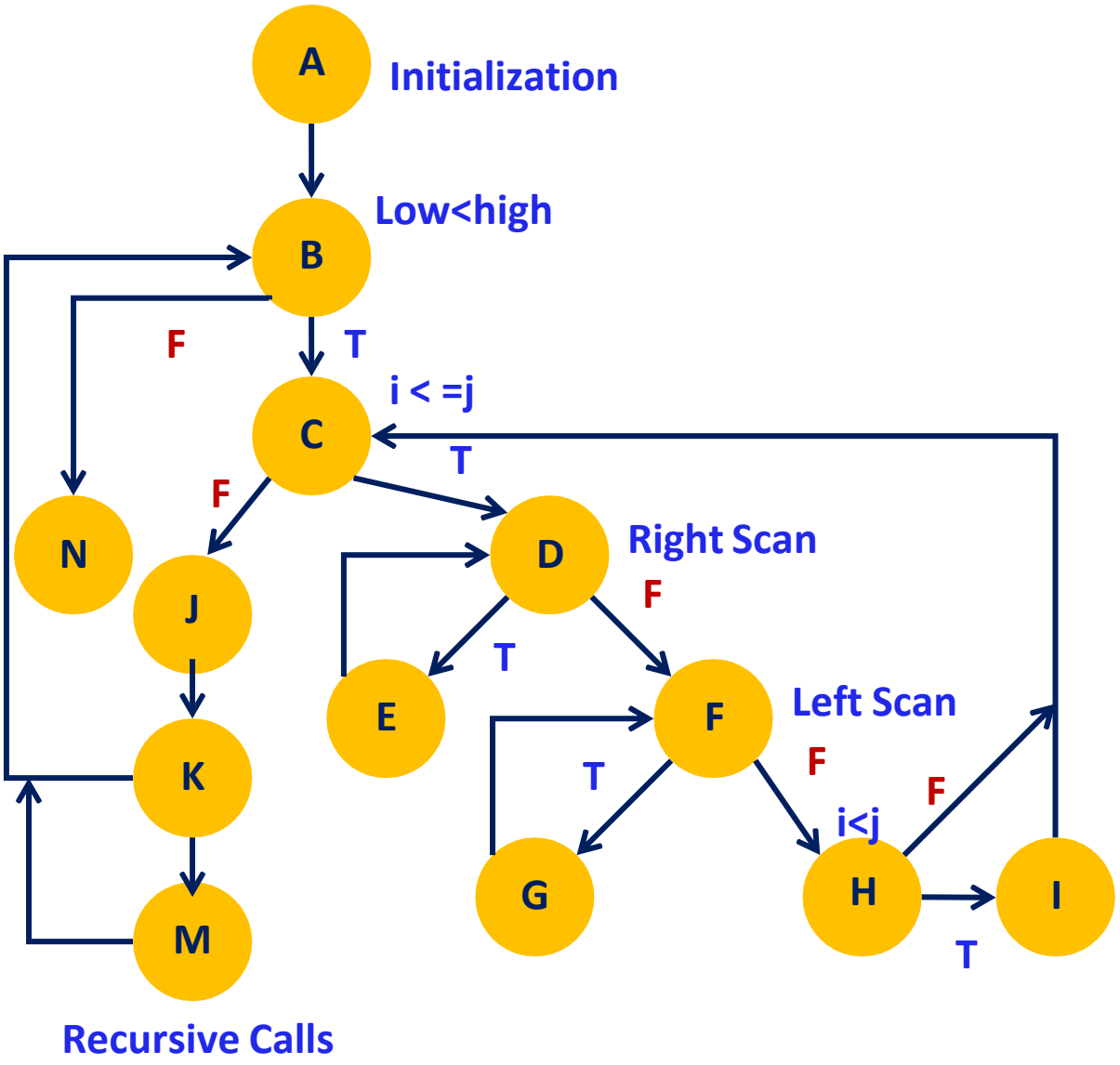
```
#include<stdio.h>
void quick_sort(int a[100],int low,int high)
{
   int i,j,temp,key;
   if(low<high)
   {
         key=a[low];
         i=low;
         j=high;


   //Partition the array into two parts
   while(i<=j)
   {
         while(key>=a[i])
         i++;
         while(key<a[j])
         j--;
         if(i<j)
         {
                  temp=a[i];
                  a[i]=a[j];
                  a[j]=temp;
         }
   }
   //partition completed

  temp=a[low];
  a[low]=a[j];
  a[j]=temp;

  //Consider j as mid and cal recursive function.

  quick_sort(a,low,j-1);
  quick_sort(a,j+1,high);

   }
}
//Main Function
void main()
{
   int i,n,a[100];
   printf("Enter the value for n\n");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
```

```
    quick_sort(a,0,n-1);
    printf("The sorted array is\n");
    for(i=0;i<n;i++)
    printf("%d\n",a[i]);
}
```

