**Triangle Problem – 28/02/2023 (LAB EXERCISE 1)**

**Problem Statement** – in notebook

**Code** –

#include<stdio.h>

int main()

{

int a, b, c, c1, c2, c3;

char istriangle;

do

{

printf("\nEnter 3 integers which are sides of the triangle :\n"); 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("\nThe value of a=%d is not in the range of permitted values\n");

if(!c2)

printf("\nThe value of b=%d is not in the range of permitted values\n");

if(!c3)

printf("\nThe value of c=%d is not in the range of permitted values\n");

}

while

(!(c1 && c2 && c3));

//to check if it is a trianlge or not

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

istriangle='y';

else

istriangle='n';

if(istriangle=='y')

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

printf("It is an Equilateral Triangle\n");

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

printf("It is a Scalene Triangle");

else

printf("It is an Isosceles Triangle");

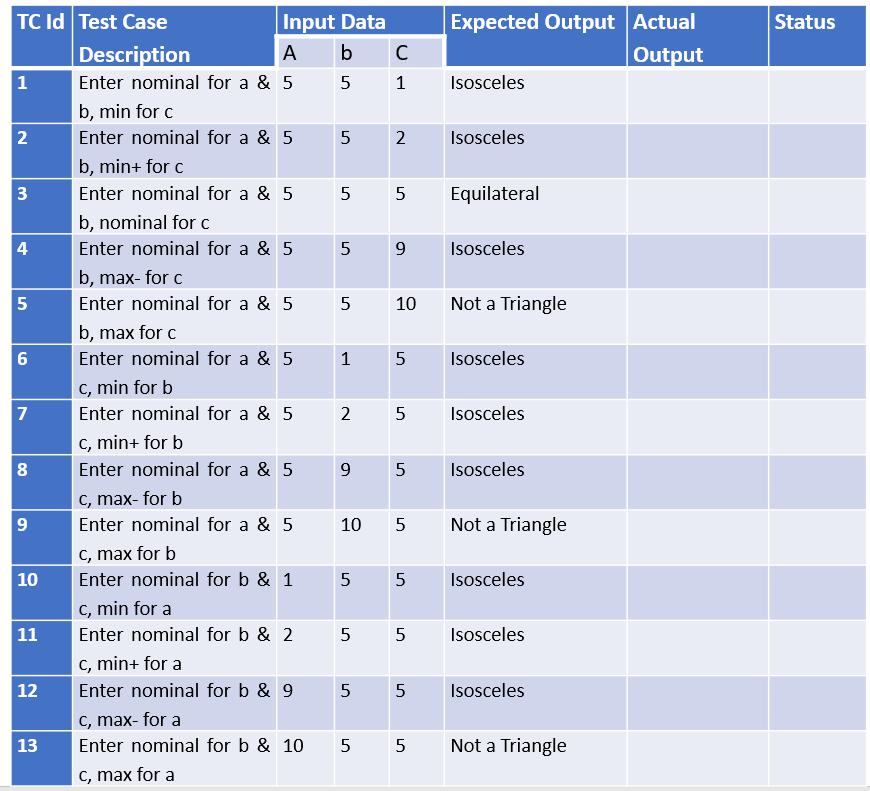
else

printf("It is Not A Triangle");

return 0;

}

**Boundary Value Analysis Test Cases** –



**Test Report –** in notebook

**Triangle Problem – 07/03/2023 (LAB EXERCISE 2)**

**Problem Statement** – in notebook

**Code** –

#include<stdio.h>

int main()

{

int a, b, c, c1, c2, c3;

char istriangle;

do

{

printf("\nEnter 3 integers which are sides of the triangle :\n"); 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("\nThe value of a=%d is not in the range of permitted values\n");

if(!c2)

printf("\nThe value of b=%d is not in the range of permitted values\n");

if(!c3)

printf("\nThe value of c=%d is not in the range of permitted values\n");

}

while

(!(c1 && c2 && c3));

//to check if it is a trianlge or not

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

istriangle='y';

else

istriangle='n';

if(istriangle=='y')

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

printf("It is an Equilateral Triangle\n");

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

printf("It is a Scalene Triangle");

else

printf("It is an Isosceles Triangle");

else

printf("It is Not A Triangle");

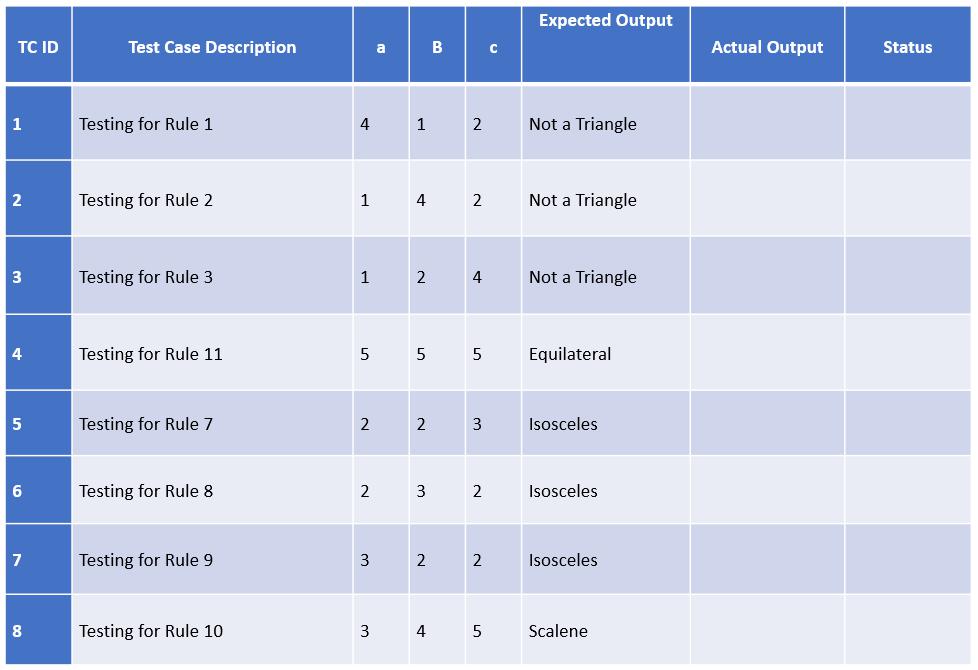
return 0;

