Computer Science & Engineering

Branch: CSE 7TH SEM

Subject: Software Testing Lab

Subject Code: 22CS72

Program 1: Commission Problem **Testing Technique :Boundary Value Analysis**

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<conio.h>

int main()
{
intc1,c2,c3,temp;
int locks, stocks, barrels, totallocks ,totalstocks, totalbarrels;
float lockprice,stockprice,barrelprice,locksales,stocksales,barrelsales,sales,com;
lockprice=45.0;
stockprice=30.0;
barrelprice=25.0;
totallocks=0;
totalstocks=0;
totalbarrels=0;
clrscr();
printf("Enter the number of locks and to exit press -
1\n");scanf("%d",&locks);
while(locks!=-1)
{
```

```
c1=(locks<=0||locks>70);
printf("\nEnter the number of stocks and
barrels\n");scanf("%d %d",&stocks,&barrels);
c2=(stocks<=0||stocks>80);
c3=(barrels<=0||barrels>90);

if(c1)
    printf("\nValueoflocksarenotintherangeof1..... 70\n");
else
{
    temp=totallocks+locks;
     if(temp>70)
         printf("Newtotallocks=%dnotintherangeof1. .....70\n",temp);
     else
         totallocks=temp;
 }
printf("Total locks = %d",totallocks);
if(c2)
printf("\nValueofstocksnotintherangeof1.....80\n");
else
   {
     temp=totalstocks+stocks;
```

```
      if(temp>80)
         printf("\nNewtotalstocks=%dnotintherangeof1..... 80",temp);
      else
         totalstocks=temp;
    }
printf("\nTotal stocks = %d",totalstocks);
if(c3)
    printf("\nValueofbarrelsnotintherangeof1..... 90\n");
else
   {
         temp=totalbarrels+barrels;
         if(temp>90)
             printf("\nNewtotalbarrels=%dnotintherangeof1..... 90\n",temp);
         else
            totalbarrels=temp;
    }
printf("\nTotalbarrels=%d",totalbarrels);
printf("\nEnter the number of locks and to exit press -
1\n");scanf("%d",&locks);
}
printf("\n Total locks = %d",totallocks);prin
tf("\n Total stocks = %d",totalstocks);
printf("\n Total barrels = %d",totalbarrels);
```

