SOFTWARE TESTING LABORATORY

Subject Code: 10ISL68 I.A. Marks: 25
Hours/Week: 03 Exam Hours: 03
Total Hours: 42 Exam Marks: 50

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

2. 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 boundaryvalue analysis, execute the test cases and discuss the results.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

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

10. 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.

11. 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.

12. 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.

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

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

PROGRAM CODE:

```
#include<stdio.h>
#include<ctype.h>
#include<conio.h>
#include<process.h>
int main()
{
        int a, b, c;
        clrscr();
        printf("Enter three sides of the triangle");
        scanf("%d%d%d", &a, &b, &c);
        if((a<b+c)&&(b<a+c)&&(c<a+b))
                {
                        if((a==b)&&(b==c))
                                {
                                printf("Equilateral triangle");
                                }
                        else if((a!=b)&&(a!=c)&&(b!=c))
                                {
                                printf("Scalene triangle");
                                }
                                else
                                printf("Isosceles triangle");
                }
        else
                {
                printf("triangle cannot be formed");
                }
getch();
return 0;

}
```

Test Report:

Input data decision Table

RULES		**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**
Conditions	C1: a < b + c											
	C2 : b < a + c											
	C3 : c < a + b											
	C4 : a = b											
	C5 : a = c											
	C6 : b = c											
Actions	a 1 : Not a triangle											
	a 2 : Scalene triangle											
	a3 : Isosceles triangle											
	a 4 : Equilateral triangle											
	a 5 : Impossible											

2. 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.

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

PROGRAM CODE:

```
#include<stdio.h>
#include<ctype.h>
#include<conio.h>
#include<process.h>
int main()
{
        int a, b, c;
        clrscr();

        printf("Enter three sides of the triangle");
        scanf("%d%d%d", &a, &b, &c);

        if((a > 10) || (b > 10) || (c > 10))
        {
                printf("Out of range");
                getch();
                exit(0);
        }
        if((a<b+c)&&(b<a+c)&&(c<a+b))
        {
                if((a==b)&&(b==c))
                {
                        printf("Equilateral triangle");
                }
                else if((a!=b)&&(a!=c)&&(b!=c))
                {
                        printf("Scalene triangle");
                }
                else
                printf("Isosceles triangle");
        }
        else
        {
                printf("triangle cannot be formed");
        }
        getch();
        return 0;
}
```

Test Report:

Case Id	Description	Input Data			Expected Output	Actual Output	Comments
		a	b	c			

3. 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.

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

PROGRAM CODE

```
#include<stdio.h>
#include<ctype.h>
#include<conio.h>
#include<process.h>
int main()
{
        int a, b, c;
        clrscr();
        printf("Enter three sides of the triangle");
        scanf("%d%d%d", &a, &b, &c);
        if((a > 10) || (b > 10) || (c > 10))
        {
                printf("Out of range");
                getch();
                exit(0);
        }
        if((a<b+c)&&(b<a+c)&&(c<a+b))
        {
                if((a==b)&&(b==c))
                {
                printf("Equilateral triangle");
                }
                else if((a!=b)&&(a!=c)&&(b!=c))
                {
                printf("Scalene triangle");
                }
                else
                printf("Isosceles triangle");
        }
        else
        {
        printf("triangle cannot be formed");
        }
        getch();
        return 0;
}
```

Test Report:

Case Id	Description	Input Data			Expected Output	Actual Output	Comments
		a	b	c			

4. 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.

ALGORITHM

STEP 1: Define lockPrice=45.0, stockPrice=30.0, barrelPrice=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 lockSales, stockSales, barrelSales and sales

STEP6: output("Total 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

PROGRAM CODE:

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int locks, stocks, barrels, t_sales, flag = 0;
        float commission;
        clrscr();
        printf("Enter the total number of locks");
        scanf("%d",&locks);
        if ((locks <= 0) || (locks > 70))
        {
                flag = 1;
        }
        printf("Enter the total number of stocks");
        scanf("%d",&stocks);
        if ((stocks <= 0) || (stocks > 80))
        {
                flag = 1;
        }
        printf("Enter the total number of barrelss");
        scanf("%d",&barrels);
        if ((barrels <= 0) || (barrels > 90))
        {
                flag = 1;
        }
        if (flag == 1)
        {
                printf("invalid input");
                getch();
                exit(0);
        }

        t_sales = (locks * 45) + (stocks * 30) + (barrels * 25);

        if (t_sales <= 1000)
        {
                commission = 0.10 * t_sales;
```

```
        }
        else if (t_sales < 1800)
        {
                commission = 0.10 * 1000;
                commission = commission + (0.15 * (t_sales - 1000));
        }
        else
        {
                commission = 0.10 * 1000;
                commission = commission + (0.15 * 800);
                commission = commission + (0.20 * (t_sales - 1800));
        }
        printf("The total sales is %d \n The commission is %f",t_sales,
        commission);
        getch();
        return;
}
```

Test Report:

Case Id	Description	Input Data			Expected Output	Actual Output	Comments
		a	b	c			

5. 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.

ALGORITHM

STEP 1: Define lockPrice=45.0, stockPrice=30.0, barrelPrice=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 lockSales, stockSales, barrelSales and sales

STEP6: output("Total 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

PROGRAM CODE:

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int locks, stocks, barrels, t_sales, flag = 0;
        float commission;
        clrscr();
        printf("Enter the total number of locks");
        scanf("%d",&locks);
        if ((locks <= 0) || (locks > 70))
        {
                flag = 1;
        }
        printf("Enter the total number of stocks");
        scanf("%d",&stocks);
        if ((stocks <= 0) || (stocks > 80))
        {
                flag = 1;
        }
        printf("Enter the total number of barrelss");
        scanf("%d",&barrels);
        if ((barrels <= 0) || (barrels > 90))
        {
                flag = 1;
        }
        if (flag == 1)
        {
                printf("invalid input");
                getch();
                exit(0);
        }

        t_sales = (locks * 45) + (stocks * 30) + (barrels * 25);

        if (t_sales <= 1000)
        {
                commission = 0.10 * t_sales;
        }
```

```
        else if (t_sales < 1800)
        {
                commission = 0.10 * 1000;
                commission = commission + (0.15 * (t_sales - 1000));
        }
        else
        {
                commission = 0.10 * 1000;
                commission = commission + (0.15 * 800);
                commission = commission + (0.20 * (t_sales - 1800));
        }
        printf("The total sales is %d \n The commission is %f",t_sales,
        commission);
        getch();
        return;
}
```

Test Report:

Case Id	Description	Input Data			Expected Output	Actual Output	Comments
		a	b	c			

6. 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.

ALGORITHM

STEP 1: Define lockPrice=45.0, stockPrice=30.0, barrelPrice=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 lockSales, stockSales, barrelSales and sales

STEP6: output("Total 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

PROGRAM CODE:

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int locks, stocks, barrels, t_sales, flag = 0;
        float commission;
        clrscr();
        printf("Enter the total number of locks");
        scanf("%d",&locks);
        if ((locks <= 0) || (locks > 70))
        {
                flag = 1;
        }
        printf("Enter the total number of stocks");
        scanf("%d",&stocks);

        if ((stocks <= 0) || (stocks > 80))
        {
                flag = 1;
        }
        printf("Enter the total number of barrelss");
        scanf("%d",&barrels);
        if ((barrels <= 0) || (barrels > 90))
        {
                flag = 1;
        }
        if (flag == 1)
        {
                printf("invalid input");
                getch();
                exit(0);
        }

        t_sales = (locks * 45) + (stocks * 30) + (barrels * 25);