}

**Decision Table** –



**Decision Table Test Cases** –



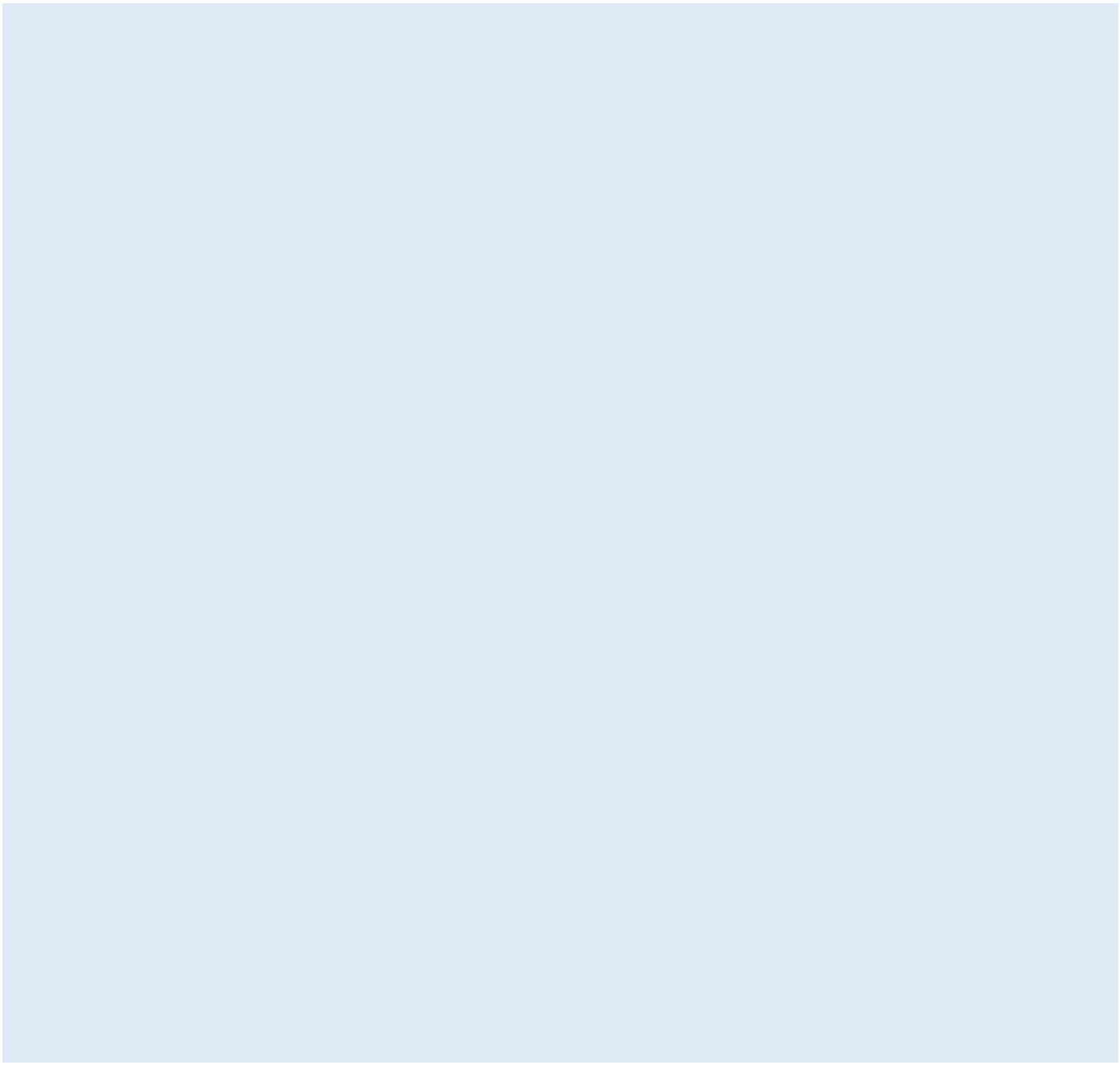
**Test Report –** in notebook

**Commission Problem – 14/03/2023 (LAB EXERCISE 3)**

**Problem Statement** – A rife salesperson in the former Arizona Territory sold rife locks, stocks, and barrels made by a gunsmith in Missouri. Locks cost $45, stocks cost $30, and barrels cost $25. The salesperson had to sell at least one complete rife per month and production limits were such that the most the salesperson could sell in a month was 70 locks, 80 stocks and 90 barrels. After each town visit, the salesperson sent a telegram to the Missouri gunsmith with the number of locks, stocks and barrels sold in that town. At the end of the month, the salesperson sent a very short telegram showing -1 locks sold. The gunsmith then knew the sales for the month were complete and computed the salesperson’s commission as follows: 10% on sales up to (and including) $1000, 15% on the next $800, and 20% on any sales in excess of $1800. The commission program produced a monthly sales report that gave the total number of locks, stocks, and barrels sold, the salesperson’s total dollar sales, and fnally, the commission.



|  |  |
| --- | --- |
| **AIM** | **To derive test cases using Boundary Value Analysis Technique** |
| **Program to Test** | Commission Problem |
| **Technique Used** | Boundary Value Analysis |
| **Type of Testing** | Black box testing |
| **Code** |  |
| #include <stdio.h> |  |
| int main() |  |
| { |  |



int locks, stocks, barrels, tlocks, tstocks, tbarrels;

foat lprice, sprice, bprice, sales, comm;

int c1, c2, c3, temp;

lprice = 45.0;

sprice = 30.0;

bprice = 25.0;

tlocks = 0.0;

tstocks = 0.0;

tbarrels = 0.0;

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("\nEnter the number of stocks & 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

{

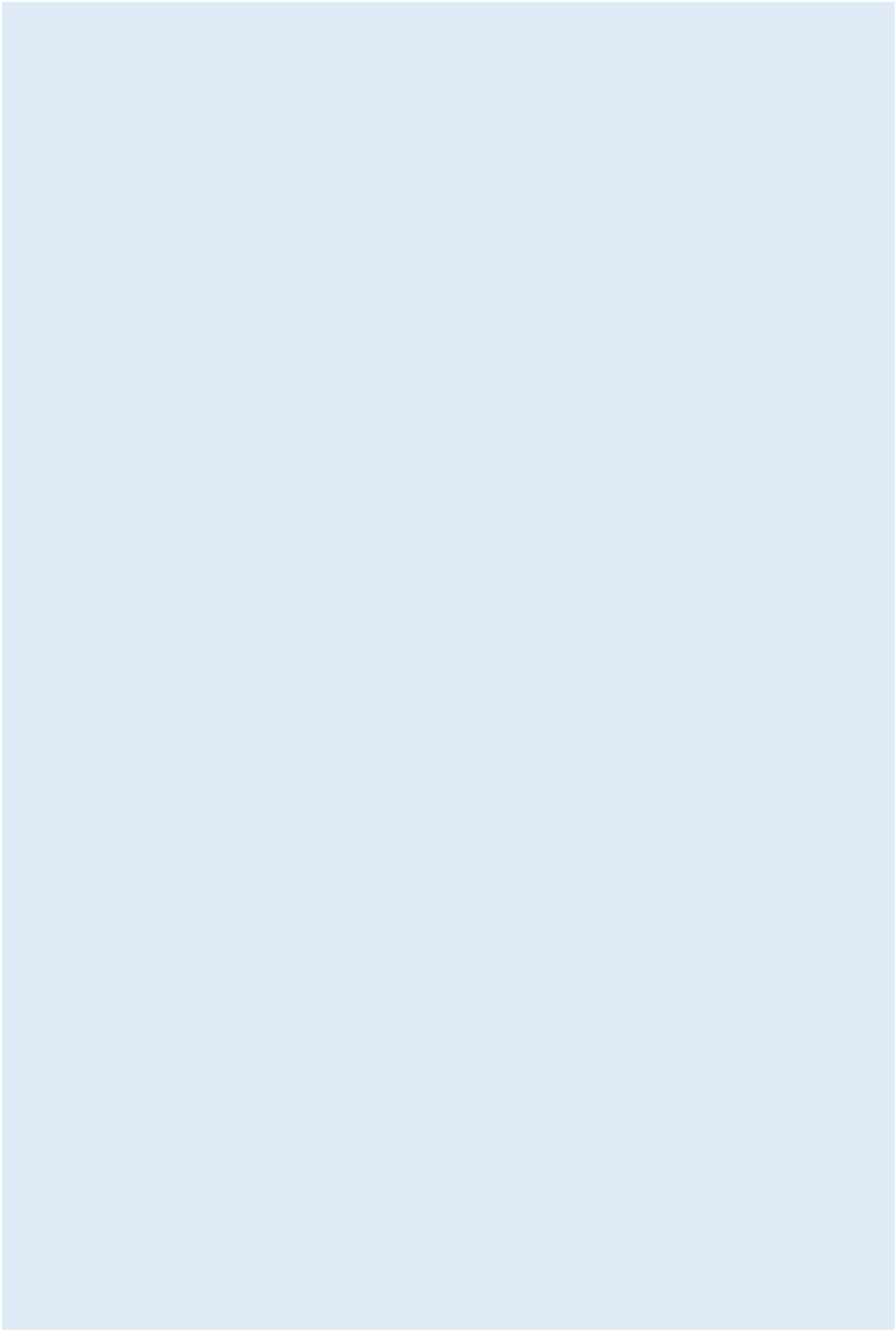
temp = tlocks + locks;

if(temp>70)

{

printf("New total locks = %d not in the range 1...70 sold", 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 sold", 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 sold", temp);

}

else

{

tbarrels=temp;

}

}

printf("Total barrels = %d\n", tbarrels);

printf("\nEnter 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 The total sales = %f\n", sales);

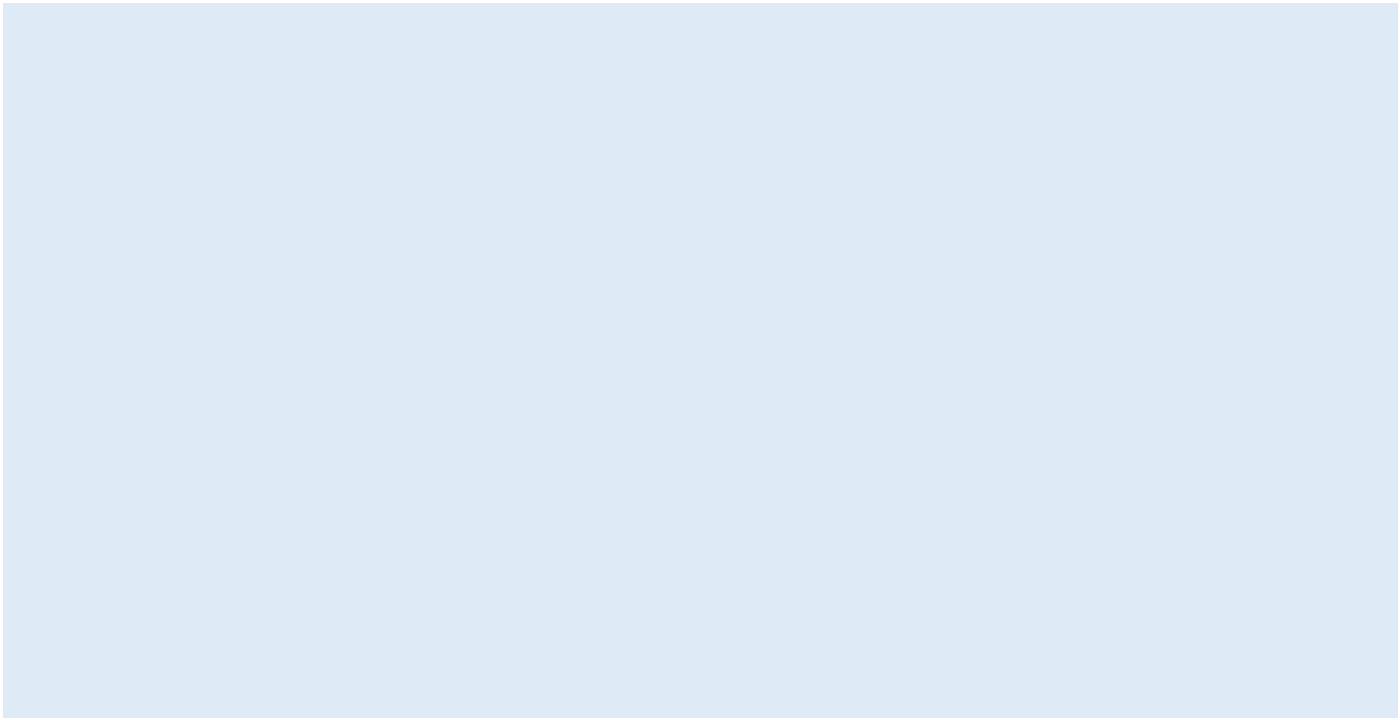
if (sales > 0)

{

if (sales > 1800)

{

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 \* sales;

}

printf("The commission is = %f\n", comm);

}

else

{

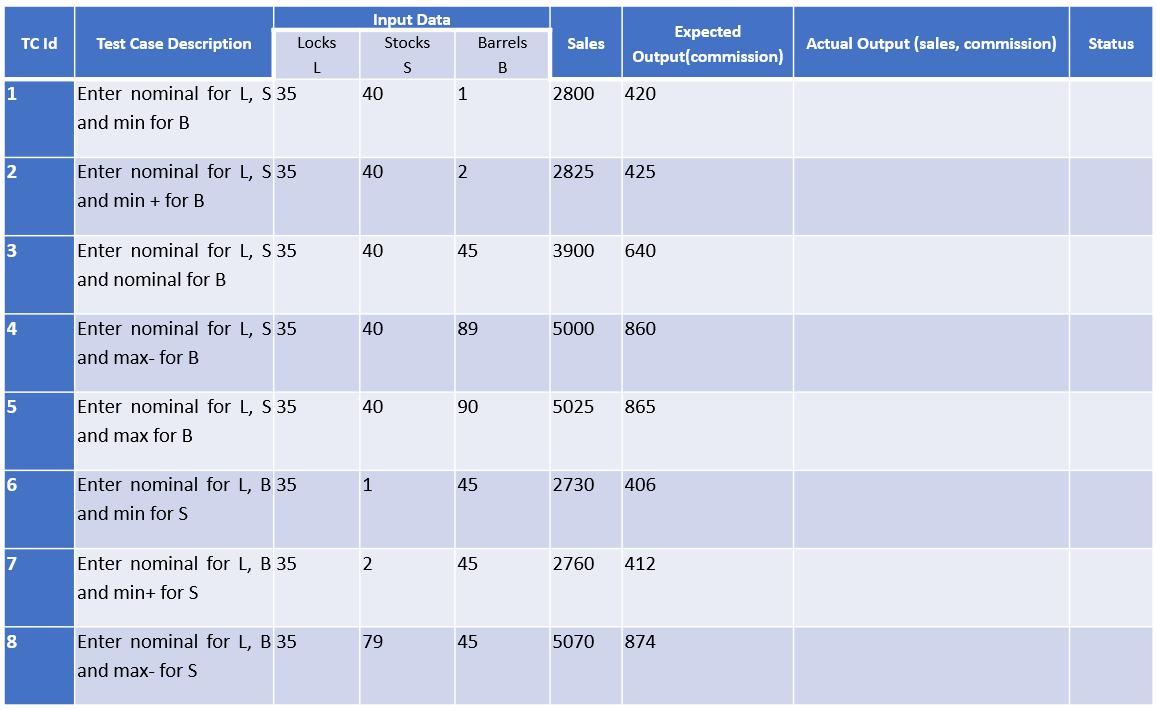
printf("There is no sales\n");

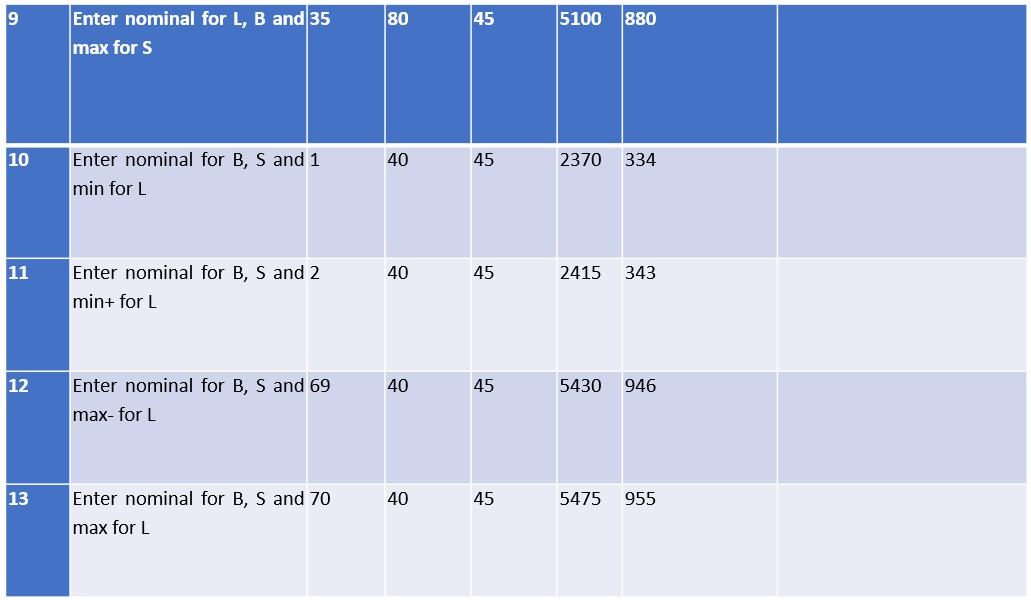
return 0;