```
locksales=totallocks*lockprice;
stocksales=totalstocks*stockprice;
barrelsales=totalbarrels*barrelprice;
sales=locksales+stocksales+barrelsales;prin
tf("\n Total sales = %f",sales);
if(sales>1800)
{
     com=0.10*1000;
     com=com+(0.15*800);
     com=com+0.20*(sales-1800);
 }
elseif(sales>1000)
{
    com=0.10*1000;
    com=com+0.15*(sales-1000);
}
else
    com=0.10*sales;
printf("\nCommission = %f",com);
getch();
return0;
}
```

Considering **Commission program**, we have three input variables **lock**, **stock** and **barrels**.

Range of value for **locks**=1-70,**stocks**=1-80and**barrels**=1-90

Variables	Min	Min+	Nom	Max-	Max
locks	1	2	35	69	70
stocks	1	2	40	79	80
barrels	1	2	45	89	90

Consideringoutputvariable**sales**wehave3slotsforcalculating**com**mission.i.e.,ifsalesarebelow1000,comis 10%,ifsalesare1001to1800thencomis15%andifsalesaregreaterthan1801,comis20%.

Sales	Min locks,stocks,barrrels	Min+ locks,stocks,barrrels	Nom locks,stocks,barrrels	Max- locks,stocks,barrrels	Max locks,stocks,barrrels
1-1000	1,1,1	2,1,1 1,2,1 1,1,2	5,5,5	9,10,10 10,9,10 10,10,9	10,10,10
1001-1800	11,10,10 10,11,10 10,10,11	12,10,10 10,12,10 10,10,12	14,14,14	17,18,18 18,17,18 18,18,17	18,18,18
1801-above	19,18,18 18,19,18 18,18,19	20,18,18 18,20,18 18,18,20	48,48,48	69,80,90 70,79,90 70,80,89	70,80,90

Testcases for commission program using INPUT Boundary value analysis

Test cases	Description	Inputs			Output		Comments
		Locks	Stocks	Barrels	Sales	Com	
BVA1	Enterthevaluesforlocks(nom),stocks(nom)andbarrels(min)	35	40	**1**	2800	420	Valid
BVA2	Enterthevaluesforlocks(nom),stocks(nom)andbarrels(min+)	35	40	**2**	2825	425	Valid
BVA3	Enterthevaluesforlocks(nom),stocks(nom)andbarrels(nom)	35	40	**45**	3900	640	Valid
BVA4	Enterthevaluesforlocks(nom),stocks(nom)andbarrels(max-)	35	40	**89**	5000	860	Valid
BVA5	Enterthevaluesforlocks(nom),stocks(nom)andbarrels(max)	35	40	**90**	5025	865	Valid
BVA6	Enterthevaluesforlocks(nom),stocks(min)andbarrels(nom)	35	**1**	45	2730	406	Valid
BVA7	Enterthevaluesforlocks(nom),stocks(min+)andbarrels(nom)	35	**2**	45	2760	412	Valid
BVA8	Enterthevaluesforlocks(nom),stocks(max-)andbarrels(nom)	35	**79**	45	5070	874	Valid
BVA9	Enterthevaluesforlocks(nom),stocks(max)andbarrels(nom)	35	**80**	45	5100	880	Valid
BVA10	Enterthevaluesforlocks(min),stocks(nom)andbarrels(nom)	**1**	40	45	2370	334	Valid
BVA11	Enterthevaluesforlocks(min+),stocks(nom)andbarrels(nom)	**2**	40	45	2415	343	Valid
BVA12	Enterthevaluesforlocks(max-),stocks(nom)andbarrels(nom)	**69**	40	45	5430	946	Valid
BVA13	Enterthevaluesforlocks(max),stocks(nom)andbarrels(nom)	**70**	40	45	5475	955	Valid

Testcases for commission program using OUTPUT Boundary value analysis

Test cases	Description	Inputs			Output		Comments
		Locks	Stocks	Barrels	Sales	Com	
BVA1	Enterthevaluesforlocks(min),stocks(min)and barrels(min)fortherange100to1000	1	1	1	100	10	Valid
BVA2	Enterthevaluesforlocks(min+),stocks(min)and barrels(min)fortherange100to1000	2	1	1	145	14.5	Valid
BVA3	Enter the values for locks(min), stocks(min+) and barrels(min)for the range 100 to 1000	1	2	1	130	13	Valid
BVA4	Enterthevaluesforlocks(min),stocks(min)and barrels(min+)fortherange100to1000	1	1	2	125	12.5	Valid
BVA5	Enterthevaluesforlocks(nom),stocks(nom)and barrels(nom)fortherange100to1000	5	5	5	500	50	Valid
BVA6	Enterthevaluesforlocks(max-),stocks(max)and barrels(max)fortherange100to1000	9	10	10	955	95.5	Valid
BVA7	Enterthevaluesforlocks(max),stocks(max-)and barrels(max)fortherange100to1000	10	9	10	970	97.0	Valid
BVA8	Enterthevaluesforlocks(max),stocks(max)and barrels(max-)fortherange100to1000	10	10	9	975	97.5	Valid
BVA9	Enterthevaluesforlocks(max),stocks(max)and barrels(max)fortherange100to1000	10	10	10	1000	100	Valid
BVA10	Enterthevaluesforlocks(min),stocks(min)and barrels(min)fortherange1000to1800	11	10	10	1045	106.75	Valid
BVA11	Enter the values for locks(min), stocks(min+) and barrels(min) for the range 1000 to 1800	10	11	10	1030	104.5	Valid
BVA12	Enterthevaluesforlocks(min),stocks(min)and barrels(min+)fortherange1000to1800	10	10	11	1025	103.75	Valid
BVA13	Enterthevaluesforlocks(min+),stocks(min)and barrels(min)fortherange1000to1800	12	10	10	1090	113.5	Valid
BVA14	Enterthevaluesforlocks(min),stocks(min+)and barrels(min)fortherange1000to1800	10	12	10	1060	109	Valid
BVA15	Enterthevaluesforlocks(min),stocks(min)and barrels(min+)fortherange1000to1800	10	10	12	1050	107.5	Valid

BVA16	1755	14	14	14	1400	160	Valid
BVA17	Enterthevaluesforlocks(max-),stocks(max)and barrels(max)fortherange1000to1800	17	18	18		213.25	Valid
BVA18	Enterthevaluesforlocks(max),stocks(max-)and barrels(max)fortherange1000to1800	18	17	18	1770	215.5	Valid
