ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B. E. (COMPUTER SCIENCE AND ENGINEERING)

VI Semester

22CSCP608 – Software Engineering Lab

Name : ....

Reg. No. : ....

ANNAMALAI UNIVERSITY

WITH COURAGE AND FAITH

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B. E. (COMPUTER SCIENCE AND ENGINEERING)

VI Semester

22CSCP608 – Software Engineering Lab

Certified that this is a bonafide record of work done by Mr./Ms...

Reg. No. .... of B.E. (CSE) in the 22CSCP608 - SOFTWARE ENGINEERING LAB during the even semester of the academic year 2025–26.

Staff-in-charge Internal Examiner

Place: Annamalai Nagar

Date :

External Examiner

CONTENTS

Annamalai University
Department of Computer Science and Engineering

VISION

To provide a congenial ambience for individuals to develop and blossom as academically superior, socially conscious and nationally responsible citizens.

MISSION

- Impart high quality computer knowledge to the students through a dynamic scholastic environment wherein they learn to develop technical, communication and leadership skills to bloom as a versatile professional.

- Develop life-long learning ability that allows them to be adaptive and responsive to the changes in career, society, technology, and environment.

- Build student community with high ethical standards to undertake innovative research and development in thrust areas of national and international needs.

- Expose the students to the emerging technological advancements for meeting the demands of the industry.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO	PEO Statements
PEO1	To equip the graduates with fundamental concepts and impart problem solving skills that will help them to pursue professional careers in Computer Science and Engineering.
PEO2	To provide the graduates with the requisite knowledge to pursue higher education and carry out research in the field of Computer Science and Engineering.
PEO3	To equip the graduates with the ability to acquire new skills in emerging areas of Computer Science and Engineering such as artificial intelligence, machine learning and data science and enable them to become successful professionals and entrepreneurs.
PEO4	To ensure that the graduates exhibit high levels of ethical and moral behavior as computer engineers in addressing societal problems and providing sustainable development for the betterment of society.

PROGRAM OUTCOMES (POs)

S. No.	Program Outcomes
PO1	**Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
PO3	**Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct Investigations of Complex Problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

S. No.	**Program Specific Outcomes**
PSO1	Ability to investigate challenging problems in various domains and exhibit programming skills to build automation solutions.
PSO2	Ability to apply the algorithms and computational techniques to construct robust and resilient computer systems and applications.
PSO3	Ability to comprehend and implement the contemporary trends in industry and research environment paving the way for innovative solutions to existing and emerging problems.

Rubrics for Laboratory Examination (Internal/External)

(**Internal:** Two tests - 15 marks each, **External:** Two questions - 25 marks each)

Rubric	**Poor** **Up to (1/2)**	**Average** **Up to (2/4)**	**Good** **Up to (3/6)**	**Excellent** **Up to (5/8*)**
Syntax and Logic **Ability to understand, specify the data structures appropriate for the problem domain**	Program does not compile with typographical errors and incorrect logic leading to infinite loops.	Program compiles that signals major syntactic errors and logic shows severe errors.	Program compiles with minor syntactic errors and logic is mostly correct with occasional errors.	Program compiles with evidence of good syntactic understanding of the syntax and logic used.
Modularity **Ability to decompose a problem into coherent and reusable functions, files, classes, or objects (as appropriate for the programming language and platform).**	Program is one big Function or is decomposed in ways that make little/no sense.	Program is decomposed into units of appropriate size, but they lack coherence or reusability. Program contains unnecessary repetition.	Program is decomposed into coherent units, but may still contain some unnecessary repetition.	Program is decomposed into coherent and reusable units, and unnecessary repetition are eliminated.
Clarity and Completeness **Ability to code formulae and algorithms that produce appropriate results. Ability to apply rigorous test case analysis to the problem domain.**	Program does not produce appropriate results for most inputs. Program shows little/no ability to apply different test cases.	Program approaches appropriate results for most inputs, but contain some miscalculations. Program shows evidence of test case analysis, but missing significant test cases or mistaken some test cases.	Program produces appropriate results for most inputs. Program shows evidence of test case analysis that is mostly complete, but missed to handle all possible test cases.	Program produces appropriate results for all inputs tested. Program shows evidence of excellent test case analysis, and all possible cases are handled appropriately.