}

}

**Boundary Value Analysis Test Cases** –





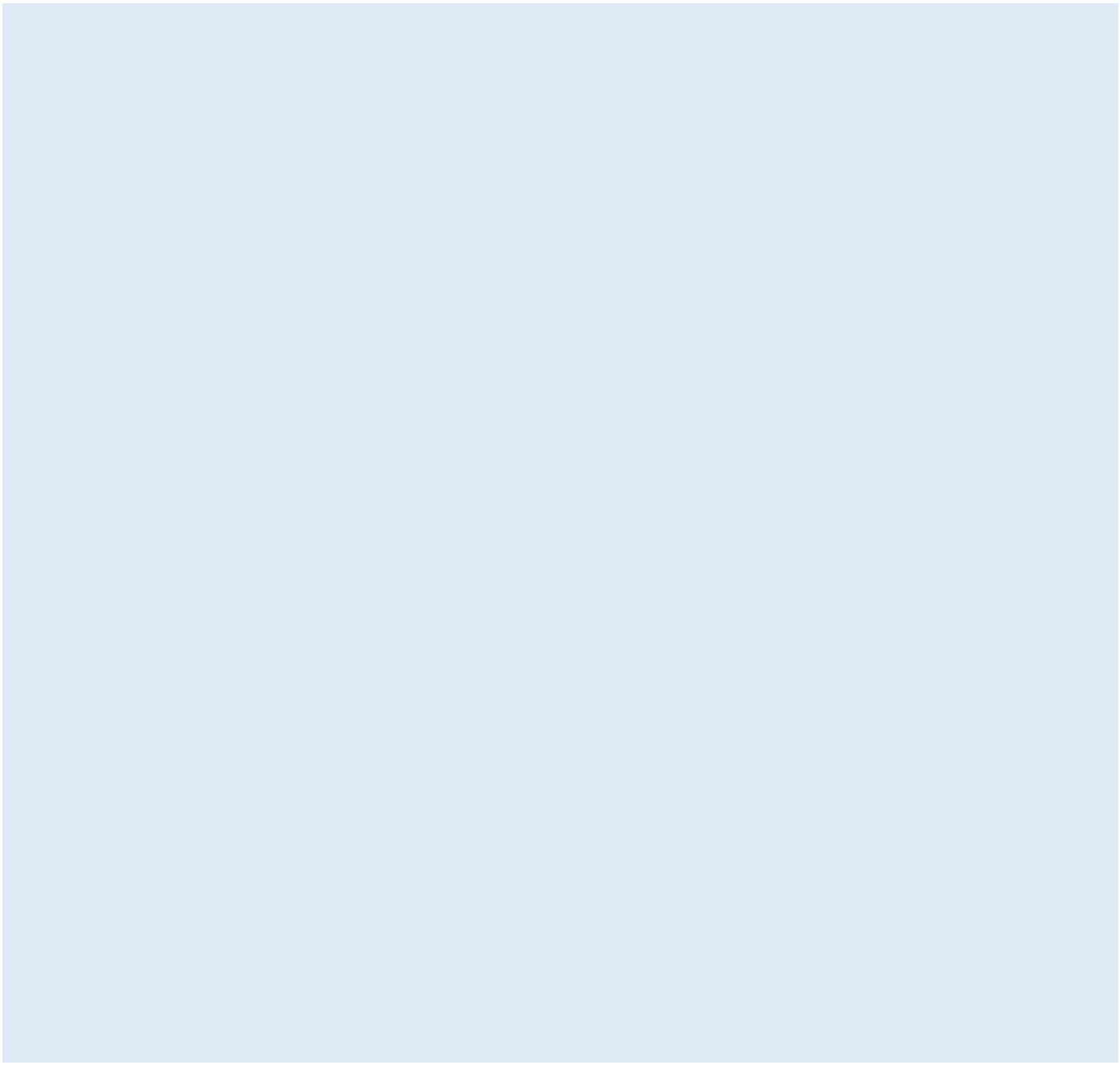
**Test Report –** in notebook

**Commission Problem – 21/03/2023 (LAB EXERCISE 4)**

**Problem Statement** – A rife salesperson in the former Arizona Territory sold rife locks, stocks, and barrels made by a gunsmith in Missouri. Locks cost $45, stocks cost $30, and barrels cost $25. The salesperson had to sell at least one complete rife per month and production limits were such that the most the salesperson could sell in a month was 70 locks, 80 stocks and 90 barrels. After each town visit, the salesperson sent a telegram to the Missouri gunsmith with the number of locks, stocks and barrels sold in that town. At the end of the month, the salesperson sent a very short telegram showing -1 locks sold. The gunsmith then knew the sales for the month were complete and computed the salesperson’s commission as follows: 10% on sales up to (and including) $1000, 15% on the next $800, and 20% on any sales in excess of $1800. The commission program produced a monthly sales report that gave the total number of locks, stocks, and barrels sold, the salesperson’s total dollar sales, and fnally, the commission.



|  |  |
| --- | --- |
| **AIM** | **To derive test cases using Boundary Value Analysis Technique** |
| **Program to Test** | Commission Problem |
| **Technique Used** | Decision Table |
| **Type of Testing** | Black box testing |
| **Code** |  |
| #include <stdio.h> |  |
| int main() |  |
| { |  |



int locks, stocks, barrels, tlocks, tstocks, tbarrels;

foat lprice, sprice, bprice, sales, comm;

int c1, c2, c3, temp;

lprice = 45.0;

sprice = 30.0;

bprice = 25.0;

tlocks = 0.0;

tstocks = 0.0;

tbarrels = 0.0;

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("\nEnter the number of stocks & 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