BVA19	Enterthevaluesforlocks(max),stocks(max)and barrels(max-)fortherange1000to1800	18	18	17	1775	216.25	Valid
BVA20	Enterthevaluesforlocks(max),stocks(max)and barrels(max)fortherange1000to1800	18	18	18	1800	220	Valid
BVA21	Enterthevaluesforlocks(min),stocks(min)and barrels(min)fortherange>1800	19	18	18	1845	229	Valid
BVA22	Enterthevaluesforlocks(min),stocks(min)and barrels(min)fortherange>1800	18	19	18	1830	226	Valid
BVA23	Enter the values for locks(min), stocks(min) and barrels(min) for the range > 1800	18	18	19	1825	225	Valid
BVA24	Enterthevaluesforlocks(min+),stocks(min)and barrels(min)fortherange>1800	20	18	18	1890	238	Valid
BVA25	Enterthevaluesforlocks(min),stocks(min+)and barrels(min)fortherange>1800	18	20	18	1860	232	Valid
BVA26	Enterthevaluesforlocks(min),stocks(min)and barrels(min+)fortherange>1800	18	18	20	1850	230	Valid
BVA27	Enterthevaluesforlocks(nom),stocks(nom)and barrels(nom)fortherange>1800	48	48	48	4800	820	Valid
BVA28	Enter the values for locks(max-), stocks(max) and barrels(max) for the range > 1800	69	80	90	7755	1411	Valid
BVA29	Enterthevaluesforlocks(max),stocks(max-)and barrels(max)fortherange>1800	70	79	90	7770	1414	Valid
BVA30	Enterthevaluesforlocks(max),stocks(max)and barrels(max-)fortherange>1800	70	80	89	7775	1415	Valid
BVA31	Enterthevaluesforlocks(max),stocks(max)and barrels(max)fortherange>1800	70	80	90	7800	1420	Valid

Program 2: Nextdate program **Testing Technique :Boundary Value Analysis**

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>
intcheck(intday,intmonth)
{
        if((month==4||month==6||month==9 ||month==11) && day==31)
                return 1;
        else
                return0;
}

intisleap(intyear)
{
        if((year%4==0 && year%100!=0) || year%400==0)
                return 1;
        else
                return0;

}
```

```
intmain()
{
        intday,month,year,tomm_day,tomm_month,tomm_year;
        char flag;
        do
        {
                flag='y';
                printf("\nenter the today's date in the form of dd mm yyyy\n");
                scanf("%d%d%d",&day,&month,&year);
                tomm_month=month;
                tomm_year= year;
                if(day<1||day>31)
                {
                        printf("value of day, not in the range
                         1...31\n");flag='n';
                }
                if(month<1||month>12)
                 {
                        printf("valueofmonth,notintherange1.....12\n");
                        flag='n';
                }

                elseif(check(day,month))
```

```
{
        printf("value of day, not in the range day<=30");
         flag='n';
}

if(year<=1812||year>2015)
{
           printf("valueofyear,notintherange1812. ..... 2015\n");
           flag='n';
        }
if(month==2)
{
        if(isleap(year)&&day>29)
         {
                printf("invalid date input for leap year");
                flag='n';
                }
        elseif(!(isleap(year))&&day>28)
        {
printf("invalid date input for not a leap year");
                flag='n';
         }
}
```

```
}while(flag=='n');

switch(month)
{
        case1:
        case3:
        case5:
        case7:
        case8:
        case10:if(day<31)
                        tomm_day=day+1;
                else
                  {
                        tomm_day=1;
                        tomm_month=month+1;
                  }

                break;
        case 4:
        case6:
        case9:
        case11:if(day<30)
                        tomm_day=day+1;
                 else
```

```
            {
                tomm_day=1;
                tomm_month=month+1;
            }
        break;

case12:if(day<31)
                tomm_day=day+1;
        else
          {
                tomm_day=1;
                tomm_month=1;
                if(year==2015)
                 {
                    printf("the next day is out of boundary value of year\n");
                    tomm_year=year+1;
                 }

                else
                   tomm_year=year+1;
          }
         break;
case2:
```

```
                    if(day<28)
                            tomm_day=day+1;
                      elseif(isleap(year)&&day==28)
                                tomm_day=day+1;
                     elseif(day==28||day==29)
                            {
                              tomm_day=1;
                              tomm_month=3;
                            }
                    break;
        }
        printf("nextdayis: %d%d%d",tomm_day,tomm_month,tomm_year); return
        0;
}
```

Considering Date program, we have three variables day, month and year.

Variables	Min	Min+	Nom	Max-	Max
day	1	2	15	30	31
month	1	2	6	11	12
year	1812	1813	1914	2014	2015

Testcases for Date program using Boundary Value Analysis

Test cases	Description	Inputs			Output	Comments
		DD	MM	YY		
BVA1	Enterthevaluesforday(nom),month(nom)andyear(min)	15	6	**1812**	16/6/1812	Valid
BVA2	Enterthevaluesforday(nom),month(nom)andyear(min+)	15	6	**1813**	16/6/1813	Valid
BVA3	Enterthevaluesforday(nom),month(min)andyear(nom)	15	6	**1914**	16/6/1914	Valid
BVA4	Enterthevaluesforday(nom),month(nom)andyear(max-)	15	6	**2014**	16/6/2014	Valid
BVA5	Enterthevaluesforday(nom),month(nom)andyear(max)	15	6	**2015**	16/6/2015	Valid
BVA6	Enterthevaluesforday(nom),month(min)andyear(nom)	15	**1**	1914	16/1/1914	Valid
BVA7	Enterthevaluesforday(nom),month(min+)andyear(nom)	15	**2**	1914	16/2/1914	Valid
BVA8	Enterthevaluesforday(nom),month(max-)andyear(nom)	15	**11**	1914	16/11/1914	Valid
BVA9	Enterthevaluesforday(nom),month(max)andyear(nom)	15	**12**	1914	16/12/1914	Valid
BVA10	Enterthevaluesforday(min),month(nom)andyear(nom)	**1**	6	1914	2/6/1914	Valid
BVA11	Enterthevaluesforday(min+),month(nom)andyear(nom)	**2**	6	1914	3/6/1914	Valid
BVA12	Enterthevaluesforday(max-),month(nom)andyear(nom)	**30**	6	1914	1/7/1914	Valid
BVA13	Enterthevaluesforday(max),month(nom)andyear(nom)	**31**	6	1914	Day out of range for the month	Valid

Program 3: Commission Problem using Decision Table

Problem Statement (Commission Problem)

The typical "commission problem" (as used in many software-testing courses) is:

A salesperson sells locks, stocks, and barrels in a month.

Prices:

- **Lock = $45 each**
- **Stock = $30 each**
- **Barrel = $25 each**

Sales constraints:

- **Must sell at least 1 of each (i.e. locks ≥ 1, stocks ≥ 1, barrels ≥ 1)**
- **Maximum possible: locks ≤ 70, stocks ≤ 80, barrels ≤ 90**

At end of month, the total sales value is computed = (locks × 45) + (stocks × 30) + (barrels × 25).

Commission is computed by a tiered structure:

1. **For the first $1000 of sales: 10% commission**
2. **For the next $800 (i.e. from $1000.01 to $1800): 15% commission**
3. **For any sales beyond $1800: 20% commission**

If any of the sales counts (locks, stocks, barrels) is outside its valid range (below 1 or above the max), then the sales are invalid and no commission should be awarded (or an error is flagged).

Thus, the program's job is: accept inputs (locks, stocks, barrels), validate ranges, compute total sales, and then compute commission per the tiered scheme, or reject invalid input.

This is the version many course materials adopt.

Decision Table Approach

Identify Conditions (Inputs) and Actions (Outputs)

We need to identify the key conditions (Boolean or discrete predicates) that affect commission calculation, and then map to actions.

Conditions (C):

Let us define:

Condition	Description
C1	locks is out of valid range (locks < 1 or locks > 70)
C2	stocks is out of valid range (stocks < 1 or stocks > 80)

C3	barrels are out of valid range (barrels < 1 or barrels > 90)
C4	locks is in valid range (1 ≤ locks ≤ 70)
C5	stocks is in valid range (1 ≤ stocks ≤ 80)
C6	barrels is in valid range (1 ≤ barrels ≤ 90)
C7	total sales ≤ 1000
C8	total sales > 1000 and ≤ 1800
C9	total sales > 1800

Note: C4, C5, C6 are complements of C1, C2, C3 respectively (i.e. if C1 is false then C4 is true, etc.). We may treat them as "valid ranges" conditions. The conditions C7−C9 are mutually exclusive tiers for total sales.

Actions (A):

Action	Description
A1	Report invalid locks input (i.e. reject)
A2	Report invalid stocks input
A3	Report invalid barrels input
A4	Compute commission via 10% (i.e. within first tier)
A5	Compute commission via mix of 10% + 15% (i.e. tier up to 1800)
A6	Compute commission via mix including 20% above 1800

A7 **Output computed commission**

We can combine A4/A5/A6 into a single "compute appropriate commission" action, but for clarity in the decision table we break tiers into actions.

Decision Table Construction

We list the possible "rules" (i.e. combinations of conditions) and assign actions. Many combinations are impossible or irrelevant; we focus on realistic ones.

One possible limited-entry decision table (simplified) might be:

Rule	C1 Actions	C2	C3	C7	C8	C9
R1	F A4, A7	F	F	T	F	F
R2	F A5, A7	F	F	F	T	F
R3	F A6, A7	F	F	F	F	T
R4	T A1 (reject)	–	–	–	–	–
R5	– A2 (reject)	T	–	–	–	–

R6	–	–	T	–	–	–
	A3 (reject)					

Explanation:

R1: All three inputs in valid ranges, total sales ≤ 1000 → compute via 10%.

R2: All valid, sales in (1000, 1800] → compute via next tier.

R3: All valid, sales > 1800 → compute via highest tier.

R4: locks invalid (out of range) → reject immediately.

R5: stocks invalid → reject.

R6: barrels invalid → reject.

We assume "–" means "don't care" (i.e. irrelevant) because once an input is invalid you don't compute commission.

We can further refine if needed (for example handling multiple invalid inputs simultaneously), but this covers core cases.

From this decision table, each rule becomes (or suggests) a test case.

Derive Test Cases

Test cases Table

Test Case	Locks	Stocks	Barrels	Total Sales	Expected Behavior / Commission
TC1 (R1)	5	5	5	5×45 + 5×30 + 5×25 = 225 + 150 + 125 =500	10% of 500 = 50.00
TC2 (R2)	15	15	15	15×45 + 15×30 + 15×25 = 675 + 450 + 375 = 1500	10% of 1000 + 15% of 500 = 175.00
TC3 (R3)	30	40	50	30×45 + 40×30 + 50×25 = 1350 + 1200 + 1250 = 3800	10% of 1000 + 15% of 800 + 20% of 2000 = 620.00
TC4 (R4)	0	10	10	Invalid	Reject – invalid locks
TC5 (R5)	10	0	10	Invalid	Reject – invalid stocks
TC6 (R6)	10	10	0	Invalid	Reject – invalid barrels
TC7	10	10	20	1045 + 1030 + 2025 = 450 + 300 + 500 = 1250	100 + 15% of 250 = 137.50
TC8	10	10	30	1045+1030+3025 = 450+300+750 = 1500	Same as TC2 → 175.00
TC9	20	20	20	2045+2030+2025 = 900 + 600 + 500 = 2000	100 + 120 + 20% of 200 = 260.00

From the rules, we choose test inputs (locks, stocks, barrels) that satisfy the condition pattern and compute expected commission or rejection.

C Program Implementation

Below is one possible C implementation of the commission problem:

```
#include <stdio.h>

double compute_commission(int locks, int stocks, int barrels, int *error_flag) {
   const int LOCK_PRICE = 45;
   const int STOCK_PRICE = 30;
   const int BARREL_PRICE = 25;

   // Validate inputs
   if (locks < 1 || locks > 70) {
      *error_flag = 1;  // locks invalid
      return 0.0;
   }
   if (stocks < 1 || stocks > 80) {
      *error_flag = 2;  // stocks invalid
```

```
    return 0.0;
  }
  if (barrels < 1 || barrels > 90) {
    *error_flag = 3;  // barrels invalid
    return 0.0;
  }

  // Compute total sales
  int total = locks * LOCK_PRICE + stocks * STOCK_PRICE + barrels * BARREL_PRICE;
  double commission = 0.0;

  if (total <= 1000) {
    commission = total * 0.10;
  } else if (total <= 1800) {
    // first 1000 at 10%, remainder at 15%
    commission = 1000 * 0.10 + (total - 1000) * 0.15;
  } else {
    // up to 1000: 10%, next 800: 15%, remainder: 20%
    commission = 1000 * 0.10 + 800 * 0.15 + (total - 1800) * 0.20;
  }

  *error_flag = 0;  // no error
```

```
    return commission;
}

int main() {
    int locks, stocks, barrels;
    printf("Enter number of locks, stocks, barrels: ");
    if (scanf("%d %d %d", &locks, &stocks, &barrels) != 3) {
        printf("Invalid input format.\n");
        return 1;
    }

    int err;
    double comm = compute_commission(locks, stocks, barrels, &err);

    if (err == 1) {
        printf("Error: locks out of range (1 to 70 allowed)\n");
    } else if (err == 2) {
        printf("Error: stocks out of range (1 to 80 allowed)\n");
    } else if (err == 3) {
        printf("Error: barrels out of range (1 to 90 allowed)\n");
    } else {
        printf("Total commission = %.2f\n", comm);
```

```
    }

    return 0;
}
```

Explanation / Comments:

compute_commission does validation first; if any input is invalid, it sets an error flag and returns 0.

Otherwise it computes the total sales value, then applies the tiered commission logic.

In main, after reading inputs, we call the function and then check the error flag to decide whether to print an error or the commission.

Executing the Test Cases & Discussion of Results

Let me simulate (by hand) the execution of our test cases and compare with expected.

