SOFTWARE TESTING

(18ISL66)

NOTE:

- COMMANDS FOR CREATE AND EXECUTE THE PROGRAMS IN UBUNTU
- TO CREATE : gedit program_name.c
- TO EXECUTE: gcc program_name.c
- TO GET OUT PUT: ./a.out
- THESE PROGRAMS WORKS ONLY ON UBUNTU

PROGRAM 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 anal sis, execute the test cases and discuss the results.

```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>


int main()

{

int a,b,c;

printf("\nenter three sides of the triangle\n");

scanf("%d%d%d",&a,&b,&c);

if((a>10)||(b>10)||(c>10))

{

printf("out of range");

exit(0);

}

if((a<b+c)&&(b<a+c)&&((c<a+b)))

{

if((a==b) && (b==c))

{

printf("equilateral triangle\n");

}

else if((a!=b)&&(a!=c)&&(b!=c))

{
```

```
printf("scalene triangle\n");
}
else
{
printf("iscoscles triangle\n");
}
}
else
{
printf("triangle cannot be formed\n");
}
return 0;
}
```

OUT PUT

Case ID	Description	INPUT DATA			Expected Output	Actual Output
		a	b	c		
1	Not satisfying the condition (a < (b+c)) & (b < (a+c)) & (c < (a+b))	2	1	9	Not a triangle	Not a triangle
2	Not satisfying the condition (a < (b+c)	4	1	2	Not a triangle	Not a triangle
3	If (a=b) & (b=c) (c=a)	5	5	5	Equilateral triangle	Equilateral triangle
4	If (a != b) & (a != c)(b != c)	5	2	1	Scalene triangle	Scalene triangle
5	If only two sides are equal a=b, b=a	5	5	9	Isosceles triangle	Isosceles triangle
6	Above the upper limit (a > 10)	12	5	3	Out of range	Out of range
7	Above the upper limit (b > 10)	10	12	9	Out of range	Out of range
8	Above the upper limit (c > 10)	10	9	12	Out of range	Out of range
9	Above the upper limit (a, b, c)	12	12	12	Out of range	Out of range

PROGRAM 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.

```
#include<stdio.h>

#include <stdlib.h>


int main()

{

int locks, stocks, barrels, t_sales, flag=0;

float Commission;

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 no of barrels");

scanf("%d",&barrels);

if((barrels<=0)||(barrels>90))

{

flag = 1;

}
if(flag == 1)
{
printf("invalid  input");
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\n",t_sales, Commission);
return 1;
}
```

OUTPUT

CASE	LOCKS	STOCKS	BARRELS	SALES	COMMISSION	COMMENT
1	1	1	1	100	10	Output minimum
2	1	1	2	125	12.5	Output min +
3	1	2	1	130	13	Output min +
4	2	1	1	145	14.5	Output min +
5	5	5	5	500	50	Mid point
6	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	10	10	10	1000	100	Border point +
10	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	14	14	14	1400	160	Mid point
14	18	18	17	1775	216.25	Border point -
15	18	17	18	1770	215.0	Border point -
16	17	18	18	1775	213.5	Border point -
17	18	18	18	1800	220	Border point
18	18	18	19	1825	225	Border point +
19	18	19	18	1830	226	Border point +
20	19	18	18	1845	229	Border point +
21	48	48	48	4800	820	Mid point
22	70	80	89	7775	1415	Output maximum
23	70	79	90	7770	1414	Output max -
24	69	80	90	7755	1411	Output max -
25	70	80	90	7800	1420	Output max

PROGRAM 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.

```
#include <stdio.h>
int main ()
{
int months[12]={31,28,31,30,31,30,31,31,30,31,30,31};
int d,m,y,nd,nm,ny,ndays;
printf("enter the date,month,year");
scanf("%d%d%d",&d,&m,&y);
ndays = months[m-1];
if (y<=18,12 && y>2012)
{
printf("invalid input year\n");
}
if(d<=0||d>ndays)
{
printf("invalid input days\n");
{
if(m<1&&m>12)
{
printf("invalid input month\n");
}
if(m==2)
{
if(y%100==0)
{
```

```
if(y%400==0)
{
ndays=22;
}
else if (y%4==0)
{
ndays=29;
}
nd=d+1;
nm=m;
ny=y;
if (nd>ndays)
{
nd=1;
nm+1;
}
if(nm>12)
{
nm=1;
ny++;
}
printf("\n Given date is %d%d%d",d,m,y);
printf("\n given date is %d%d%d",nd,nm,ny);
}
}
}
}
}
```

OUT PUT

Considering data program, we have three variables day, month & year

VARIABLES	MIN	MIN +	NON	MAX -	MAX
Day	1	2	15	30	31
Month	1	2	6	11	12
Year	1812	1813	1914	2014	2015