Quick sort function with line number

```
void quick_sort(int a[100],int low,int high)
{
    int i,j,temp,key;                                   1   A
    if(low<high)                                        2 B
    {
          key=a[low];                                   3 C
          i=low;                                        4 C
          j=high;                                       5 C


    //Partition the array into two parts
    while(i<=j)                                         6 D
    {
          while(key>=a[i])                              7 E
          i++;                                          8 F
          while(key<a[j])                               9G
          j--;                                          10 H
          if(i<j)                                       11  I
          {
                temp=a[i];                              12 J
                a[i]=a[j];                              13 J
                a[j]=temp;                              14 J
          }
    }
    //partition completed

   temp=a[low];                                         15 K
   a[low]=a[j];                                         16 K
   a[j]=temp;                                           17 K

   //Consider j as mid and cal recursive function.

   quick_sort(a,low,j-1);                               18 L
   quick_sort(a,j+1,high);                              19 M

    }                                                   20
}
```

Program Graph – Quick Sort



Independent Paths– Quick Sort

P1: A-B-N
P2: A-B-C-J-K-B
P3: A-B-C-J-K-M-B
P4: A-B-C-D-F-H-C
P5: A-B-C-D-F-H-I-C
P6: A-B-C-D-E-D-F-H
P7: A-B-C-D-F-G-F-H

Independent Paths:

#Edges=18, #Nodes=13, #P=1

V(G)= E-N+2P = 18-13+2 = **7**

Pre-Conditions/Issues:

Array has only one Element, Two Elements, Three Elements (6 Possibilities)
Array has Elements in ASC/DSC/Arbitrary(Any of the Permutations)
EX: 3 elements: 123, 132, 213, 231, 312, 321, 222,111,333

Test Cases – Quick Sort

Paths	Inputs		Expected Output	Remarks
	X[]	First , Last		
P1: A-B-N	5	1,1	Sorted	Only one Elem
P2: A-B-C-J-K-B	5,4	1,2	Repeat & Sorted	Two Elements
P3: A-B-C-J-K-M-B	1,2,3 OR 3,1,2	1,3	Repeat & Sorted	Three Elements
P4: A-B-C-D-F-H-C	1,2,3,4,5	1,5	Repeat & Sorted	ASC Sequence
P5: A-B-C-D-F-H-I-C	5,4,3,2,1	1,5	Repeat & Sorted	DSC Sequence
P6: A-B-C-D-E-D-F-H	1,4,3,2,5 OR 2,2,2,2,2	1,5	Repeat & Sorted	Pivot is MIN
P7: A-B-C-D-F-G-F-H	5,2,3,1,4	1,5	Repeat & Sorted	Pivot is MAX

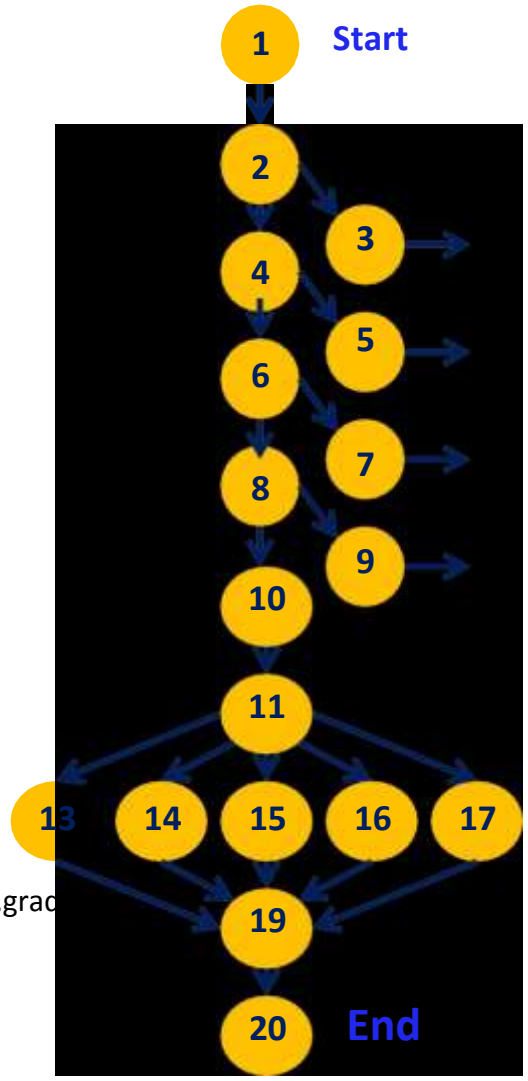
Program 12 (Absolute Letter Grading Path Testing)