        if (t_sales <= 1000)
        {
                commission = 0.10 * t_sales;
```

```
        }
        else if (t_sales < 1800)
        {
                commission = 0.10 * 1000;
                commission = commission + (0.15 * (t_sales - 1000));
        }
        else
        {
                commission = 0.10 * 1000;
                commission = commission + (0.15 * 800);
                commission = commission + (0.20 * (t_sales - 1800));
        }
        printf("The total sales is %d \n The commission is %f",t_sales,
        commission);
        getch();
        return;
}
```

Test Report:

Case Id	Description	Input Data			Expected Output	Actual Output	Comments
		a	b	c			

7. 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.

ALGORITHM

Step 1: Input 3 integer numbers which represents number of Locks, Stocks and Barrels sold.

Step 2: compute the total sales =(Number of Locks sold *45) + (Number of Stocks sold *30) + (Number of Barrels sold *25)

Step 3: if a totals sale in dollars is less than or equal to $1000
then commission = 0.10* total Sales do step 6

Step 4: else if total sale is less than $1800
then commission1 = 0.10* 1000
commission = commission1 + (0.15 * (total sales – 1000))
do step 6

Step 5: else commission1 = 0.10* 1000
commission2 = commission1 + (0.15 * 800))
commission = commission2 + (0.20 * (total sales – 1800)) do step 6

Step 6: Print commission.

Step 7: Stop.

PROGRAM CODE:

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int locks, stocks, barrels, t_sales, flag = 0;
        float commission;
        clrscr();
        printf("Enter the total number of locks");
        scanf("%d",&locks);
        if ((locks <= 0) || (locks > 70))
        {
                flag = 1;
        }
        printf("Enter the total number of stocks");
        scanf("%d",&stocks);
        if ((stocks <= 0) || (stocks > 80))
        {
                flag = 1;
        }
        printf("Enter the total number of barrelss");
        scanf("%d",&barrels);
        if ((barrels <= 0) || (barrels > 90))
        {
                flag = 1;
        }
        if (flag == 1)
        {
                printf("invalid input");
                getch();
                exit(0);
        }

        t_sales = (locks * 45) + (stocks * 30) + (barrels * 25);

        if (t_sales <= 1000)
        {
                commission = 0.10 * t_sales;
        }
```

```
        else if (t_sales < 1800)
        {
                commission = 0.10 * 1000;
                commission = commission + (0.15 * (t_sales - 1000));
        }
        else
        {
                commission = 0.10 * 1000;
                commission = commission + (0.15 * 800);
                commission = commission + (0.20 * (t_sales - 1800));
        }
        printf("The total sales is %d \n The commission is %f",t_sales,
        commission);
        getch();
        return;
}
```

Test Report:

RULES		**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**
Conditions	C1:											
	C2 :											
	C3 :											
	C4 :											
	C5 :											
	C6 :											
Actions	a1 :											
	a2 :											
	a3 :											
	a4 :											
	a5 :											

8. 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.

ALGORITHM

Step 1: Input value of ‘n’. Enter ‘n’ integer numbers in array int mid;

Step 2: Initialize low = 0, high = n -1

Step 3: until (low <= high) do
mid = (low + high) / 2
if (a[mid] == key)
then do Step 5
else if (a[mid] > key)
then do
high = mid - 1
else
low = mid + 1

Step 4: Print unsuccessful search do step 6.

Step 5: Print Successful search. Element found at position mid+1.

Step 6: Stop.

PROGRAM CODE:

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int a[20],n,low,high,mid,key,i,flag=0;
        clrscr();
        printf("Enter the value of n:\n");
        scanf("%d",&n);
        if(n>0)
        {
                printf("Enter %d elements in ASCENDING order\n",n);
                for(i=0;i<n;i++)
                {
                        scanf("%d",&a[i]);
                }
                printf("Enter the key element to be searched\n");
                scanf("%d",&key);
                low=0;
                high=n-1;
                while(low<=high)
                {
                        mid=(low+high)/2;
                        if(a[mid]==key)
                        {
                                flag=1;
                                break;
                        }
                        else if(a[mid]<key)
                        {
                                low=mid+1;
                        }
                        else
                        {
                                high=mid-1;
                        }
                }
```

```
        if(flag==1)
                printf("Successful search\n Element found at Location
                %d\n",mid+1);
        else
                printf("Key Element not found\n");
        }

        else
                printf("Wrong input");
        getch();
        return 0;
}
```

Test Report:

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

PROGRAM CODE:

```
#include <stdio.h>
void swap ( int* a, int* b )
{
	int t = *a;
	*a = *b;
	*b = t;
}

int partition (int arr[], int l, int h)
{
	int x = arr[h];
	int i = (l - 1),j;
	for (j = l; j <= h- 1; j++)
	{
		if (arr[j] <= x)
		{
			i++;
			swap (&arr[i], &arr[j]);
		}
	}
	swap (&arr[i + 1], &arr[h]);
	return (i + 1);
}

void quickSortIterative (int arr[], int l, int h)
{
	int stack[10],p;
	int top = -1;
	stack[ ++top ] = l;
	stack[ ++top ] = h;
	while ( top >= 0 )
	{
		h = stack[ top-- ];
```

```
            l = stack[ top-- ]
            p = partition( arr, l, h );
            if ( p-1 > l )
            {
                  stack[ ++top ] = l;
                  stack[ ++top ] = p - 1;
            }
            if ( p+1 < h )
            {
                  stack[ ++top ] = p + 1;
                  stack[ ++top ] = h;
            }
      }
}

int main()
{
      int arr[20],n,i;
      clrscr();

      printf("Enter the size of the array");
      scanf("%d",&n);

      printf("Enter %d elements",n);
      for(i=0;i<n;i++)
      scanf("%d",&arr[i]);

      quickSortIterative( arr, 0, n - 1 );

      printf("Elements of the array are;");
      for(i=0;i<n;i++)
      printf("%d",arr[i]);

      getch();
      return 0;
}
```

Test Report:

10. 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 CODE:

```
#include<stdio.h>
main()
{
        float kan,eng,hindi,maths,science, sst,avmar;

        printf("Letter Grading\n");
        printf("SSLC Marks Grading\n");

        printf("Enter the marks for Kannada:");
        scanf("%f",&kan);

        printf("enter the marks for English:");
        scanf("%f",&eng);

        printf("enter the marks for Hindi:");
        scanf("%f",&hindi);

        printf("enter the marks for Maths");
        scanf("%f",&maths);

        printf("enter the marks for Science:");
        scanf("%f",&science);

        printf("enter the marks for Social Science:");
        scanf("%f",&sst);

        avmar=(kan+eng+hindi+maths+science+sst)/6.25;
        printf("the average marks are=%f\n",avmar);

        if((avmar<35)&&(avmar>0))
                printf("fail");

        else if((avmar<=40)&&(avmar>35))
```

```
            printf("Grade C");
        else if((avmar<=50)&&(avmar>40))
            printf("Grade C+");

        else if((avmar<=60)&&(avmar>50))
            printf("Grade B");

        else if((avmar<=70)&&(avmar>60))
            printf("Grade B+");

        else if((avmar<=80)&&(avmar>70))
            printf("Grade A");

        else if((avmar<=100)&&(avmar>80))
            printf("Grade A+");
}
```

Test Report:

11. 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.

ALGORITHM

STEP 1: Input date in format DD.MM.YYYY

STEP2: if MM is 01, 03, 05,07,08,10 do STEP3 else STEP6

STEP3:if DD < 31 then do STEP4 else if DD=31 do STEP5 else
output(Invalid Date);

STEP4: tomorrowday=DD+1 goto STEP18

STEP5: tomorrowday=1; tomorrowmonth=month + 1 goto STEP18

STEP6: if MM is 04, 06, 09, 11 do STEP7

STEP7: if DD<30 then do STEP4 else if DD=30 do STEP5 else
output(Invalid Date);

STEP8: if MM is 12

STEP9: if DD<31 then STEP4 else STEP10

STEP10: tomorrowday=1, tommorowmonth=1,
tommorowyear=YYYY+1; goto STEP18

STEP11: if MM is 2

STEP12: if DD<28 do STEP4 else do STEP13

STEP13: if DD=28 & YYYY is a leap do STEP14 else STEP15

STEP14: tommorowday=29 goto STEP18

STEP15: tommorowday=1, tomorrowmonth=3, goto STEP18;

STEP16: if DD=29 then do STEP15 else STEP17

STEP17: output("Cannot have feb", DD); STEP19

STEP18: output(tomorrowday, tomorrowmonth, tomorrowyear);

STEP19: exit

PROGRAM CODE:

```
#include<stdio.h>
#include<conio.h>
main( )
{
        int
        month[12]={31,28,31,30,31,30,31,31,30,31,30,31};
        int d,m,y,nd,nm,ny,ndays;
        clrscr( );
        printf("enter the date,month,year");
        scanf("%d%d%d",&d,&m,&y);

        ndays=month[m-1];

        if(y<=1812 && y>2012)
        {
                printf("Invalid Input Year");
                exit(0);
        }

        if(d<=0 || d>ndays)
        {
                printf("Invalid Input Day");
                exit(0);
        }

        if(m<1 && m>12)
        {
                printf("Invalid Input Month");
                exit(0);
        }

        if(m==2)
        {
                if(y%100==0)
                {
                        if(y%400==0)
                        ndays=29;
                }
```

```
            else if(y%4==0)
                    ndays=29;
        }

        nd=d+1;
        nm=m;
        ny=y;

        if(nd>ndays)
        {
                nd=1;
                nm++;
        }

        if(nm>12)
        {
                nm=1;
                ny++;
        }

        printf("\n Given date is %d:%d:%d",d,m,y);
        printf("\n Next day's date is %d:%d:%d",nd,nm,ny);
        getch( );
}
```

Test Report:

12. 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.

ALGORITHM

STEP 1: Input date in format DD.MM.YYYY

STEP2: if MM is 01, 03, 05,07,08,10 do STEP3 else STEP6

STEP3:if DD < 31 then do STEP4 else if DD=31 do STEP5 else
output(Invalid Date);

STEP4: tomorrowday=DD+1 goto STEP18

STEP5: tomorrowday=1; tomorrowmonth=month + 1 goto STEP18

STEP6: if MM is 04, 06, 09, 11 do STEP7

STEP7: if DD<30 then do STEP4 else if DD=30 do STEP5 else
output(Invalid Date);

STEP8: if MM is 12

STEP9: if DD<31 then STEP4 else STEP10

STEP10: tomorrowday=1, tommorowmonth=1, tommorowyear=YYYY+1;
goto STEP18

STEP11: if MM is 2

STEP12: if DD<28 do STEP4 else do STEP13

STEP13: if DD=28 & YYYY is a leap do STEP14 else STEP15

STEP14: tommorowday=29 goto STEP18

STEP15: tommorowday=1, tomorrowmonth=3, goto STEP18;

STEP16: if DD=29 then do STEP15 else STEP17

STEP17: output("Cannot have feb", DD); STEP19

STEP18: output(tomorrowday, tomorrowmonth, tomorrowyear);

STEP19: exit

PROGRAM CODE:

```
#include<stdio.h>
#include<conio.h>
main( )
{
        int month[12]={31,28,31,30,31,30,31,31,30,31,30,31};
        int d,m,y,nd,nm,ny,ndays;
        clrscr( );
        printf("enter the date,month,year");
        scanf("%d%d%d",&d,&m,&y);
        ndays=month[m-1];
        if(y<=1812 && y>2012)
        {
                printf("Invalid Input Year");
                exit(0);
        }
        if(d<=0 || d>ndays)
        {
                printf("Invalid Input Day");
                exit(0);
        }
        if(m<1 && m>12)
        {
                printf("Invalid Input Month");
                exit(0);
        }
        if(m==2)
        {
                if(y%100==0)
                {
                        if(y%400==0)
                        ndays=29;
                }
                else if(y%4==0)
                ndays=29;
        }

        nd=d+1;
        nm=m;
```

```
        ny=y;

        if(nd>ndays)
        {
                nd=1;
                nm++;
        }
        if(nm>12)
        {
                nm=1;
                ny++;
        }
        printf("\n Given date is %d:%d:%d",d,m,y);
        printf("\n Next day's date is %d:%d:%d",nd,nm,ny);
        getch( );
}
```

Test Report: