UNITED
UNIVERSITY

SOFTWARE TESTING & AUDIT

LAB PRACTICAL

SESSION – 2024-2025

SUBMITTED BY-

NAME-EKTA SINGH
COURSE-BCA(IBM).
ROLL NO. -22201020026

SUBMITTED TO-

MR. ANURAG TRIPATHI

INDEX

SR. NO	PRACTICAL NAME	DATE	SIGN
1.	Roots of Quadratic Equations.		
2.	Sum of factorials up to n.(n=user input)		
3.	White Box Testing (Statement Coverage)		
4.	White Box Testing (Branch Coverage)		
5.	White Box Testing (Condition Coverage)		
6.	Black Box Testing (Equivalence Class Partitioning)		
7.	Black Box Testing (Boundary Value Analysis)		
8.	Mutation Testing		
9.	Regression Testing		
10.	Class Testing		

EXPERIMENT -1

Write a program to find roots of the quadratic equation entered by user and also depict the various test cases execution as output.

CODE

```
#include <math.h>
#include <stdio.h>
int main() {
    double a, b, c, discriminant, root1, root2, realPart, imagPart;
    printf("Enter coefficients a, b and c: ");
    scanf("%lf %lf %lf", &a, &b, &c);

    discriminant = b * b - 4 * a * c;

    // condition for real and different roots
    if (discriminant > 0) {
        root1 = (-b + sqrt(discriminant)) / (2 * a);
        root2 = (-b - sqrt(discriminant)) / (2 * a);
        printf("root1 = %.2lf and root2 = %.2lf", root1, root2);
    }

    // condition for real and equal roots
    else if (discriminant == 0) {
        root1 = root2 = -b / (2 * a);
        printf("root1 = root2 = %.2lf;", root1);
    }

    // if roots are not real
    else {
        realPart = -b / (2 * a);
        imagPart = sqrt(-discriminant) / (2 * a);
        printf("r1 = %.2lf+%.2lfi and r2 = %.2f-%.2fi", realPart,
imagPart, realPart, imagPart);
    }

    return 0;
}
```

OUTPUT

Test Case Id	Input-1 (a)	Input-2 (b)	Input-3 (c)	Output	Remarks
TC1	1	-3	2	root1=2.00 root2=1.00	Distinct roots
TC2	1	-2	1	root1=root2 =1.00	Equal roots
TC3	1	2	5	root1=1.00+2.00i root2=-1.00-2.00i	Imaginary roots

EXPERIMENT -2

Write a program the sum of factorials upto n (n is entered by user).

Example- Input=4

Output=34

Explaination-When n is eneterd as 4, so the sum is computed as follows-:sum=0!+1!+2!+3!+4

=1+1+2+6+24

=34

Check for all the possible domains of n.

CODE

```
#include <stdio.h>

// Factorial calculation
long long factorial(int num) {
    long long fact = 1;
    for (int i = 1; i <= num; i++) {
        fact *= i;
    }
    return fact;
}
int main() {
    int n;
    long long sum = 0;
    //Input
    printf("Enter a non-negative integer n: ");
    scanf("%d", &n);

    // Validate input
    if (n < 0) {
        printf("Invalid Input.\n");
        return 1;
    }

    // Calculate the sum of factorials from 0! to n!
    for (int i = 0; i <= n; i++) {
        sum += factorial(i);
    }

    // Output the result
    printf("The sum of factorials from 0! to %d! is: %lld\n", n,
sum);

    return 0;
}
```

OUTPUT

Test Case Id	Input	Output	Remarks
TC1	n=5	154	Valid Input
TC2	n=-4	Invalid Input.	Invalid Input

EXPERIMENT -3

Write a program in C to print greatest number between 2 entered numbers.Implement statement coverage technique to test all statements of program.