/* Design, develop, code and run the program in any suitable language to implement an absolute letter grading procedure, making suitable assumptions. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results */

```
#include<stdio.h>
int main()
{
        float per;
        char grade;
        scanf("%f",&per);
        if(per>=90)
                grade= 'A';
        else if(per>=80 && per<90)
                grade ='B';
        else if(per>=70 && per<80)
                grade ='C';
        else if(per>=60 && per<70)
                grade='D';
        else grade='E';
           switch(grade)
        {
        case 'A': printf("\nEXCELLENT");  break;
        case  'B':printf("\nVery Good"); break;
        case 'C' : printf("\nGood"); break;
        case 'D': printf("\nAbove Average"); break;
        case 'E': printf("\n Satisfactory"); break;
        }
    printf("\t The percentage = %f and grade is %c ",per,grade);
          return 0;
}
```

Absolute Grading Program with Line Numbers and Program Graph

```
int main()
{
        float per;
        char grade;
1.          scanf("%f",&per);
2.          if(per>=90)
3.                  grade= 'A';
4.          else if(per>=80 && per<90)
5.                  grade ='B';
6.          else if(per>=70 && per<80)
7.                  grade ='C';
8.          else if(per>=60 && per<70)
9.                  grade='D';
10.         else grade='E';
11.       switch(grade)
12.         {
13.         case 'A': printf("\nEXCELLENT");  break;
14.         case 'B':printf("\nVery Good"); break;
15.         case 'C' : printf("\nGood"); break;
16.         case 'D': printf("\nAbove Average"); break;
17.         case 'E': printf("\n Satisfactory"); break;
18.         }
19. printf("\t The percentage = %f and grade is %c ",per,grad
20.       return 0;
 }
```



Independent Paths:

#Edges=25, #Nodes=18, #P=1
V(G)= E-N+2P = 25-18+2 = **09**

P1: 1-2-4-6-8-10-11-17-19-20	E Grade
P2: 1-2-4-6-8-9-11-16-19-20	D Grade
P3: 1-2-4-6-7-11-15-19-20	C Grade
P4: 1-2-4-5-11-14-19-20	B Grade
P5: 1-2-3-11-13-19-20	A Grade
P6: 1-2-4-6-8-10-11-13-19-20	
P7: 1-2-4-6-8-10-11-14-19-20	
P8: 1-2-4-6-8-10-11-15-19-20	
P9: 1-2-4-6-8-10-11-16-19-20	

Pre-Conditions/Issues:

Percentage Per is a positive Float Number

Test Cases – Absolute Grading

Paths	Input	Expected Output	Remarks
	Per		
P1: 1-2-4-6-8-10-11-17-19-20	< 60	E Grade, Satisfactory	Pass
P2: 1-2-4-6-8-9-11-16-19-20	60-69	D Grade, Above Average	Pass
P3: 1-2-4-6-7-11-15-19-20	70-79	C Grade, Good	Pass
P4: 1-2-4-5-11-14-19-20	80-89	B Grade, Very Good	Pass
P5: 1-2-3-11-13-19-20	>= 90	A Grade, Excellent	Pass
P6: 1-2-4-6-8-10-11-13-19-20	< 60	Excellent	Fail
P7: 1-2-4-6-8-10-11-14-19-20	< 60	Very Good	Fail
P8: 1-2-4-6-8-10-11-15-19-20	< 60	Good	Fail
P9: 1-2-4-6-8-10-11-16-19-20	< 60	Above Average	Fail

Program 3 and 5 (Next date program)