Test cases for date program using Boundary value analysis

TEST CASES	DESCRIPTION	INPUTS			OUTPUTS	COMMENTS
		DD	MM	YYYY		
BVA1	Enter values for day (nom), month(nom) & year (min)	15	6	1812	16/6/1812	Valid
BVA2	Enter values for day (nom), month(nom) & year (min +)	15	6	1813	16/6/1813	Valid
BVA3	Enter values for day (nom), month(min) & year (nom)	15	6	1914	16/6/1914	Valid
BVA4	Enter values for day (nom), month(min) & year (max+)	15	6	2014	16/6/2014	Valid
BVA5	Enter values for day (nom), month(nom) & year (max)	15	6	2015	16/6/2015	Valid
BVA6	Enter values for day (nom), month(min) & year (nom)	15	1	1914	16/1/1914	Valid
BVA7	Enter values for day (nom), month(min+) & year (nom)	15	2	1914	16/2/1914	Valid
BVA8	Enter values for day (nom), month(max-) & year (nom)	15	11	1914	16/11/1914	Valid
BVA9	Enter values for day (nom), month(max) & year (nom)	15	12	1914	16/12/1914	Valid
BVA10	Enter values for day (min), month(nom) & year (nom)	1	6	1914	2/6/1914	Valid
BVA11	Enter values for day (min+), month(nom) & year (nom)	2	6	1914	3/7/1914	Valid
BVA12	Enter values for day (max-), month(nom) & year (nom)	30	6	1914	1/7/1914	Valid
BVA13	Enter values for day (max), month(nom) & year (nom)	31	6	1914	Day out of range of the month	Valid

PROGRAM 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.

```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

int main()

{

int a, b, c;

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");

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("isoscales triangle");
}
}
else
{
printf("triangle cannot be found");
}
return 0;
}
```

OUTPUT

TEST REPORT :-

Weak equivalence class testing

CASE ID	DESCRIPTION	INPUT DATA			EXPEXTED OUTPUT	ACTUAL OUTPUT
		A	B	C		
1	Enter min values for a, b, c	5	5	5	Should display message equilateral triangle	Equilateral triangle
2	a, b, c values	2	2	3	Should display message isosceles triangle	Isosceles triangle
3	a, b, c values	3	4	5	Should display message scalene triangle	Scalene triangle
4	a, b, c values	4	1	2	Should display message not a triangle	Not a triangle

Weak robust equivalence class testing

CASE ID	DESCRIPTION	INPUT			OUTPUT
		A	B	C	
5	a, b, c values	-1	5	5	Triangle cannot be formed
6	a, b, c values	5	-1	5	Triangle cannot be formed
7	a, b, c values	11	5	5	Triangle cannot be formed
8	a, b, c values	5	11	5	Triangle cannot be formed
9	a, b, c values	5	11	5	Triangle cannot be formed

Strong robust equivalence class testing

CASE ID	DESCRIPTION	INPUT			OUTPUT
		A	B	C	
11	Enter one invalid & two valid input a, b, c	-1	5	5	Triangle cannot be formed
12	a, b, c values	5	-1	5	Triangle cannot be formed
13	a, b, c values	-1	-1	5	Triangle cannot be formed
14	a, b, c values	-1	-1	-1	Triangle cannot be formed

PROGRAM 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.

```
#include<stdio.h>

#include <stdlib.h>

int main()

{

int locks,stocks,barrels,t_sales,flag=0;

float commission;

printf("enter 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 barrels");

scanf("%d",&barrels);

if((barrels<=0) || (barrels>90))

{

flag=1;

}
if(flag==1)

{

printf("invalid  input");

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-1800));

}

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);

return 1;

}
```

OUTPUT

EQUIVALENC CLASS TESTING

Weak robust

CASE ID	LOCKS	STOCKS	BARRELS	EXPECTED OUTPUT
1	10	10	10	100
2	-1	40	45	Not in range
3	71	40	45	Not in range
4	35	81	45	Not in range
5	35	40	45	Not in range

Strong robust

CASE ID	LOCKS	STOCKS	BARRELS	EXPECTED OUTPUT
1	-2	-10	45	Not in range
2	35	-1	45	Not in range
3	-2	-1	45	Locks & stocks not in range
4	35	-1	-1	Stocks & barrels not in range
5	-2	-1	-1	Stocks, locks & barrels are not in range

PROGRAM 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.

```
#include<stdio.h>

#include <stdlib.h>


int main( )

{

int month[12]={31,28,31,30,31,30,31,31,30,31,30,31};

int d,m,y,nd,nm,ny,ndays;


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);

}
```

PROGRAM 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.

```
#include<stdio.h>

#include<ctype.h>



int main()

{

int a, b, c;


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");
}

return 0;
}
```

PROGRAM 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.

```
#include<stdio.h>
#include <stdlib.h>
#include<curses.h>
int main()
{
int locks, stocks, barrels, t_sales, flag = 0;
float commission;

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");

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);

return 1;
}
```

PROGRAM 9

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

```
#include<stdio.h>
#include <stdlib.h>
#include<curses.h>
int main()
{
int locks, stocks, barrels, t_sales, flag = 0;
float commission;

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");


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);


return 1;

}
```

PROGRAM 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.

```
#include<stdio.h>

#include <curses.h>

int main()

{

int a[20],n,low,high,mid,key,i,flag=0;


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;
}
```

PROGRAM 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.

```
#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;

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; \n");
for(i=0;i<n;i++)
printf("%d \n",arr[i]);
```

```
return 0;

}
```

PROGRAM 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

```
#include<stdio.h>

int 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+");

}
```