CODE

```
#include <stdio.h>

int main() {
   int num1, num2;

   // Input
   printf("Enter two integers: ");
   scanf("%d %d", &num1, &num2);

   // Check which number is greater and display the result
   if (num1 > num2) {
      printf(" %d\n", num1);
   } else if (num2 > num1) {
      printf("%d\n", num2);
   } else {
      printf("Equal.\n");
   }

   return 0;
}
```

OUTPUT

Test Case Id	Input-1 (a)	Input-2 (b)	Output	Remarks
TC1	167	96	167	Statement Coverage=60%
TC2	9	15	15	Statement Coverage=65%
TC3	9	9	Equal	Statement Coverage=70%

EXPERIMENT -4

Write a program in C to accept score from student and calculate grade of that student.Implement branch coverage technique to test all branches of program.

CODE

```
#include <stdio.h>
char calculateGrade(int score) {
    if (score >= 90 && score <= 100) {
        return 'A';
    } else if (score >= 80 && score < 90) {
        return 'B';
    } else if (score >= 70 && score < 80) {
        return 'C';
    } else if (score >= 60 && score < 70) {
        return 'D';
    } else if (score >= 0 && score < 60) {
        return 'F';
    } else {
        return 'I';  // Invalid score
    }
}

int main() {
    int score;
    printf("Enter the student's score (0-100): ");
    scanf("%d", &score);

    char grade = calculateGrade(score);

    if (grade == 'I') {
        printf("Invalid score \n");
    } else {
        printf("The grade for the score %d is: %c\n", score,
grade);
    }

    return 0;
}
```

OUTPUT

Test Case Id	Input	Output	Remarks
TC1	95	A	Branch coverage=16.7%
TC2	85	B	Branch coverage=16.7%
TC3	75	C	Branch coverage=16.7%
TC4	65	D	Branch coverage=16.7%
TC5	53	F	Branch coverage=16.7%
TC6	-87	Invalid score	Branch coverage=16.7%

EXPERIMENT -5

Write a C program to analyse a given integer and determine the following:

1. Whether the number is **positive**, **negative**, or **zero**.
2. Whether the number is **even** or **odd**.

CODE

```
#include <stdio.h>

void checkNumber(int number) {
    if (number > 0) {
        if (number % 2 == 0) {
            printf(" positive  even.\n");
        } else {
            printf("positive odd.\n");
        }
    } else if (number < 0) {
        if (number % 2 == 0) {
            printf("negative even.\n");
        } else {
            printf(" negative and odd.\n");
        }
    } else {
        printf("Zero.\n");
    }
}

int main() {
   // Test cases for conditional coverage
   int testNumbers[] = {10, -10, 15, -15, 0};
   for (int i = 0; i < 5; i++) {
      checkNumber(testNumbers[i]);
   }
   return 0;
}
```

OUTPUT

Test Case Id	Input	Output	Remarks
TC1	10	positive even	Condition coverage=50%
TC2	-10	negative even	Condition coverage=50%
TC3	15	positive odd	Condition coverage=50%
TC4	-15	negative odd	Condition coverage=50%
TC5	0	Zero	Condition coverage=33.33%

EXPERIMENT -6

Write a program in C that accepts user’s age (0-100) as input and categorises them as minor, adults or seniors.A company wants to hire employees that are categorised as adults.Implement equivalence partitioning technique for testing.

CODE

```
#include <stdio.h>

int main() {
    int age;

    // Input the age
    printf("Enter your age: ");
    scanf("%d", &age);

    // Categorize the age directly within the main function
    if (age < 18) {
        printf("Not Eligible");
    } else if (age >= 18 && age < 65) {
        printf(“Eligible\n");
    } else {
        printf(“Not Eligible\n");
    }
    return 0;
}
```

Equivalence Class Partitioning

Now, company will hire adults only so the range would be as follows-

Equivalence classes

- **Valid Equivalence Class -** [18,64] (for adults)
- **Invalid Eqivalence Class -** (0,18) (minor)
- **Invalid Eqivalence Class -** (65,100) (senior)

OUTPUT

Test Case Id	Input	Output	Remarks
TC1	35	Eligible	Valid
TC2	6	Not Eligible	Invalid
TC3	75	Not Eligible	Invalid

EXPERIMENT -7

Write a program in C that takes user input and calculates square root of the number. Implement Boundary Value Analysis (BVA) technique to perform testing.

Range=[0,5000]

CODE

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

int main() {
    int number;
    double result;

    // Ask the user to input a number
    printf("Enter a number (0 to 5000): ");
    scanf("%d", &number);

    // Check if the number is within the valid range (0 to 5000)
    if (number < 0 || number > 5000) {
        printf("Invalid input.\n");
    } else {
        // Calculate the square root
        result = sqrt(number);
        printf("%.2f\n",result);
    }
    return 0;
}
```

Boundary Value Analysis

1. Lower Boundary:
    - Inside the boundary: 0
    - Outside the boundary: -1
2. Upper Boundary:
    - Inside the boundary: 4999
    - At the upper boundary: 5000
    - Outside the boundary: 5001

Test Case List:

- Boundary test cases: -1, 0, 4999, 5000, 5001

OUTPUT

Test Case Id	Input	Output	Remarks
TC1	-1	Invalid input.	Invalid
TC2	0	0.00	Valid
TC3	4999	70.70	Valid
TC4	5000	70.71	Valid
TC5	5001	Invalid input.	Invalid

EXPERIMENT -8

Write a C program that takes user input in form of array and performs linear search on it . Implement Mutation Testing on this program.

ORIGINAL CODE

```
#include <stdio.h>
int main() {
    int n, i, key;
    printf("Enter no. of elements in array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter elements of array:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);}
    printf("Array is: ");
    for (i = 0; i < n; i++) {
        printf("%d\t", arr[i]);}
    printf("\nEnter key: ");
    scanf("%d", &key);
    int found = 0;
    for (i = 0; i < n; i++) {
        if (arr[i] == key) {
            printf("Found\n");
            found = 1;
            break;
        }}
    if (!found) {
        printf("Not Found\n");
    }
    return 0;
}
```

OUTPUT

Test Case Id	Input (Array)	Input (Key)	Output	Remarks
TC1	[54,67,89]	89	Found	Valid
TC2	[21,23,34,45]	23	Found	Valid

MUTANT CODE

```
#include <stdio.h>

int main() {
    int n, i, key;
    printf("Enter no. of elements in array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter elements of array:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Array is: ");
    for (i = 0; i < n; i++) {
        printf("%d\t", arr[i]);
    }
    printf("\nEnter key: ");
    scanf("%d", &key);
    int found = 0;
    for (i = 0; i < n; i++) {
        if (arr[i] != key) {  // Mutant
            printf("Not Found\n");
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Found\n");
    }
    return 0;
}
```

OUTPUT

Test Case Id	Input (Array)	Input (Key)	Output	Remarks
TC1	[54,67,89]	89	Not Found	Valid
TC2	[21,23,34,45]	23	Not Found	Valid

EXPERIMENT -9

Write a C program for calculator . Perform Regression Testing on the program.

CODE

```
#include <stdio.h>
//  addition
double add(double a, double b) {
    return a + b;
}
// subtraction
double subtract(double a, double b) {
    return a - b;
}
// multiplication
double multiply(double a, double b) {
    return a * b;
}
// division
double divide(double a, double b) {
    return a / b;
}
int main() {
    double num1, num2, result;
    char operator;

    printf("Enter first number: ");
```

```
scanf("%lf", &num1);
printf("Enter an operator (+, -, *, /): ");
scanf(" %c", &operator);
printf("Enter second number: ");
scanf("%lf", &num2);

switch (operator) {
    case '+':
        result = add(num1, num2);
        printf("Result: %.2f\n", result);
        break;
    case '-':
        result = subtract(num1, num2);
        printf("Result: %.2f\n", result);
        break;
    case '*':
        result = multiply(num1, num2);
        printf("Result: %.2f\n", result);
        break;
    case '/':
        result = divide(num1, num2);
        printf("Result: %.2f\n", result);
        break;
    default:
        printf("Error: Invalid operator.\n");
        break;
}
```