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

```
#include<stdio.h>
int check(int day,int month)
{
        if((month==4||month==6||month==9 ||month==11) && day==31)
                return 1;
        else
                return 0;
}
int isleap(int year)
{
        if((year%4==0 && year%100!=0) || year%400==0)
                        return 1;
        else
                        return 0;

}
int main()
{
        int day,month,year,tomm_day,tomm_month,tomm_year;
        char flag;
        do
        {
                flag='y';
                printf("\nenter the today's date in the form of dd mm yyyy\n");
                scanf("%d%d%d",&day,&month,&year);
                tomm_month=month;
                tomm_year= year;
                if(day<1 || day>31)
                {
                        printf("value of day, not in the range 1...31\n");
                        flag='n';
                }
                if(month<1 || month>12)
                 {
                        printf("value of month, not in the range 1....12\n");
                        flag='n';
                }
                else if(check(day,month))
                {
                        printf("value of day, not in the range day<=30");
                        flag='n';
                }

                if(year<=1812 || year>2013)
                {
```

```
                printf("value of year, not in the range 1812..... 2013\n");
                flag='n';
        }
        if(month==2)
        {
                if(isleap(year) && day>29)
                 {
                        printf("invalid date input for leap year");
                        flag='n';
                }
                else if(!(isleap(year))&& day>28)
                {
                        printf("invalid date input for not a leap year");
                        flag='n';
                 }
        }
}while(flag=='n');


switch (month)
{
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:if(day<31)
                        tomm_day=day+1;
                else
                   {
                        tomm_day=1;
                        tomm_month=month+1;
                   }
                break;
        case 4:
        case 6:
        case 9:
        case 11: if(day<30)
                        tomm_day=day+1;
                 else
                   {
                        tomm_day=1;
                        tomm_month=month+1;
                   }
                break;

        case 12: if(day<31)
                        tomm_day=day+1;
                else
                   {
                        tomm_day=1;
                        tomm_month=1;
```

```
                        if(year==2013)
                          {
                    printf("the next day is out of boundary value of year\n");
                            tomm_year=year+1;
                          }
                        else
                           tomm_year=year+1;
                    }
                break;
        case 2:
                if(day<28)
                        tomm_day=day+1;
                else if(isleap(year)&& day==28)
                                tomm_day=day+1;
                 else if(day==28 || day==29)
                        {
                          tomm_day=1;
                          tomm_month=3;
                        }
                break;
}
printf("next day is : %d  %d  %d",tomm_day,tomm_month,tomm_year);
return 0;}
```

Test Case Name : Boundary Value Analysis test cases for Next date program

Experiment Number : 3

Test data : Enter the three integer value

Pre-condition : Month 1 to 12 , DAY 1 TO 31 AND YEAR 1812 TO 2013 / we consider one corner for the input space

Brief Description :

	Min	Min +1	Normal	Max -1	Max
Month	1	2	6	11	12
Day	1	2	15	30	31
Year	1812	1813	1912	2012	2013

Next date Output Boundary Value Analysis Cases

Case Id	Description	Input Data			Expected Output			Actual output			Status	Comment
		Month	day	Year	Month	Day	year	Month	day	year		
1	Enter the min value month, day and year	1	1	1812	1	2	1812					
2	Enter the min+1 value for year and min for month and day	1	1	1813	1	2	1813					
3	Enter the normal value for year and min for month and day	1	1	1912	1	2	1912					
4	Enter the max -1 value for year and min for month and day	1	1	2012	1	2	2012					
5	Enter the max value for year and min for month and day	1	1	2013	1	2	2013					
6	Enter the min+1 value of day and min for month and year	1	2	1812	1	3	1812					

7	Enter the min+1 value for day and year and min for month	1	2	1813	1	3	1813					
8	Enter the min+1 value for day , normal value for year and min value for month	1	2	1912	1	3	1912					
9	Enter the min+1 value for day , max -1 value for year and min value for month	1	2	2012	1	3	2012					
10	Enter the min+1 value for day , max value for year and min value for month	1	2	2013	1	3	2013					
11	Enter the normal value of day and min for year and month	1	15	1812	1	16	1812					
12	Enter the normal value for day and min+1 for year and min for month	1	15	1813	1	16	1813					
13	Enter the normal value for day normal value for year and min value for month	1	15	1912	1	16	1912					
14	Enter the normal value for day , max -1 value for year and min value for month	1	15	2012	1	16	2012					
15	Enter the normal value for day , max value for year and min value for month	1	15	2013	1	16	2013					
16	Enter the max - 1 value of day and min for day and year	1	30	1812	1	31	1812					
17	Enter the max -1 value for day and min for month and min+1 for year	1	30	1813	1	31	1813					
18	Enter the max - 1 value for day , normal value for year and min value for month	1	30	1912	1	31	1912					

19	Enter the max - 1 value for day , max -1 value for year and min value for month	1	30	2012	1	31	2012					
20	Enter the max -1 value for day , max value for year and min value for month	1	30	2013	1	31	2013					
21	Enter the max value of day and min for year and month	1	31	1812	2	1	1812					
22	Enter the max value for day and min for month and min + 1 for year	1	31	1813	2	1	1813					
	Enter the max value for day , normal value for year and min value for month	1	31	1912	2	1	1912					
24	Enter the max value for day , max -1 value for year and min value for month	1	31	2012	2	1	2012					
25	Enter the max value for day , max value for year and min value for month	1	31	2013	2	1	2013					

Case Id	Description	Input Data			Expected Output			Actual output			Status	
		month	day	Year	month	Day	year	month	day	year		Comment
1	Enter the D1, M1 and Y1 valid cases	12	31	1811	Should display the message value of the year in range 1812..2013							
2	Enter the D1, M1 and Y2 valid cases	12	31	2012	1	1	2013					
3	Enter the D1, M1 and Y3 valid cases	12	31	2013	Should display the message Next is out of boundary 2013							

Test Case Name : Equivalence class test cases for Next date

Experiment Number : 5

Test data : Enter the three integer value

Pre-condition : Month 1 to 12 , DAY 1 TO 31 AND YEAR 1812 TO 2013

Valid Cases

M1 = { month ; 1 ≤ month ≤ 12 }

D1 = { day : 1 ≤ day ≤ 31 }

Y1 = { year : 1812 ≤ year ≤ 2013 }

Invalid cases

M2 = {month : month < 1}

M3 = {month : month > 12}

D2 = {day : day < 1}

D3 = {day : day > 31}

Y2 = {year : year < 1812}

Y3 = {year : year > 2013}

Next date Output Equivalence Class Testing

(Weak and Strong Normal Equivalence Class)

Case Id	Description	Input Data			Expected Output			Actual output			Status	Comment
		month	day	year	month	day	year	month	day	year		
WN1,SN1	Enter the M1, D1 and Y1 valid cases	6	15	1912	6	16	1912					

(Weak Robustness Equivalence Class)

Case Id	Description	Input Data			Expected Output			Actual output			Status	Comment
		month	day	year	month	Day	year	month	day	year		
WR1	Enter the M1, D1 and Y1 cases	6	15	1912	6	16	1912					
WR2	Enter the M2 , D1 and Y1 cases	-1	15	1912	Should display the message value of the month not in the range 1..12							
WR3	Enter the M3 ,D1 and Y1 cases	13	15	1912	Should display the message value of the month not in the range 1..12							
WR4	Enter the M1, D2 and Y1 cases	6	-1	1912	Should display the message value of the day not in the range 1..31							
WR5	Enter the M1, D3 and Y1 cases	6	32	1912	Should display the message value of the day not in the range 1..31							
WR6	Enter the M1, D1 and Y2 cases	6	15	1811	Should display the message value of the year not in the range 1812..2013							
WR7	Enter the M1, D1 and Y3 cases	6	15	2014	Should display the message value of the year not in the range 1812..2013							

(Strong Robustness Equivalence Class)

Case Id	Description	Input Data			Expected Output	Actual Output	Status	Comment
		month	day	year				
SR1	Enter the M2 , D1 and Y1 cases	-1	15	1912	Should display the message value of the month not in the range 1..12			
SR2	Enter the M1, D2 and Y1 cases	6	-1	1912	Should display the message value of the day not in the range 1..31			
SR3	Enter the M1, D1 and Y2 cases	6	15	1811	Should display the message value of the year not in the range 1812..2013			
SR4	Enter the M2 , D2 and Y1 cases	-1	-1	1912	(**i**)Should display the message value of the month in range 1..12			
					(ii) Should display the message value of the day in range 1..31			
SR5	Enter the M1, D2 and Y2 cases	6	-1	1811	**(i)** Should display the message value of the day in range 1..31			
					(ii) Should display the message value of the year in range 1812..2013			
SR6	Enter the M2, D1 and Y2 cases	-1	15	1811	(**i**) Should display the message value of the month in range 1..12			
					(ii) Should display the message value of the year in range 1812..2013			
SR7	Enter the M2, D2 and Y2 cases	-1	-1	1811	(**i**)Should display the message value of the month in range 1..12			
					(ii) Should display the message value of the day in range 1..31			

					(iii) Should display the message value of the year in range 1812..2013			

Case Id	Description	Input Data			Expected Output			Actual Output			Status	Comment
		day	month	year	day	Month	year	day	month	year		
1	Enter the D1, M1 and Y1 valid cases	31	12	1811	Should display the message value of the year in range 1812..2013							
2	Enter the D1, M1 and Y2 valid cases	31	12	2012	1	1	2013					
3	Enter the D1, M1 and Y3 valid cases	31	12	2013	Should display the message Next is out of boundary 2013							

LIST OF Viva Questions

1. What are the importance of Software Testing?
2. What are the main tools you are used for Software Testing?
3. What are the different types of Software Testing?
4. What are the difference between Black Box and White Box testing?
5. What are the difference between Manual Testing and Automated Testing ?
6. What is Unit Testing ?
7. What is Integration Testing ?
8. What is acceptance testing ?
9. What is Static testing?
10. What is System testing?
11. What is Load Testing?
12. What is Smoke Testing?
13. What is Soak Testing?
14. What is Scalability Testing?
15. What is Sanity Testing?
16. What is Ramp Testing?
17. What is Monkey Testing?
18. What is Gray Box Testing?
19. What is Functional Testing?
20. What is Glass Box Testing?
21. What is Dynamic Testing?
22. What is Compatibility Testing?
23. What is Concurrency Testing?
24. What is Component Testing?
25. What is Ad Hoc Testing?
26. What is Agile Testing?
27. What are the different phases in Software Testing?
28. How you define defects and Bugs?
29. What are the roles of a QA specialist?
30. What is Test Case and Test Plan ?
31. Tell me about Top Down and Bottom Up approaches in testing?
32. Tell me about Software Testing Life cycle

What is 'software Testing'?

Software testing is the act of operating a system or application under control and then assessing their results. It is intentionally simulating a problem situation in order to work out a possible remedy in case a situation like that actually happens.

What kind of testing should we consider?

The basic testing to consider include Blackbox testing, Integration testing, Whitebox testing, User acceptance testing, Load testing, Acceptance testing, Performance testing, and Smoke testing.

What is Software 'quality'?

Quality software is software that is reasonably free from bug, is up to the requirements and/or expectations, delivered on time and according to the budget, and is easy to maintain.

What is 'Software Quality Assurance'?
Software Quality Assurance also known as Software QA, encompasses the entire process of developing software: observing and improving the process, seeing that the standards and procedures agreed upon are followed, and making sure that problems are discovered and also fixed.

Does every software project need testers?
This solely depends on the context or size of the project, the methodology of development, risks involved, and the experience and skills of the developers.

What is verification? And what is validation?
Verifications are reviews and meetings which are intended to help evaluate documents, code, specifications, and requirements. It is usually done with checklists, walkthroughs, issues lists, and inspection meetings.

Validation, on the other hand, entails the real testing and it is done after completing the verification.

What is a 'test plan'? What is a 'test case'?
A software project test plan is a document that explains the objective, approach, focus, and scope of a software testing. A test case, on the other hand, is a document that explains an action, input, or event, and a response that is expected, to help decide if an application's feature is functioning properly.

Why does software have bugs?
Software has bugs for the following reasons

Errors in programming
Lack of communication or no communications at all
Requirements changes
Time pressures

What should you do after finding a bug?
You need to report the bug and then assign it to developers that can take care of it. When you have fixed the problem, retest the fixes, decide the requirements for regression testing in order to be sure that no more problems are eventually caused by the fixes.

How can you introduce new Software QA processes to an existing Organization?
This is usually determined by the organization's size as well as the risks involved. If it is a small project or group, it may be better to use a more ad-hoc process, depending on the customer and project type. It can also be done via incremental self-managed team approaches.

What steps are needed in developing and running software test?

Get the requirements, functional design, and plan for the internal design, and other documents that are relevant.
Get budget and the schedule conditions
Decide project framework
Recognize risks
Decide testing approaches, process, test setting, test data
Carry out test
Carry out reviews and evaluations
Sustain and keep documents up to date

What is Boundary value testing?

Test boundary conditions on, below and above the edges of input and output equivalence classes. For instance, let say a bank application where you can withdraw maximum Rs.20,000 and a minimum of Rs.100, so in boundary value testing we test only the exact boundaries, rather than hitting in the middle. That means we test above the maximum limit and below the minimum limit.

What is failure?

Failure is a departure from specified behavior.

What is V-Model?

A software development model that illustrates how testing activities integrate with software development phases

What is test coverage?

Test coverage measures in some specific way the amount of testing performed by a set of tests (derived in some other way, e.g., using specification-based techniques).

What is component testing?

Component testing, also known as unit, module, and program testing, searches for defects in and verifies the functioning of software (e.g., modules, programs, objects, classes, etc.) that are separately testable.

What is Alpha testing?

Pre-release testing by end user representatives at the developer's site.

What is beta testing?

Testing performed by potential customers at their own locations.

What is White Box Testing?

White Box Testing is also called as Glass Box, Clear Box, and Structural Testing. It is based on applications internal code structure. In white-box testing, an internal perspective of the system, as well as programming skills, are used to design test cases.

What is Black Box Testing?
Black Box Testing is a software testing method in which testers evaluate the functionality of the software under test without looking at the internal code structure.

What is Test Suite?
Test Suite is a collection of test cases. The test cases which are intended to test an application.

What is Test Scenario?
Test Scenario gives the idea of what we have to test. Test Scenario is like a high-level test case.

What is Test Case?
Test cases are the set of positive and negative executable steps of a test scenario which has a set of pre-conditions, test data, expected result, post-conditions and actual results

What is Unit Testing?
Unit Testing is also called as Module Testing or Component Testing. It is done to check whether the individual unit or module of the source code is working properly. It is done by the developers in the developer's environment.

What is Integration Testing?
Integration Testing is the process of testing the interface between the two software units. Integration testing is done by three ways. Big Bang Approach, Top-Down Approach, Bottom-Up Approach

What is System Testing?
Testing the fully integrated application to evaluate the system's compliance with its specified requirements is called System Testing AKA End to End testing. Verifying the completed system to ensure that the application works as intended or not.

What is Functional Testing?
In simple words, what the system actually does is functional testing. To verify that each function of the software application behaves as specified in the requirement document.

What is Non-Functional Testing?
In simple words, how well the system performs is non-functionality testing. Non-functional testing refers to various aspects of the software such as performance, load, stress, scalability, security, compatibility etc., Main focus is to improve the user experience on how fast the system responds to a request.

What is Acceptance Testing?

It is also known as pre-production testing. This is done by the end users along with the testers to validate the functionality of the application. After successful acceptance testing.

What is Gamma Testing?
Gamma testing is done when the software is ready for release with specified requirements. It is done at the client place. It is done directly by skipping all the in-house testing activities.

What is Smoke Testing?
Smoke Testing is done to make sure if the build we received from the development team is testable or not. It is also called as "Day 0" check. It is done at the "build level".

What is Sanity Testing?
Sanity Testing is done during the release phase to check for the main functionalities of the application without going deeper. It is also called as a subset of Regression testing.

What is Regression Testing?
Repeated testing of an already tested program, after modification, to discover any defects introduced or uncovered as a result of the changes in the software being tested or in another related or unrelated software components.

What is Load Testing?
It is to verify that the system/application can handle the expected number of transactions and to verify the system/application behavior under both normal and peak load conditions.

What is Stress Testing?
It is to verify the behavior of the system once the load increases more than its design expectations.

What is Walk Through?
A walkthrough is an informal meeting conducts to learn, gain understanding, and find defects.

What is a Defect?
The variation between the actual results and expected results is known as a defect. If a developer finds an issue and corrects it by himself in the development phase then it's called a defect.

What is a Bug?
If testers find any mismatch in the application/system in testing phase then they call it as Bug

What is an Error?
We can't compile or run a program due to a coding mistake in a program. If a developer unable to successfully compile or run a program then they call it as an error.

What is a Failure?

Once the product is deployed and customers find any issues then they call the product as a failure product. After release, if an end user finds an issue then that particular issue is called as a failure.