* 8 marks for syntax and logic, 8 marks for modularity, and 9 marks for Clarity and Completeness.

Rubric for CO3

Rubric for CO3 in Laboratory Courses				
Rubric	**Distribution of 10 Marks for CIE/SEE Evaluation Out of 40/60 Marks**			
	Up To 2.5 Marks	**Up To 5 Marks**	**Up To 7.5 Marks**	**Up To 10 marks**
Demonstrate an ability to listen and answer the viva questions related to programming skills needed for solving real-world problems in Computer Science and Engineering.	Poor listening and communication skills. Failed to relate the programming skills needed for solving the problem.	Showed better communication skill by relating the problem with the programming skills acquired but the description showed serious errors.	Demonstrated good communication skills by relating the problem with the programming skills acquired with few errors.	Demonstrated excellent communication skills by relating the problem with the programming skills acquired and have been successful in tailoring the description.

Ex. No: 01 **MATRIX MULTIPLICATION - CAUSES OF FAILURE**
Date:

AIM:

To write a C program for matrix multiplication to understand the causes of failure.

ALGORITHM:

1. Start the program.
2. Input the number of rows and columns for matrix A.
3. Input the number of rows and columns for matrix B.
4. Check for multiplication compatibility:
   - If the number of columns in A is not equal to the number of rows in B, print an error message and exit.
5. Input the elements of matrix A.
   - If any input is not an integer, print an error message and exit.
6. Input the elements of matrix B.
   - If any input is not an integer, print an error message and exit.
7. Initialize the result matrix with zeros.
8. Perform matrix multiplication using the formula:
   `result[i][j] += a[i][k] * b[k][j];`
9. Display the resultant matrix.
10. End the program.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a[10][10], b[10][10], result[10][10];
    int p, q, r, s, i, j, k;
    printf("Enter rows and columns of A Matrix: ");
    scanf("%d %d", &p, &q);
    printf("Enter rows and columns of B Matrix: ");
    scanf("%d %d", &r, &s);
    if (q != r) {
        printf("Test Case 2: Matrix Multiplication is not possible\n");
        printf("The columns of A Matrix are not equal to rows of B Matrix\n");
        printf("Case 2: Failure\n");
        return 0;
    }
    printf("Test Case 1: Matrix Multiplication can be done\n");
    printf("Case 1: Success\n");
    printf("Enter the elements of Matrix A:\n");
    for (i = 0; i < p; i++) {
        for (j = 0; j < q; j++) {
            if (scanf("%d", &a[i][j]) != 1) {
                printf("Test Case 3: Matrix Multiplication is not possible\n");
                printf("One or more values are not an integer\n");
                printf("Case 3: Failure\n");
                return 0;
            }
        }
    }
    printf("Enter the elements of Matrix B:\n");
    for (i = 0; i < r; i++) {
        for (j = 0; j < s; j++) {
            if (scanf("%d", &b[i][j]) != 1) {
                printf("Test Case 3: Matrix Multiplication is not possible\n");
                printf("One or more values are not an integer\n");
                printf("Case 3: Failure\n");
                return 0;
            }
        }
    }
    for (i = 0; i < p; i++) {
        for (j = 0; j < s; j++) {
            result[i][j] = 0;
            for (k = 0; k < q; k++) {
                result[i][j] += a[i][k] * b[k][j];
            }
        }
    }
    printf("The result of matrix multiplication is:\n");
    for (i = 0; i < p; i++) {
        for (j = 0; j < s; j++) {
            printf("%d\t", result[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT:

```
Enter rows and columns of A Matrix: 2 3
Enter rows and columns of B Matrix: 2 1
Test Case 2: Matrix Multiplication is not possible
The columns of A Matrix are not equal to rows of B Matrix
Case 2: Failure

Enter rows and columns of A Matrix: 3 2
Enter rows and columns of B Matrix: 2 2
Test Case 1: Matrix Multiplication can be done
Case 1: Success
Enter the elements of Matrix A:
1 5
3 4
7 1
Enter the elements of Matrix B:
2 3
4 1
The result of matrix multiplication is:
22      8
22      13
18      22
```

```
Enter rows and columns of A Matrix: 2 2
Enter rows and columns of B Matrix: 2 3
Test Case 1: Matrix Multiplication can be done
Case 1: Success
Enter the elements of Matrix A:
1 2
1.5 2
Test Case 3: Matrix Multiplication is not possible
One or more values are not an integer
Case 3: Failure
```

RESULT:

Thus, the C Program of Matrix Multiplication to understand causes of failure has been executed successfully.

Ex. No: 02 **BINARY SEARCH - PATH TESTING**
Date:

AIM:

To write a C program for binary search and analyze the path testing for test cases.

ALGORITHM:

1. Start the program.
2. Input the number of elements in the array.
3. Input the elements of the array (assumed to be sorted).
4. Input the key value to be searched.
5. Perform binary search:
    - Set low = 0 and high = n - 1.
    - While low is less than or equal to high:
        - Find mid = (low + high) / 2.
        - If x[mid] == key, return mid (element found).
        - If x[mid] < key, set low = mid + 1 (search in the right half).
        - Otherwise, set high = mid - 1 (search in the left half).
    - If the loop ends, return -1 (element not found).
6. Display whether the key is found along with its position or print that it is not found.
7. End the program.

PROGRAM:

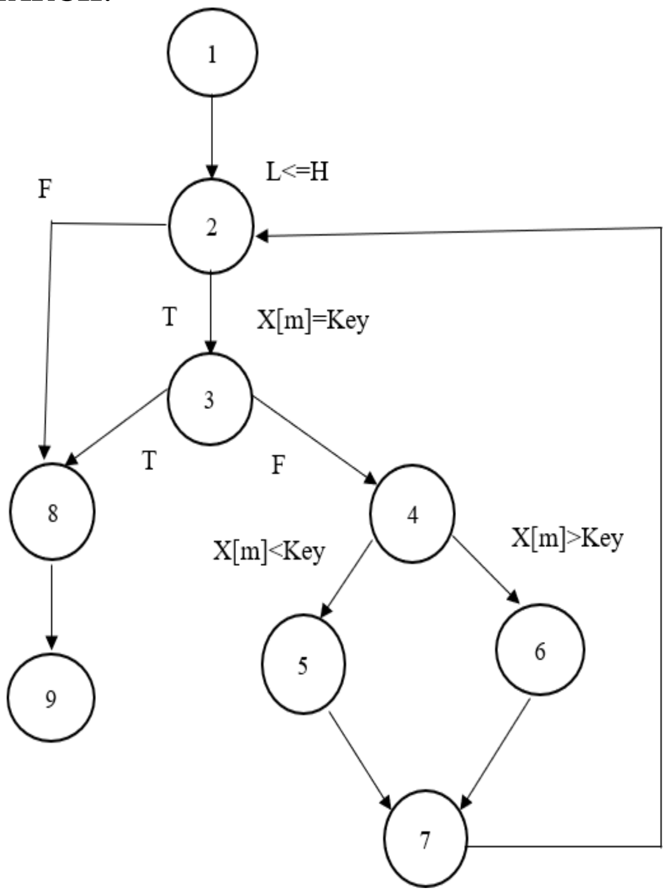
```
#include <stdio.h>
int bs(int x[], int low, int high, int key) {
    printf("1");
    int mid;
    printf("-2");
    while (low <= high) {
        mid = (low + high) / 2;
        printf("-3");
        if (x[mid] == key) {
            printf("-8-9");
            return mid;
        }
        printf("-4");
        if (x[mid] < key) {
            printf("-5");
            low = mid + 1;
        } else {
            printf("-6");
            high = mid - 1;
        }
        printf("-7");
    }
    printf("-8");
    return -1;
    printf("-9");
}
```

```
int main() {
    int a[200], n, s, k;
    printf("Enter the Element Length: ");
    scanf("%d", &n);
    printf("\nEnter the Elements Value: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
    printf("\nEnter the Key Value: ");
    scanf("%d", &k);
    printf("\nPath: ");
    s = bs(a, 0, n - 1, k);
    if (s != -1) {
        printf("\nThe Element %d found at index of %d\n", k, s + 1);
    } else {
        printf("\nThe Element %d not found.",k);
    }
}
```

BINARY SEARCH FUNCTION WITH LINE NUMBER:

Program	Line No.
`int bs(int x[], int low, int high, int key) {`	
`    int mid;`	1
`    while (low <= high) {`	2
`        mid = (low + high) / 2;`	
`        if (x[mid] == key) {`	3
`            return mid;`	8
`        }`	
`        if (x[mid] < key) {`	4
`            low = mid + 1;`	5
`        } else {`	
`            high = mid - 1;`	6
`        }`	
`    }`	7
`    return -1;`	8
`}`	9

GRAPH FOR BINARY SEARCH:



OUTPUT:

Test Cases – Binary Search:

Paths	Inputs		Expected Output	Remarks
	X[]	Key		
P: 1-2-3-8-9	{10, 20, 30, 40, 50}	30	Success	Key ∈ X[ ] and Key == X[mid]
P: 1-2-3-4-6-7-3-4-5-7-3-8-9	{10, 20, 30, 40, 50}	20	Repeat and Success	Key < X[mid] Search 1st Half
P: 1-2-3-4-5-7-3-8-9	{10, 20, 30, 40, 50}	40	Repeat and Success	Key > X[mid] Search 2nd Half
P: 1-2-3-4-5-7-3-4-5-7-3-4-5-7-8	{10, 20, 30, 40, 50}	60 or 05	Repeat and Failure	Key ∉ X[ ]

RESULT:

Thus, the C program for binary search and analyzing path testing for test cases has been executed successfully.

Ex. No: 03 **BOUNDARY VALUE ANALYSIS**
Date:

AIM:

To write a C program to derive test cases based on Boundary Value Analysis.

ALGORITHM:

1. Start the program.
2. Define a function `nature_of_roots(a, b, c)` to determine the nature of roots of the quadratic equation $ax^2 + bx + c = 0$.
   - Compute the discriminant: $D = b^2 - 4ac$.
   - If `a=0`, print "Not a Quadratic Equation" and return.
   - If `D>0`, print "Real Roots".
   - If `D=0`, print "Equal Roots".
   - If `D<0`, print "Imaginary Roots".
3. In `main()`, initialize `testcase = 1`.
4. Print the table header: `Testcase a b c Result`.
5. Run a loop for 13 test cases using Boundary Value Analysis (BVA):
   - Assign values to `a, b,` and `c` for each test case.
   - Print the test case number, values of `a, b,` and `c`.
   - Call the function `nature_of_roots(a, b, c)` to determine the nature of roots.
   - Increment the test case number.
6. End the program.

PROGRAM:

```c
#include <stdio.h>
void nature_of_roots(int a, int b, int c) {
    int D = b * b - 4 * a * c;
    if (a == 0) {
        printf("Not a Quadratic Equation\n");
        return;
    }

    if (D > 0) {
        printf("Real Roots\n");
    } else if (D == 0) {
        printf("Equal Roots\n");
    } else {
        printf("Imaginary Roots\n");
    }
}
int main() {
    int a, b, c, testcase = 1;
    printf("Testcase\t a\t b\t c\t Result\n");
    while (testcase <= 13) {
        if (testcase == 1)  { a = 0;   b = 50;  c = 50; }
        if (testcase == 2)  { a = 1;   b = 50;  c = 50; }
        if (testcase == 3)  { a = 50;  b = 50;  c = 50; }
        if (testcase == 4)  { a = 99;  b = 50;  c = 50; }
        if (testcase == 5)  { a = 100; b = 50;  c = 50; }
        if (testcase == 6)  { a = 50;  b = 0;   c = 50; }
        if (testcase == 7)  { a = 50;  b = 1;   c = 50; }
        if (testcase == 8)  { a = 50;  b = 99;  c = 50; }
        if (testcase == 9)  { a = 50;  b = 100; c = 50; }
        if (testcase == 10) { a = 50;  b = 50;  c = 0;  }
        if (testcase == 11) { a = 50;  b = 50;  c = 1;  }
        if (testcase == 12) { a = 50;  b = 50;  c = 99; }
        if (testcase == 13) { a = 50;  b = 50;  c = 100;}
        printf("%d\t\t%d\t%d\t%d\t", testcase, a, b, c);
        nature_of_roots(a, b, c);
        testcase++;
    }
    return 0;
}
```

OUTPUT:

```
Testcase         a       b       c       Result
1               0       50      50      Not a Quadratic Equation
2               1       50      50      Real Roots
3               50      50      50      Imaginary Roots
4               99      50      50      Imaginary Roots
5               100     50      50      Imaginary Roots
6               50      0       50      Imaginary Roots
7               50      1       50      Imaginary Roots
8               50      99      50      Imaginary Roots
9               50      100     50      Equal Roots
10              50      50      0       Real Roots
11              50      50      1       Real Roots
12              50      50      99      Imaginary Roots
13              50      50      100     Imaginary Roots
```

RESULT:

Thus, the C program to derive test cases based on Boundary Value Analysis has been executed successfully.

Ex. No: 04 **CAUSE - EFFECT GRAPH**
Date:

AIM:

To write a C program for a cause-effect graph to check whether a defect is found in the program.

ALGORITHM:

1. Start the program.
2. Prompt the user to enter the values of three sides: a, b, and c.
3. Validate input:
   - If the input is not three valid integers, print "Impossible" and terminate the program.
4. Check if the given values form a valid triangle:
   - A triangle is valid if (a + b > c) && (b + c > a) && (c + a > b).
   - If the condition fails, print "Not a Triangle" and terminate.
5. Classify the triangle type:
   - If a == b == c, print "It is an Equilateral Triangle".
   - Else if any two sides are equal, print "It is an Isosceles Triangle".
   - Else, print "It is a Scalene Triangle".
6. End the program.

PROGRAM:

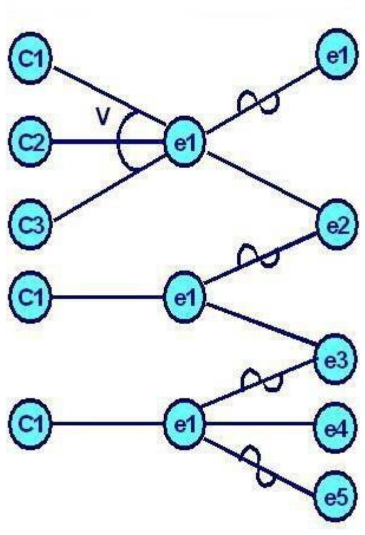
```
#include <stdio.h>
void main() {
    int a, b, c;
    printf("Enter the values of a, b, and c: ");
    if (scanf("%d %d %d", &a, &b, &c) != 3) {
        printf("\nImpossible\n");
        return;
    }
    if (((a + b) > c) && ((b + c) > a) && ((c + a) > b)) {
        if ((a == b) && (b == c)) {
            printf("\nIt is an Equilateral Triangle");
        } else if ((a == b) || (b == c) || (c == a)) {
            printf("\nIt is an Isosceles Triangle");
        } else {
            printf("\nIt is a Scalene Triangle");
       }
    } else {
        printf("\nNot a Triangle");
    }
}
```

STEP: 1: Recognize and describe the input conditions (causes) and actions (effect).

The causes designated by letter "C" are as under		The effects designated by letter "e" are as under	
C1	Side "x" is less than sum of "y" and "z"	e1	Not a triangle
C2	Side "y" is less than sum of "x" and "z"	e2	Scalene triangle
C3	Side "z" is less then sum of "x" and "y"	e3	Isosceles triangle
C4	Side "x" is equal to side "y"	e4	Equilateral Triangle
C5	Side "x" is equal to side "z"	e5	Impossible
C6	Side "y" is equal to side "z"		

STEP: 2: Build up a cause-effect graph

Cause – Effect graph for Triangle:



STEP: 3: Convert cause-effect graph into a decision table

Conditions	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
C1: X < Y+Z?	0	1	1	1	1	1	1	1	1	1	1
C2: X < Y+Z?	X	0	1	1	1	1	1	1	1	1	1
C3: X < Y+Z?	X	X	0	1	1	1	1	1	1	1	1
C3: X=Y?	X	X	X	1	1	1	1	0	0	0	0
C4: X=Y?	X	X	X	1	1	0	0	1	1	0	0
C5: X=Y?	X	X	X	1	1	0	0	1	1	0	0
C6: X=Y?	X	X	X	1	0	1	0	1	0	1	0
e1: Not a Triangle	1	1	1								
e2: Scalene											1
e3: Isosceles							1		1	1	
e4: Equilateral				1							
e5: Impossible					1	1		1			

OUTPUT:

Test Case	X	Y	Z	Expected Output
1	4	1	2	Not a triangle
2	1	4	2	Not a triangle
3	1	2	4	Not a triangle
4	5	5	5	It is an Equilateral
5	?	?	?	Impossible
6	?	?	?	Impossible
7	2	2	3	It is an Isosceles
8	?	?	?	Impossible
9	2	3	2	It is an Isosceles
10	3	2	2	It is an Isosceles
11	3	4	5	It is a Scalene

RESULT:

Thus, the C program for cause-effect graph to check whether a defect is found in the program has been executed successfully.

Ex. No: 05 **DATA FLOW TESTING**
Date:

AIM:

To write a C program to perform data flow testing for the given code and find out all definition-use (d-use) pairs.

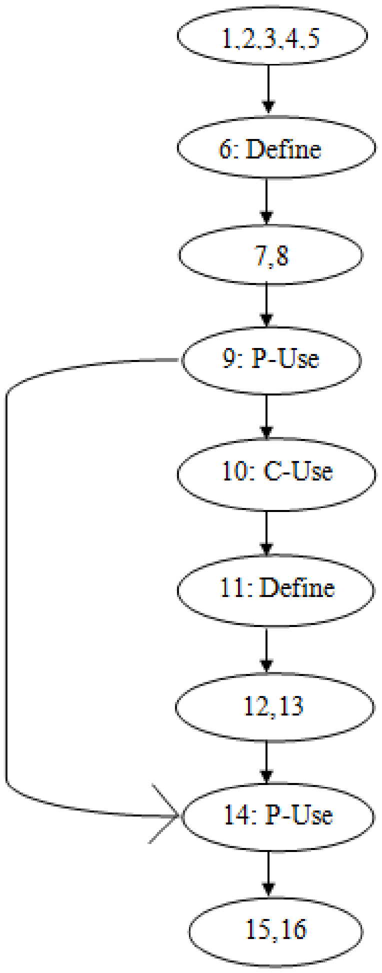
ALGORITHM:

1. Start the program.
2. Declare three integer variables: a, b, and t.
3. Clear the screen using system("cls").
4. Prompt the user to enter the first number and store it in variable a.
5. Prompt the user to enter the second number and store it in variable b.
6. Compare values of a and b:
   - If a < b, perform the swap:
     - Store a in t (definition of t, use of a).
     - Assign b to a (definition of a, use of b).
     - Assign t to b (definition of b, use of t).
7. Display the values of a and b after ensuring a >= b.
8. End the program.