{

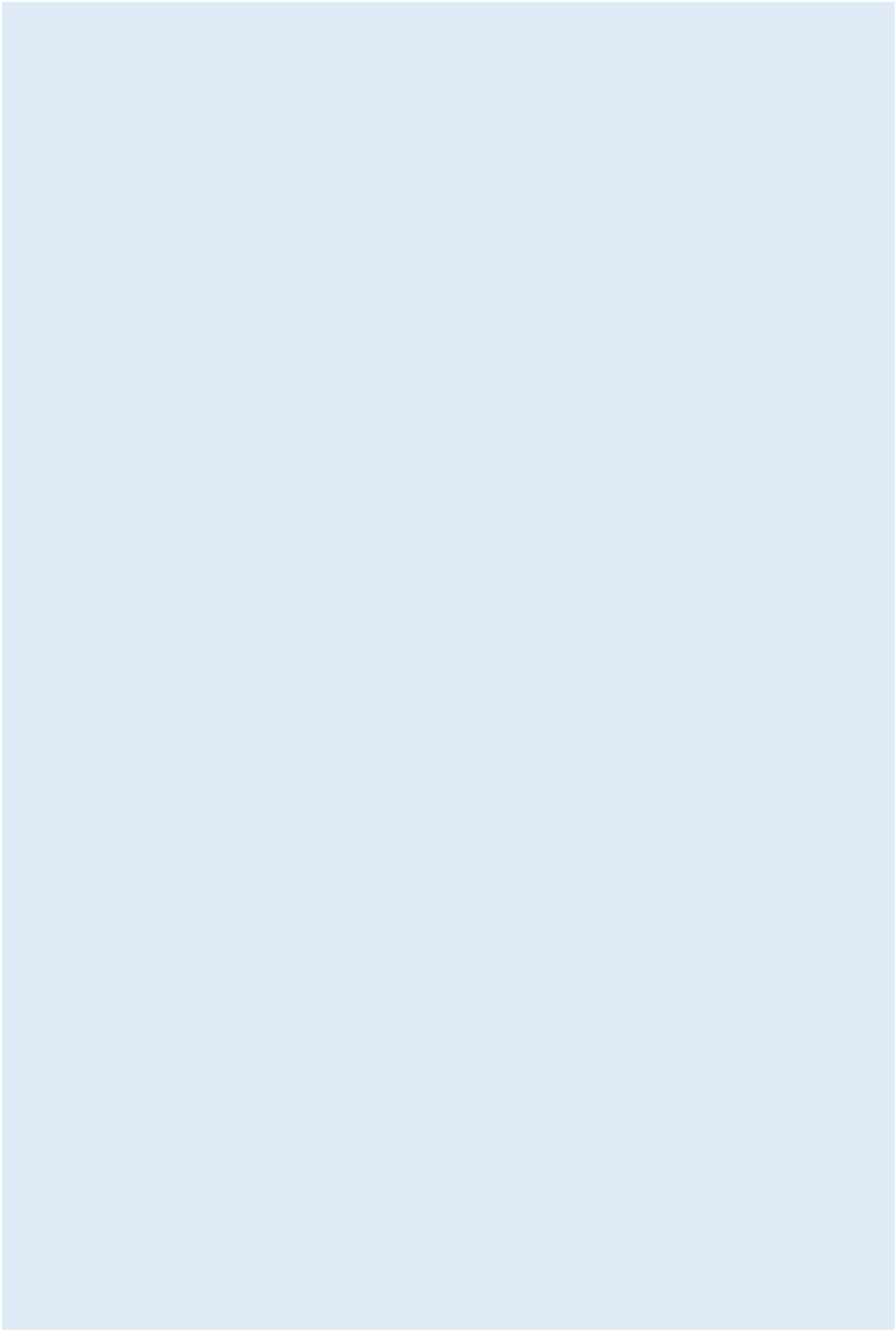
temp = tlocks + locks;

if(temp>70)

{

printf("New total locks = %d not in the range 1...70 sold", 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 sold", 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 sold", temp);

}

else

{

tbarrels=temp;

}

}

printf("Total barrels = %d\n", tbarrels);

printf("\nEnter 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 The total sales = %f\n", sales);

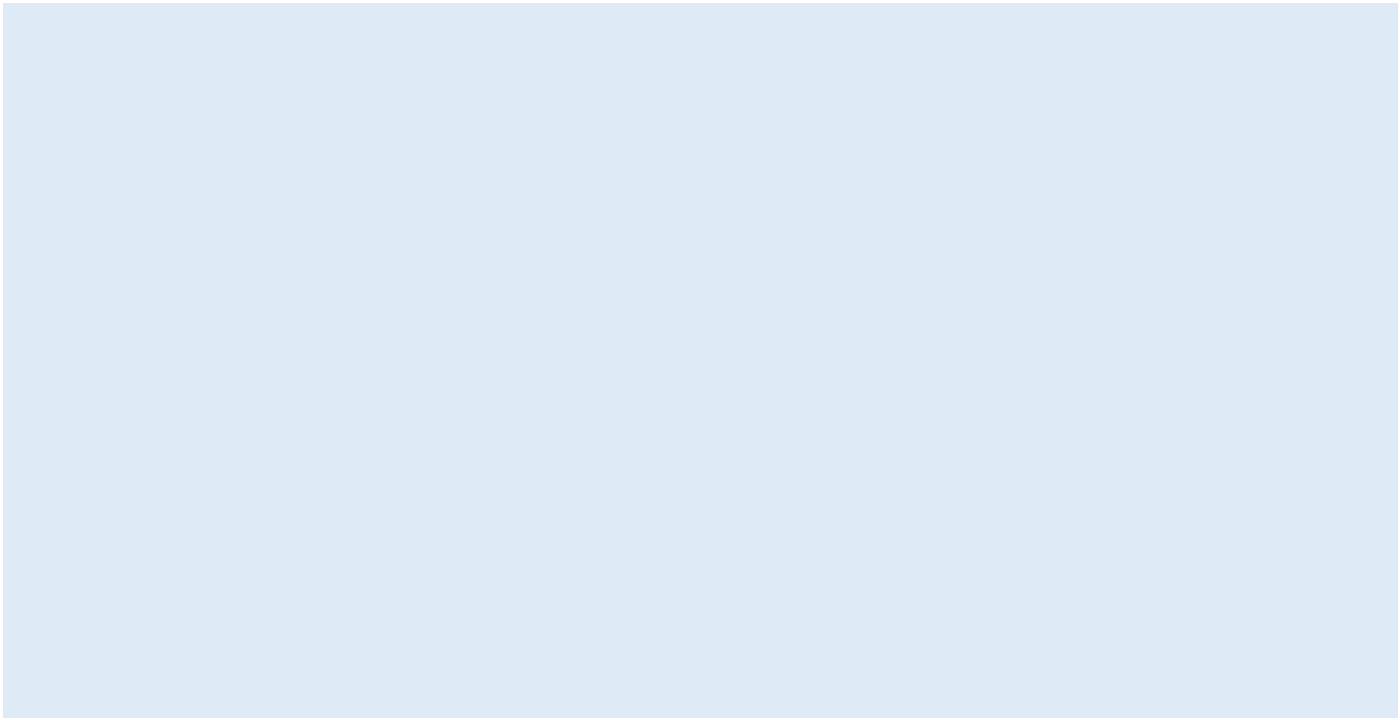
if (sales > 0)

{

if (sales > 1800)

{

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 \* sales;

}

printf("The commission is = %f\n", comm);

}

else

{

printf("There is no sales\n");

return 0;

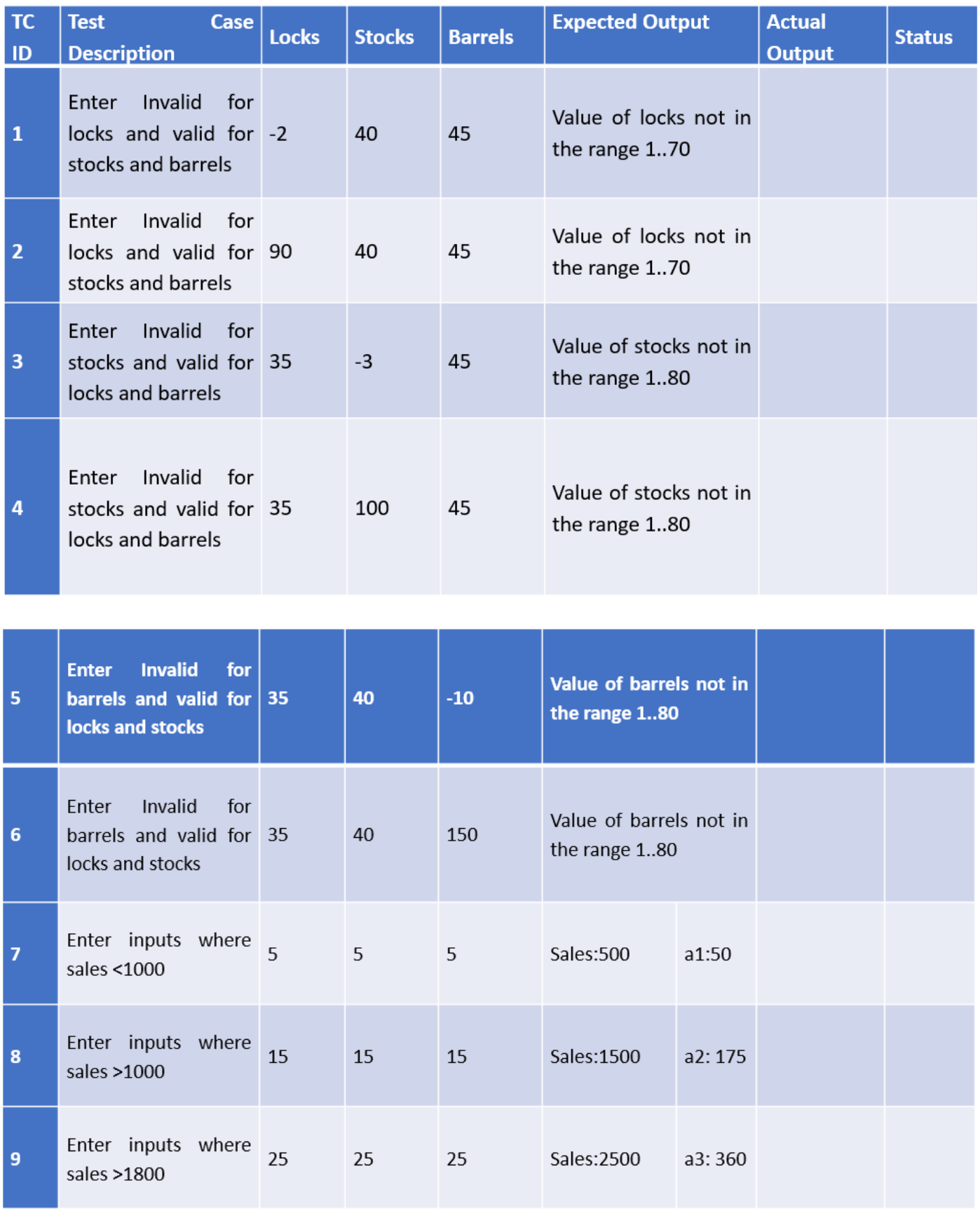
}

}

**Decision Table** –



**Decision Table Test Cases** –



**Test Report –** in notebook

**Next Date Problem – 28/03/2023 (LAB EXERCISE 5)**

**Problem Statement** – in notebook

**Code** –

#include<stdio.h>

#include<ctype.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%400==0 || (year%100==0 && year%4==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 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("\nValue of day not in the range 1...31\n");\

flag = 'n';

}

if(month<1 || month>12)

{

printf("\nValue of month not in the range 1...12\n");\

flag = 'n';

}

else if (check(day,month))

{

printf("\nValue of year not in the range day<=30\n");

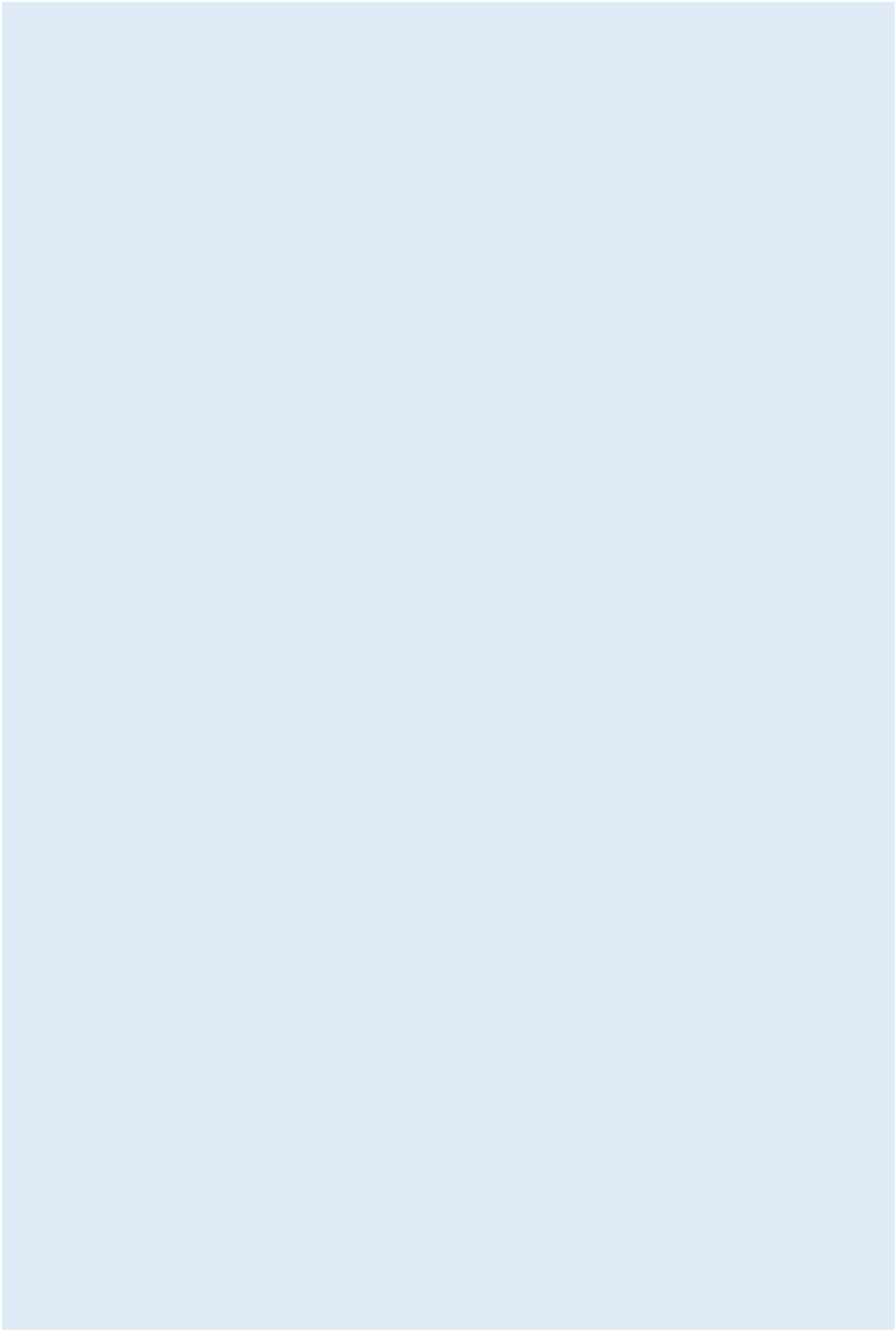
flag='n';

}

if(year<=1812 || year>2020)

{

printf("\nValue of year not in the range 1812...2020\n"); flag='n';

}

if(month==2)

{

if(isleap(year)&&day>29)

{

printf("\nInvalid date input for leap year");

flag='n';

}

else if(isleap(year)&&day>28)

{

printf("\nInvalid 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=month+1;

