SOFTWARE TESTING LABORATORY

Subject Code: 18ISL66 I.AMarks:40
Hours/Week: 03 ExamHours: 03
Total Hours: 36 Exam Marks: 60
Number of Lecture Hours/Week: 01I + 02P

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 boundaryvalue 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. 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.

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.

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<stdlib.h.h>
int main()
{
        int a, b, c,c1,c2,c3;
        char istriangle, y ,n;
        clrscr();
        do
        {
        printf("Enter three sides of the triangle a,b & c");
        scanf("%d%d%d", &a, &b, &c);
        c1= a>=1 && a<=10;
        c2= b>=1 && b<=10;
        c3= c>=1 && c<=10;
        if(!c1)
              printf("value of a Out of range");
        if(!c2)
                printf("value of b Out of range ");
        if(!c3)
                printf("value of c Out of range ");
       } while(!(c1&&c2&&c3));

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

if(istriangle==’y’)
{
 if(a==b) && (b==c))
        printf(“equilateral triangle \n”);
else if((a!=b)&&(b!=c)&&(c!=a))
        printf(“scalene triangle\n”);
else
        printf(“isosceles triangle\n”);
}
else
        printf(“ not a triangle \n”);
        return 0;
```

}

Test Report:

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

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.

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, 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;
        tbarrels=0;
        tstocks=0;

        printf("Enter 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 no. of stocks and barrels “);
                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
                              “,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
                      ",temp);
        else
                 tstocks=temp;
}
printf("total socks=%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 ",temp);
                else
                         tbarrels=temp;
}
printf("total barrels=%d\n",tbarrels);
printf("enter the no. of locks & to exit the loop enter -1 for locks \n");
scanf("%d",&locks);
}
printf("total  locks=%d \n total stocks =%d \n total barrels=%d
\n",tlocks,tstocks,tbarrels)


sales = lprice*tlocks+sprice*tstocks+bprice*tbarrels;
printf("\n total sales=%f \n", sales);

if (sales >0)
{
```

```
            if (sales > 1800.0)
            {
                    comm = 0.10 * 1000;
                    comm = comm + 0.15 * 800;
                    comm = comm + (0.20 * (sales - 1800));

            }
        else if(sales > 1000)
            {
                    comm = 0.10 * 1000;
                    comm = comm + 0.15 * (sales-1000);
            }
        else
                  comm = 0.10 * 1000;
                    printf("The total commission is %f",comm);
}
else
printf("there is no sales \n");
        return 0;
}
```

Test Report:

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

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.

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>
        Int check (int day,int month)
        {
                If((month==4||month==6 ||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("\n enter the todays date in the form of ddmmyyyy \n");
        scanf("%d%d%d",&day,&month,&year);
        tomm_month=month;
        tomm_year=year;
        if(day<1 || day>31)
        {
                printf(" value of the day not in the range 1…31\n");
                flag='n';
        }
                if(month<1||month>12)
```

```
        {
                printf(" value of the month not in the range 1---12\n");
                flag='n';
        }
else if(check(day,month))
{
        printf("the value of the day not in the range day<=30");
        flag='n';
}
if(year<=1812 || year >2018)
{
        printf("the value of the year not in the range 1812…2018 \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==2018)
                   {
                         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("nextday is %d%d%d",tomm_day, tomm_month, tomm_year);
    return 0;
    }
```

Test Report:

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.

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<stdlib.h.h>
int main()
{
        int a, b, c,c1,c2,c3;
        char istriangle,y,n;
        clrscr();
        do
        {
        printf("Enter three sides of the triangle a,b & c");
        scanf("%d%d%d", &a, &b, &c);
        c1= a>=1 && a<=10;
        c2= b>=1 && b<=10;
        c3= c>=1 && c<=10;
        if(!c1)
              printf("value of a Out of range");
        if(!c2)
                printf("value of b Out of range ");
        if(!c3)
                printf("value of c Out of range ");
       } while(!(c1&&c2&&c3));

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

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

return 0;
}
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 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, 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;
        tbarrels=0;
        tstocks=0;

        printf("Enter 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 no. of stocks and barrels ");
        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 ",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 ",temp);
  else
  tstocks=temp;
}
printf("total socks=%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 ",temp);
  else
  tbarrels=temp;
}
printf("total barrels=%d\n",tbarrels);
printf("enter the no. of locks & to exit the loop enter -1 for locks \n");
scanf("%d",&locks);
}
printf("total  locks=%d \n total stocks =%d \n total barrels=%d
\n",tlocks,tstocks,tbarrels)


sales = lprice*tlocks+sprice*tstocks+bprice*tbarrels;
printf("\n total sales=%f \n", sales);

if (sales >0)
{
if (sales > 1800.0)
{
        comm = 0.10 * 1000;
        comm = comm + 0.15 * 800;
        comm = comm + (0.20 * (sales - 1800));
```

```
        }
        else if(sales > 1000)
        {
                comm = 0.10 * 1000;
                comm = comm + 0.15 * (sales-1000);
        }
        else
                        comm = 0.10 * 1000;
        printf("The total commission is %f",comm);
}
else
printf("there is no sales \n");
        return 0;
}
```