Test Case	Input (locks, stocks, barrels)	Total Sales	Error Flag	Computed Commission	Expected	Pass/Fail	Comments

TC1 0.10 = 50.00	5, 5, 5 50.00	500 Pass	err = 0	500 *
TC2	15, 15, 15 10000.10 + 5000.15 = 100 + 75 = 175.00 175.00	1500 Pass	err = 0	
TC3 120 + (3800−1800)*0.20 = 100 + 120 + 400 = 620.00	30, 40, 50 620.00	3800 Pass	err = 0	100 +
TC4	0, 10, 10 reject	— Pass	err = 1	0
TC5	10, 0, 10 reject	— Pass	err = 2	0
TC6	10, 10, 0 reject	— Pass	err = 3	0
TC7	10, 10, 20 100 + 0.15*(250) = 100 + 37.5 = 137.50 Pass	computed 1250		err = 0 137.50
TC8 TC2 = 175.00	10, 10, 30 175.00	1500 Pass	err = 0	same as
TC9 120 + (2000−1800)*0.20 = 100 + 120 + 40 = 260.00	20, 20, 20 260.00	2000 Pass	err = 0	100 +

All test cases produce the expected result with this implementation, so they pass.

If during testing we had found discrepancies, we would inspect logic (e.g. boundary off-by-one errors) or error handling

omissions.

One might also include more test cases (e.g. maximal values, exactly 1000, exactly 1800, multiple invalid errors) to further validate.

Analysis of the Decision Table Testing Approach

Strengths

Systematic coverage: The decision table makes explicit which combinations of conditions must be considered (valid vs invalid inputs, and sales tiers).

Clarity: It's easy to see that invalid input conditions short-circuit commission computation.

Test derivation: Each rule gives a clear test scenario.

Non-redundancy: You avoid redundant test cases by collapsing "don't care" conditions appropriately.

Weaknesses / Caveats

If there are many conditions, the number of columns (rules) explodes—combinatorial explosion.

Some combinations are infeasible or meaningless (e.g. locks invalid & valid simultaneously), so care is needed to

prune.

It does not automatically give boundary tests (e.g. exactly 1000, exactly 1800) unless we explicitly add them.

It assumes decisions are independent; in practice there could be dependencies (e.g. input ranges affect sales tiers).

Enhancements / Additional Testing Techniques

In addition to decision-table testing, one should also apply:

Boundary value analysis: test at edges (locks = 1, locks = 70, just outside, etc.).

Equivalence partitioning: partition combinations of sales amounts into classes (≤1000, (1000,1800], >1800).

Negative / robustness testing: test invalid/negative values (e.g. locks = −1, stocks = 81, barrels = 100).

Path / control flow testing: ensure each branch in the program (e.g. the if (total ≤ 1000), etc.) is executed.

Summary & Suggestions

We have designed a clean decision table covering the major valid/invalid and tiered-sales cases.

We derived test cases directly from the table, implemented a C program, ran the test cases (simulated), and confirmed the results match expected.

The decision table approach helps ensure logical combinations are covered, but it should be complemented with boundary and equivalence testing to catch edge errors.

Program 4: Triangle Problem **Testing Technique: Boundary Value Analysis**

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.

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

int main( )
{
  int a,b,c,c1,c2,c3;
  do
  {
    printf("enter the sides of 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)SSSSSSS
      printf("value of a is out of range");
    if(!c2)
      printf("value of b is out of range");
   if(!c3)
```

```
        printf("value of c is out of range");
    }while(!c1 || !c2 || !c3);
    if((a+b)>c&&(b+c)>a&&(c+a)>b)
    {
        if(a==b&&b==c)
            printf("Triangle is equilateral\n");
        else if(a!=b && b!=c && c!=a)
            printf("Triangle is scalene\n");
        else
            printf("Triangle is iosceles\n");
    }
    else
        printf("Triangle cannot be formed \n");
        getch( );
    return0;
}
```

Boundary Value Analysis

Boundary value analysis focuses on the boundary of the input and output space to identify test cases because errors tend to occur near the extreme values of an input variable. The basic idea is to use input variables at their minimum, just above minimum, nominal, just below their maximum and maximum.

Considering Triangle program, we have three variables a, b and c. Each variables value ranges from1to10.

Variables	Min	Min+	Nom	Max-	Max
a	1	2	5	9	10
b	1	2	5	9	10
c	1	2	5	9	10

Boundary Value Analysis=4n+1 test cases, where n is number of variables

In Triangle program for BVA, we start by taking nominal values for **a** and **b** variables then cross product it with values min, min-, nom, max- and max values of variable **c**. similarly keeping nominal values for variables **a** and**c**, we cross product it with min ,min-,no m, max-,max values of variable **b**. Again keeping variable **b** and **c** as nominal combine with 5 values of **a**. By this we get 15 test cases in which a test case with all nominal values for**a**,**b**and**c**isrepeatedthrice,sowediscard2duplicatesuchcasesandfinallyweget15-2=**13**test cases which is equal to BVA i.e.,4(3)+1=**13**.

Test cases using Boundary value analysis for Triangle Program

Test cases	Description	Inputs			Output	Comments
		A	B	C		
BVA1	Enter the values of a(nom),b(nom)andc(min)	5	5	**1**	Isosceles	Valid
BVA2	Enter the values of a(nom),b(nom)andc(min+)	5	5	**2**	Isosceles	Valid
BVA3	Enter the values of a(nom),b(nom)andc(nom)	5	5	**5**	Equilateral	Valid
BVA4	Enterthevaluesofa(nom),b(nom)andc(max-)	5	5	**9**	Isosceles	Valid
BVA5	Enterthevaluesofa(nom),b(nom)andc(max)	5	5	**10**	Triangle cannot be formed	Valid
BVA6	Enterthevaluesofa(nom),b(min)andc(nom)	5	**1**	5	Isosceles	Valid
BVA7	Enterthevaluesofa(nom),b(min+)andc(nom)	5	**2**	5	Isosceles	Valid
BVA8	Enterthevaluesofa(nom),b(max-)andc(nom)	5	**9**	5	Isosceles	Valid
BVA9	Enterthevaluesofa(nom),b(max)andc(nom)	5	**10**	5	Trianglecannotbeformed	Valid
BVA10	Enterthevaluesofa(min),b(nom)andc(nom)	**1**	5	5	Isosceles	Valid
BVA11	Enterthevaluesofa(min+),b(nom)andc(nom)	**2**	5	5	Isosceles	Valid
BVA12	Enterthevaluesofa(max-),b(nom)andc(nom)	**9**	5	5	Isosceles	Valid
BVA13	Enterthevaluesofa(max),b(nom)andc(nom)	**10**	5	5	Trianglecannotbeformed	Valid

Lab 5: Install Selenium WebDriver with Java and Eclipse

Steps to Configure Java Environment

Step 1: Firstly, configure the Java Development Kit on your system. If not configured, then refer to the following installation steps here.

Step 2: Ensure that Java has been successfully installed on your system by running the command.

java -version

Step 3: Now, install an Integrated Development Environment (IDE) such as Eclipse, IntelliJ IDEA, or NetBeans. For this, we are using Eclipse so download it from here.

Step 4: Install the Selenium WebDriver library for Java. Download it from here. Extract the Zip File and complete the installation.

Step 5: We need web browsers such as Chrome, Firefox, Edge, or Safari. So, for this article demonstration, we will use the Chrome browser. A ChromeDriver executable file that matches your Chrome version. Download the latest release from here.

Step 6: Extract the zip file and copy the path where the chromedriver.exe file is, it is required in further steps. (e.g. - D:/chromedriver_win64/chromedriver.exe)

Next:

Create a Selenium with Java Project in Eclipse

Step 1: Launch Eclipse and select File -> New -> Java Project.

Step 2: Enter a name for your project (e.g., SeleniumJavaTutorial) and click Finish.

Step 3: Right-click on your project in Package Explorer and select Properties.

Step 4: Select Java Build Path from the left panel click on the libraries tab and then select Classpath. Click on Add External JARs and browse to the location where you downloaded the Selenium WebDriver library (e.g., selenium-java-4.1.0.zip).

Step 5: Select all the JAR files inside the zip file and click Open and also all the files inside the lib folder. (D:\selenium-java-4.11.0, D:\selenium-java-4.11.0\lib).

Step 6: Click Apply and Close to save the changes.

Step 7: Create a new package under your project by right-clicking on the src folder and selecting New -> Package.

Step 8: Add the name of your package (e.g., WebDriver)

Step 9: Create a new class under your package (e.g., WebDriver) by right-clicking on the package and selecting New -> Class, then Enter a name for your class and click Finish.

Steps for Writing Code

Step 1: Import the required packages at the top of your class:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
```

After importing if still getting errors in import just delete the module-info.java file.