}

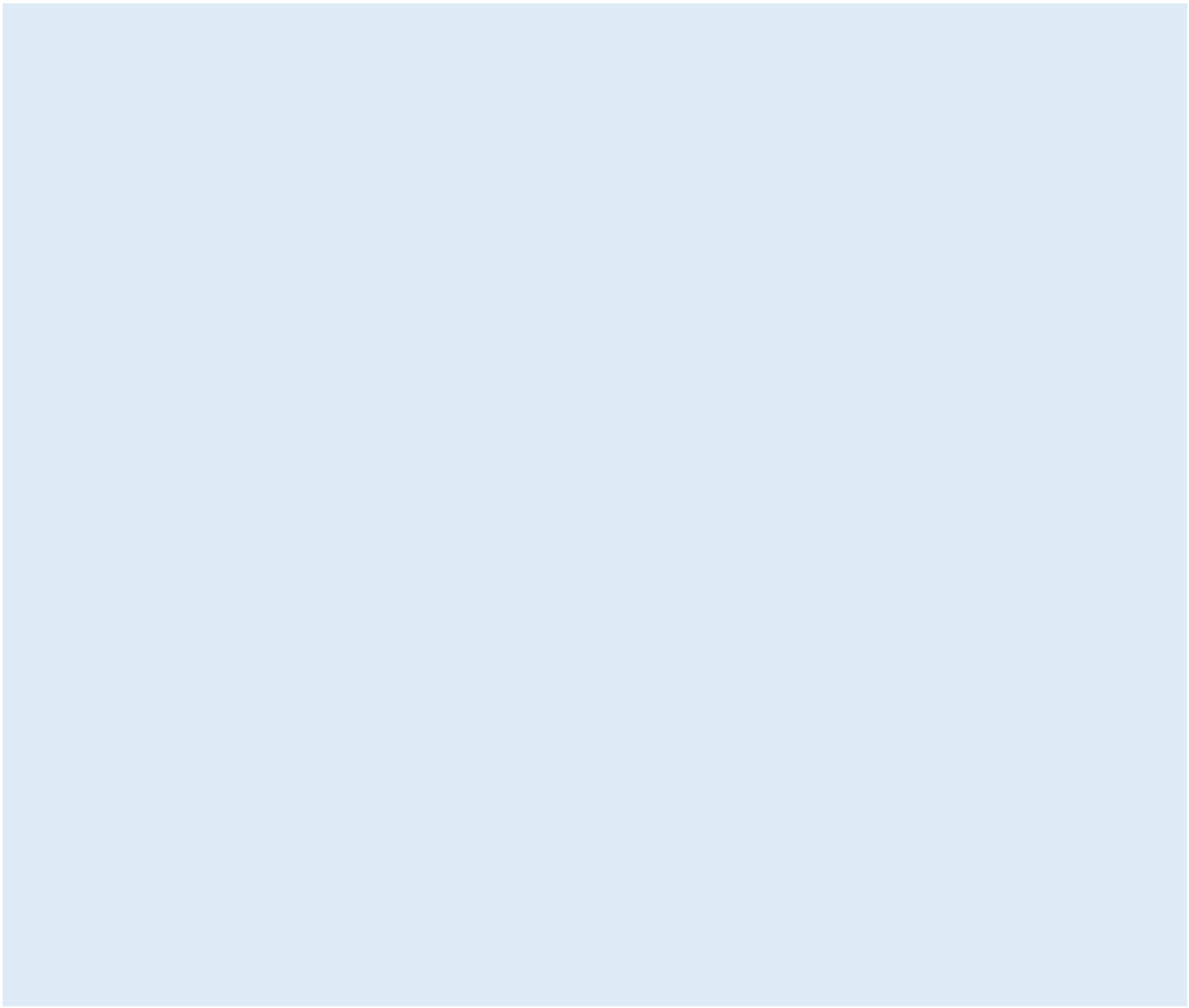
break;

case 12:

if(day<31)

{

tomm\_day=day+1;

}

else

{

tomm\_day=1;

tomm\_month=1;

if(year==2020)

{

printf("\nThe next day is out of boundary value of year\n"); return 0;

}

else

{

tomm\_year=year+1;

}

}

break;

case 2:

if(day<28)

{

tomm\_day=day+1;

}

else if(isleap(year) && day==28)

{

tomm\_day=1;

tomm\_month=3;

}

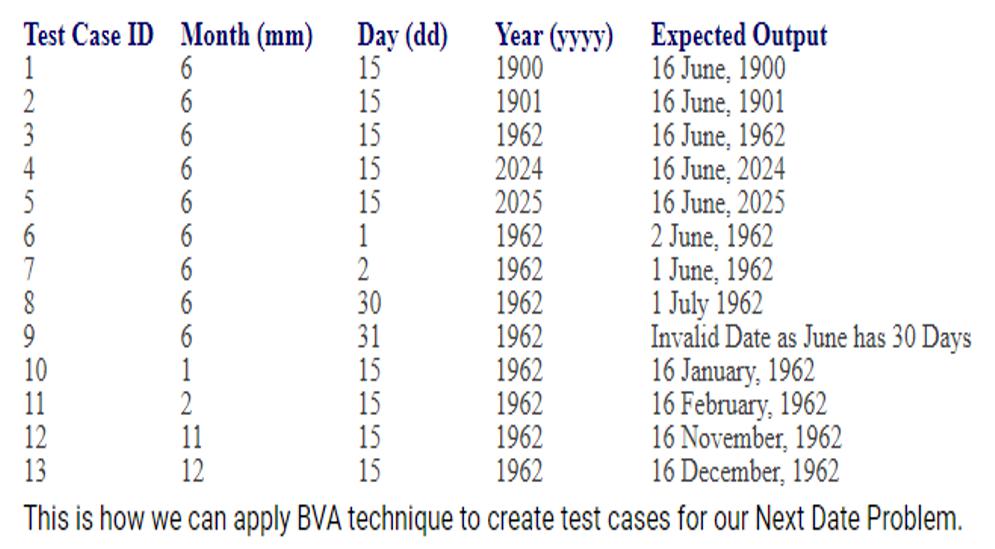
break;

}

printf("\nNext day is %d%d%d\n", tomm\_day, tomm\_month, tomm\_year); return 0;