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 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>
        int check (int day,int month)
        {
         if((month==4||month==6 ||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("\n enter the todays date in the form of ddmmyyyy \n");
        scanf("%d%d%d",&day,&month,&year);
        tomm_month=month;
        tomm_year=year;
        if(day<1 || day>31)
        {
         printf(" value of the day not in the range 1…31\n");
        flag='n';
        }
        if(month<1||month>12)
        {
        printf(" value of the month not in the range 1---12\n");
        flag='n';
        }
        else if(check(day,month))
        {
```

```
printf("the value of the day not in the range day<=30");
flag='n';
}
if(year<=1812 || year >2018)
{
printf("the value of the year not in the range 1812…2018 \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==2018)
                  {
                        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("nextday is %d%d%d",tomm_day, tomm_month, tomm_year);
return 0;
}
```

Test Report:

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.

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<stdlib.h.h>
int main()
{
        int a, b, c,c1,c2,c3;
        char istriangle,y,n;
        clrscr();
        do
        {
        printf("Enter three sides of the triangle a,b & c");
        scanf("%d%d%d", &a, &b, &c);
        c1= a>=1 && a<=10;
        c2= b>=1 && b<=10;
        c3= c>=1 && c<=10;
        if(!c1)
              printf("value of a Out of range");
        if(!c2)
                printf("value of b Out of range ");
        if(!c3)
                printf("value of c Out of range ");
       } while(!(c1&&c2&&c3));

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

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

```
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											

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.

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, 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;
        tbarrels=0;
        tstocks=0;

        printf("Enter 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 no. of stocks and barrels ");
        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 ",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 ",temp);
  else
  tstocks=temp;
}
printf("total socks=%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 ",temp);
  Else
  tbarrels=temp;
}
printf("total barrels=%d\n",tbarrels);
printf("enter the no. of locks & to exit the loop enter -1 for locks \n");
scanf("%d",&locks);
}
printf("total  locks=%d \n total stocks =%d \n total barrels=%d
\n",tlocks,tstocks,tbarrels)


sales = lprice*tlocks+sprice*tstocks+bprice*tbarrels;
printf("\n total sales=%f \n", sales);

if (sales >0)
{
if (sales > 1800.0)
{
        comm = 0.10 * 1000;
        comm = comm + 0.15 * 800;
        comm = comm + (0.20 * (sales - 1800));
```

```
        }
        else if(sales > 1000)
        {
                comm = 0.10 * 1000;
                comm = comm + 0.15 * (sales-1000);
        }
        else
                        comm = 0.10 * 1000;
        printf("The total commission is %f",comm);
}
else
Printf(“there is no sales \n”);
        return 0;
}
```

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 :											

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.

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, 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;
tbarrels=0;
tstocks=0;

printf("Enter 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 no. of stocks and barrels ");
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 ",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 ",temp);
```

```
  else
  tstocks=temp;
}
printf("total socks=%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 ",temp);
  else
  tbarrels=temp;
}
printf("total barrels=%d\n",tbarrels);
printf("enter the no. of locks & to exit the loop enter -1 for locks \n");
scanf("%d",&locks);
}
printf("total  locks=%d \n total stocks =%d \n total barrels=%d
\n",tlocks,tstocks,tbarrels)


sales = lprice*tlocks+sprice*tstocks+bprice*tbarrels;
printf("\n total sales=%f \n", sales);

if (sales >0)
{
if (sales > 1800.0)
{
        comm = 0.10 * 1000;
        comm = comm + 0.15 * 800;
        comm = comm + (0.20 * (sales - 1800));

}
else if(sales > 1000)
{
        comm = 0.10 * 1000;
        comm = comm + 0.15 * (sales-1000);
```

```
        }
        else
                        comm = 0.10 * 1000;
        printf("The total commission is %f",comm);
}
else
printf(“there is no sales \n”);
        return 0;
}
```

Test Report:

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

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.