```
    return 0;
}
```

OUTPUT

Test Case Id	Input (1stnum)	Input (operator)	Input (2nd num)	Output	Remarks
TC1	65	+	98	163.00	Valid
TC2	112	-	876	-764.00	Valid
TC3	10	/	0	inf	Not Valid

MODIFIED CODE

```
#include <stdio.h>
// addition
double add(double a, double b) {
    return a + b;
}
// subtraction
double subtract(double a, double b) {
    return a - b;
}

// multiplication
double multiply(double a, double b) {
    return a * b;
}
```

```
//  division
double divide(double a, double b) {
    if (b == 0) {
        printf("Division by 0 error.\n");
        return 0;
    }
    return a / b;
}

int main() {
    double num1, num2, result;
    char operator;

    printf("Enter first number: ");
    scanf("%lf", &num1);
    printf("Enter an operator (+, -, *, /): ");
    scanf(" %c", &operator);
    printf("Enter second number: ");
    scanf("%lf", &num2);

    switch (operator) {
        case '+':
            result = add(num1, num2);
            printf("Result: %.2f\n", result);
            break;
        case '-':
```

```
        result = subtract(num1, num2);
        printf("Result: %.2f\n", result);
        break;
    case '*':
        result = multiply(num1, num2);
        printf("Result: %.2f\n", result);
        break;
    case '/':
        result = divide(num1, num2);
        printf("Result: %.2f\n", result);
        break;
    default:
        printf("Error: Invalid operator.\n");
        break;}
return 0;}
```

OUTPUT

Test Case Id	Input (1stnum)	Input (operator)	Input (2nd num)	Output	Remarks
TC1	65	+	98	163.00	Valid
TC2	112	-	876	-764.00	Valid
TC3	10	/	0	Division by 0 error.	Not Valid

EXPERIMENT -10

Write a program in java to take length and breadth of rectangle as input and calculate area and perimeter. Perform class testing on program.

CODE

```
// Defining the Rectangle class
class Rectangle {
    // Properties
    private int length;
    private int width;

    // Constructor
    public Rectangle(int length, int width) {
        this.length = length;
        this.width = width;
    }

    // Getter for length
    public int getLength() {
        return length;
    }

    // Getter for width
    public int getWidth() {
        return width;
    }

    // Method to calculate area
    public int calculateArea() {
        return length * width;
    }

    // Method to calculate perimeter
    public int calculatePerimeter() {
        return 2 * (length + width);
    }
}

// Testing the Rectangle class
public class RectangleTest {
    public static void main(String[] args) {
        // Create Rectangle objects for testing
        Rectangle rect1 = new Rectangle(5, 3);
        Rectangle rect2 = new Rectangle(10, 2);

        // Test case 1
        System.out.println("Rectangle 1:");
        System.out.println("Length: " + rect1.getLength());
        System.out.println("Width: " + rect1.getWidth());
        System.out.println("Area: " + rect1.calculateArea());
        System.out.println("Perimeter: " +
rect1.calculatePerimeter());
```

```
        // Test case 2
        System.out.println("\nRectangle 2:");
        System.out.println("Length: " + rect2.getLength());
        System.out.println("Width: " + rect2.getWidth());
        System.out.println("Area: " + rect2.calculateArea());
        System.out.println("Perimeter: " +
rect2.calculatePerimeter());
    }
}
```

OUTPUT

Test Case Id	Input (length)	Input (width)	Output (area)	Output (perimeter)	Remarks
TC1	5	3	15	16	Valid
TC2	10	2	20	24	Valid