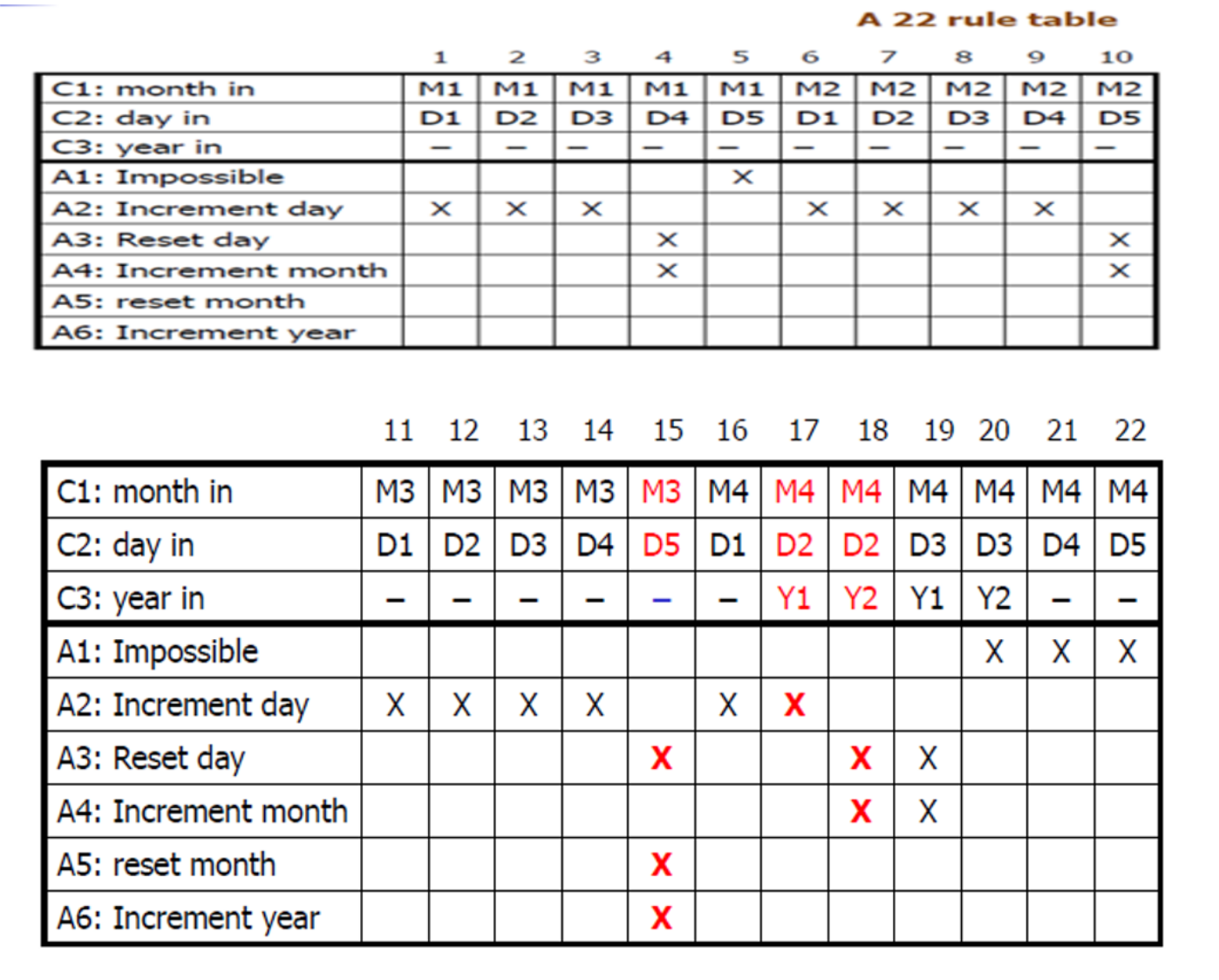
}

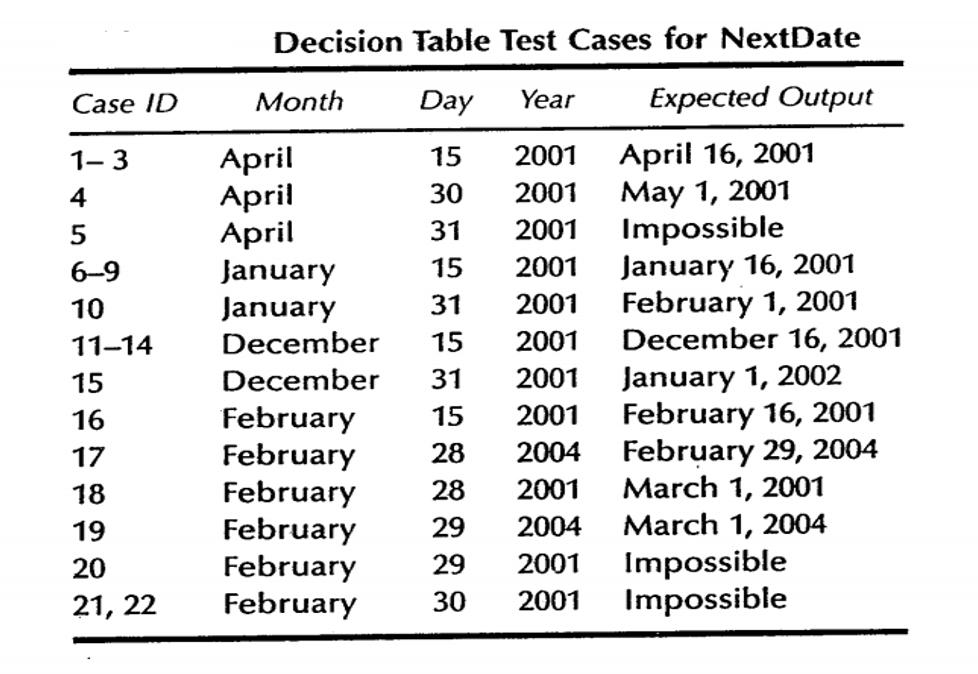
**Boundary Value Analysis Test Cases** –



**Test Report –** in notebook

**Decision Table**



**Decision Table**

**Binary Search Problem – 02/05/2023 (LAB EXERCISE 1)**

Design, develop, code and run the program in any programming language to implement the Binary Search Algorithm. Determine the basis path and using them to derive different test cases and execute these test cases and discuss the test result.



|  |  |
| --- | --- |
| **AIM** | **To derive test cases using Basis Path Testing Technique** |
| **Program to Test** | Binary Search |
| **Technique Used** | Basis Path Testing |
| **Type of Testing** | White Box Testing |

**Code** –

|  |
| --- |
| #include <stdio.h>  int main()  {  int a[20], key, i, n, succ;  printf("Enter the n value\n");  scanf("%d", &n);    if(n>0)  {  printf("Enter the elements in ascending order\n");    for(i=0; i<n; i++)  scanf("%d", &a[i]);  printf("Enter the key element to be searched\n");  scanf("%d", &key);  succ=binsrc(a, 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 elements should be greater than zero\n");  }  return 0;  }  int binsrc(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;  } |

**Program Graph** – in notebook

**Basis Path –**  in notebook

**Test Cases –** in notebook

**Test Report –** in notebook

**Letter Grading Procedure – 09/05/2023 (LAB EXERCISE 2)**

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



|  |  |
| --- | --- |
| **AIM** | **Write code and test cases, execute test cases** |
| **Program to Test** | Letter Grading Procedure |
| **Technique Used** | Basis Path Testing |
| **Type of Testing** | White Box Testing |

**Program Graph** – in notebook

**Basis Path –**  in notebook

**Code** –

|  |
| --- |
| #include<stdio.h>  void main()  {  int per;  printf("Enter percentage: \n");  scanf("%d",&per);  if(per<35 || per>=1)  {  printf("Fail\n");  }  else if(per<=40 && per>=35)  {  printf("You have secured Grade C\n");  }  else if(per<=50 && per>40)  {  printf("You have secured Grade C+\n");  }  else if(per<=60 && per>50)  {  printf("You have secured Grade B\n");  }  else if(per<=70 && per>60)  {  printf("You have secured Grade B+\n");  }  else if(per<=80 && per>70)  {  printf("You have secured Grade A\n");  }  else if(per<=100 && per>80)  {  printf("You have secured Grade A+\n");  }  else  {  printf("Invalid Input\n");  }  } |

**Test Cases –** in notebook

**Test Report –** in notebook

**Prime Number Checking Procedure – 16/05/2023 (LAB EXERCISE 3)**

Design, develop, code and run the program in any suitable language to implement a PRIME NUMBER CHECKING procedure, making suitable assumptions. Determine the basis path and using them derive different test cases, execute these test cases and discuss the test result.



|  |  |
| --- | --- |
| **AIM** | **Write code and test cases, execute test cases** |
| **Program to Test** | Prime Number Checking Procedure |
| **Technique Used** | Basis Path Testing |
| **Type of Testing** | White Box Testing |

**Program Graph** – in notebook

**Basis Path –**  in notebook

**Code** –

|  |
| --- |
| #include<stdio.h>  int main()  {  int n, i, flag=0;  printf("Enter a positive integer: ")  scanf("%d", &n);  //0 and 1 are not prime numbers  //change flag to 1 for non-prime number  if(n==0 | n==1)  flag=1;  for(i=2; i<=n/2; ++1)  {  //if n is divisible by i, then n is not prime  //change flag to 1 for non-prime number    if(n%i==0)  {  flag=1;  break;  }  }    if(flag==0)  printf("%d is a prime number.\n",n);  else  printf("%d is not a prime number.\n",n);  return 0;  } |

**Test Cases –** in notebook

**Test Report –** in notebook