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 bin(int x[ ],int low,int high,int key)
{
 int mid;
```

```
while(low<=high)
{
  mid=(low+high)/2;
   if(x[mid]==key)
     return[mid];

   if(x[mid]<key)
      low=mid+1;

  else
     high=mid-1;
}
    return -1;
}
int main()
{
        int a[20],n,key,i,n,success;
        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);
                success=bin(a,0,n-1,key);
                if(success>0)
                printf("element found in %d \n",success);
                else
                printf("element not found");
      }

else

                printf("no. of elements should be greater than zero \n");
        return(0);
```

}

Test Report:

11. 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 quicksort (int x[10], int first, int last)
{
   int temp,pivot,i,j;

if(first<last)
{
        pivot=first;
        i=first;
        j=last;

while(i<j)
{
  while(x[i]<=x[pivot]&& i<last)
        i++;
  while(x[j]>x[pivot])
        j--;
if(i<j)
{
                temp=x[i];
                x[i]=x[j];
                x[j]=temp;
  }
}
temp=x[pivot];
x[pivot]=x[j];
x[j]=temp;
quicksort(x,first,j-1);
quicksort(x,j+1,last);
 }
}

int main()
{
int a[20],i,key,n;
printf("enter the size of the array \n");
scanf("%d",&n);
if(n>0)
{
  printf("enter the elements of the array \n");
   for(i=0;i<n;i++)
```

```
  {
        scanf("%d",&a[i]);
        quicksort(a,0,n-1);
        printf("the elements are in sorted form \n");
        for(i=0;i<n;i++)
        printf("%d",a[i]);
}
else
 {
        printf("size of array is invalid\n");
 }
}
```

Test Report:

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 CODE:

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

```
int main()
{
        float per;
        char grade;

        printf("Enter the percentage \n");
        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("Excellent");
                        break;
          case 'B': printf("very good");
                        break;
          case 'C': printf("good");
                        break;
          case 'D': printf("above average");
                        break;
          case 'E': printf("Satisfactory");
                        break;
        }
        printf("percentage is %f & grade is %c",per,grade);
        return 0;
        }
```

Test Report:

Viva Questions

1. **What is the MAIN benefit of designing tests early in the life cycle?**
   It helps prevent defects from being introduced into the code.
2. **What is the purpose of exit criteria?**
   The purpose of exit Criteria is to define when a test level is completed.
3. **What determines the level of risk?**
   The likelihood of an adverse event and the impact of the event determine the level of risk.
4. **When is used Decision table testing?**
   Decision table testing is used for testing systems for which the specification takes the form of rules or cause-effect combinations. In a decision table the inputs are listed in a column, with the outputs in the same column but below the inputs. The remainder of the table explores combinations of inputs to define the outputs produced.
5. **What is beta testing?**
   Testing performed by potential customers at their own locations.
6. **We use the output of the requirement analysis, the requirement specification as the input for writing …**
   User Acceptance Test Cases
7. **What are the benefits of Independent Testing?**
   Independent testers are unbiased and identify different defects at the same time.
8. **In a REACTIVE approach to testing when would you expect the bulk of the test design work to be begun?**
   The bulk of the test design work begun after the software or system has been produced.
9. **What are the phases of a formal review**
   In contrast to informal reviews, formal reviews follow a formal process. A typical formal review process consists of six main steps: Planning, Kick-off, Preparation, Review meeting, Rework,Follow-up.
10. **What is an equivalence partition (also known as an equivalence class)?**

An input or output ranges of values such that only one value in the range becomes a test case.

11. What is negative and positive testing?

A negative test is when you put in an invalid input and receives errors. While a positive testing, is when you put in a valid input and expect some action to be completed in accordance with the specification.

12. What is the purpose of a test completion criterion?

The purpose of test completion criterion is to determine when to stop tes

13. What is the difference between re-testing and regression testing?

Re-testing ensures the original fault has been removed; regression testing looks for unexpected side effects.

14. Could reviews or inspections be considered part of testing?

Yes, because both help detect faults and improve quality.

15. How much testing is enough?"

The answer depends on the risk for your industry, contract and special requirements.

16. Which of the following is the main purpose of the integration strategy for integration testing in the small?

The main purpose of the integration strategy is to specify which modules to combine when and how many at once.

17. 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). Wherever we can count things and can tell whether or not each of those things has been tested by some test, then we can measure coverage.

18. What is called the process starting with the terminal modules?

Bottom-up integration

References:

1. Paul C. Jorgensen: Software Testing, A Craftsman's Approach,3rd Edition, Auerbach Publications,2008
2. Mauro Pezze, Michal Young :Software Testing and Analysis- Process, Principles and Techniques, Wiley India,2008
3. https://www.guru99.com/software-testing-interview-questions.html