Step 2: Create a main class inside the Web class.

```
public class Web {
   public static void main(String[] args) {
}
```

Lab 6: Automate Login functionality using Selenium WebDriver

```
package LoginPage;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
//Main class
public class Login {
 // Main driver method
 public static void main(String[] args)
 {
    // Path of chrome driver
    // that will be local directory path passed
    System.setProperty(
       "webdriver.chrome.driver",
       "D:\\selenium\\All softwares\\chromedriver-win64\\chromedriver.exe");
    WebDriver driver = new ChromeDriver();
```

```
try
{ // URL of the login website that is tested

                                                    driver.get("https://www.browserstack.com/users/sign_
in");

    // Maximize window size of browser
    driver.manage().window().maximize();
    Thread.sleep(5000);
    // Enter your login email id
    driver.findElement(By.id("user_email_Login")).sendKeys("xyz@gmail.com");
    // Enter your login password
    driver.findElement(By.id("user_password")).sendKeys("xyz12345");
driver.findElement(By.id("commit")).click();
 } catch (Exception e) {
    e.printStackTrace();
 } finally {
    // Close the browser
    driver.quit();
 }
}
}
```

Lab 7: DataDrivenTest with TestNG and Excel

```
// Create a class DataDrivenTestNG
package MyPack;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.*;

import java.io.IOException;

public class DataDrivenTestNG {

   private WebDriver driver;

   @BeforeMethod
   public void setUp() {
      System.setProperty("webdriver.chrome.driver", "D:\\selenium\\All softwares\\chromedriver-win64\\chromedriver.exe");
      driver = new ChromeDriver();
   }

   @AfterMethod
   public void tearDown() {
      if (driver != null) {
         driver.quit();
```

```
    }
}

@DataProvider(name = "loginData")
public Object[][] loginDataProvider() throws IOException {
    String excelPath = "D:\\ST\\Module 5\\TestData.xlsx";
    String sheetName = "Sheet1";   // or whichever sheet you use
    return ExcelUtils.getExcelData(excelPath, sheetName);
}

@Test(dataProvider = "loginData")
public void testLogin(String username, String password) throws InterruptedException {
    driver.get("D:\\selenium\\Selenium Workspace\\login.html");

    driver.findElement(By.id("username")).sendKeys(username);
    driver.findElement(By.id("password")).sendKeys(password);

    Thread.sleep(2000);
    driver.findElement(By.id("loginBtn")).click();

    String currentURL = driver.getCurrentUrl();
    if (currentURL.equals("D://selenium//Selenium Workspace//loginnext.html")) {
        System.out.println("Login successful for: " + username);
    } else {
        System.out.println("Login failed for: " + username);
    }
```

```
        WebElement message = driver.findElement(By.id("div1"));
        String messageText = message.getText();
        System.out.println("Message after submission: " + messageText);
        Thread.sleep(5000);

        Assert.assertEquals(messageText, "Successfully Logedin", "Unexpected message for user " + username);
    }
}

// Create another class in the same package MyPack
package MyPack;
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

public class ExcelUtils {
    public static Object[][] getExcelData(String filePath, String sheetName) throws IOException {
        FileInputStream file = new FileInputStream(new File(filePath));
        Workbook workbook = new XSSFWorkbook(file);
        Sheet sheet = workbook.getSheet(sheetName);

        int totalRows = sheet.getLastRowNum();  // assuming first row is header
        int totalCols = sheet.getRow(0).getLastCellNum();
```

```
        Object[][] data = new Object[totalRows][totalCols];
        for (int i = 1; i <= totalRows; i++) {
            Row row = sheet.getRow(i);
            for (int j = 0; j < totalCols; j++) {
                Cell cell = row.getCell(j);
                // Convert different cell types if needed
                data[i - 1][j] = cell.getStringCellValue();
            }
        }

        workbook.close();
        file.close();
        return data;
    }
}
```

Explanation

•@BeforeMethod — launches the browser before each test.

•@AfterMethod — quits the browser after each test.

•@DataProvider(name = "loginData") — returns all (username, password) pairs read from Excel.

•@Test(dataProvider = "loginData") — this test method will run once for each row of the Excel data.

•Use of Assert.assertEquals(...) provides proper pass/fail indication instead of just printing.

// Update pom.xml file

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>123</groupId>
  <artifactId>abc</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>abc</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
```

```
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.11.0</version>
  </dependency>

  <!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->
  <dependency>
     <groupId>org.apache.poi</groupId>
     <artifactId>poi</artifactId>
     <version>5.4.1</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->
  <dependency>
     <groupId>org.apache.poi</groupId>
     <artifactId>poi-ooxml</artifactId>
     <version>5.4.1</version>
  </dependency>

  <!-- Maven pom.xml example -->
  <dependency>
     <groupId>org.testng</groupId>
     <artifactId>testng</artifactId>
     <version>7.8.0</version> <!-- Use the latest stable version -->
     <scope>compile</scope>
  </dependency>
 </dependencies>
```

</project>

//Create Microsoft Excel Worksheet (TestData.xlsx)