PROGRAM CODE:

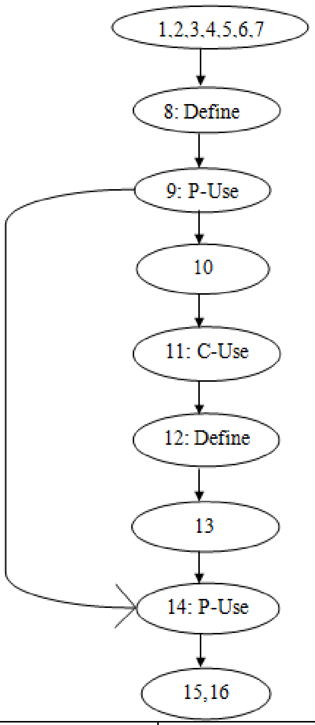
```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void main() {
	int a, b, t;
	system("cls");
	printf("Enter the first number ");
	scanf("%d",&a);
	printf("Enter the second number ");
	scanf("%d", &b);
	if (a<b) {
		t=a;
		a=b;
		b=t;
	}
	printf("a=%d b=%d",a,b);
	getch();
}
```

Variable 'a' Data Flow Graph:



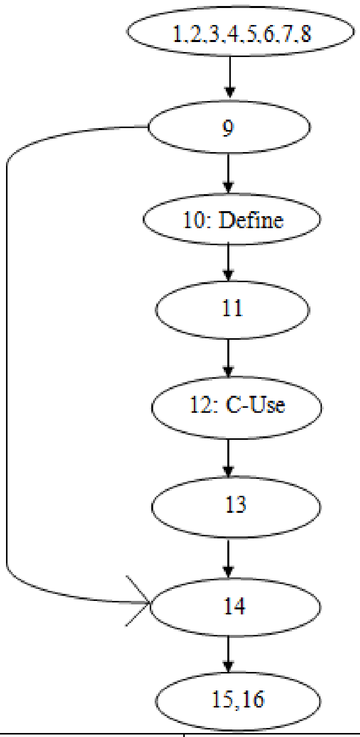
DU-PATHS	DC-PATHS

Variable 'b' Data Flow Graph:



DU-PATHS	DC-PATHS

Variable 't' Data Flow Graph:



DU-PATHS	DC-PATHS

VARIABLE	DEFINED AT	USED AT
a		
b		
t		

OUTPUT:

```
Enter the first number 4
Enter the second number 6
a=6 b=4
```

RESULT:

Thus, the C program to perform data flow testing and find out all d-use pairs has been executed successfully.

Ex. No: 06 **WORKING OF LOOPS**
Date:

AIM:

To write a C program to demonstrate the working of the looping constructs.

ALGORITHM:

For the while loop program

1. Start the program.
2. Declare an integer variable i and initialize it to 0.
3. Print the value of i before entering the loop.
4. Use a while loop that continues as long as i < 5.
   - Inside the loop, print the current value of i.
   - Increment i using i++.
5. After the loop exits, print the final value of i.
6. End the program.

For the do-while loop program

1. Start the program.
2. Declare an integer variable month.
3. Use a do-while loop to prompt the user to enter their birth month.
4. Read the input using scanf().
5. Continue the loop if month is less than 1 or greater than 12.
6. End the program once a valid month (1 to 12) is entered.

PROGRAM:

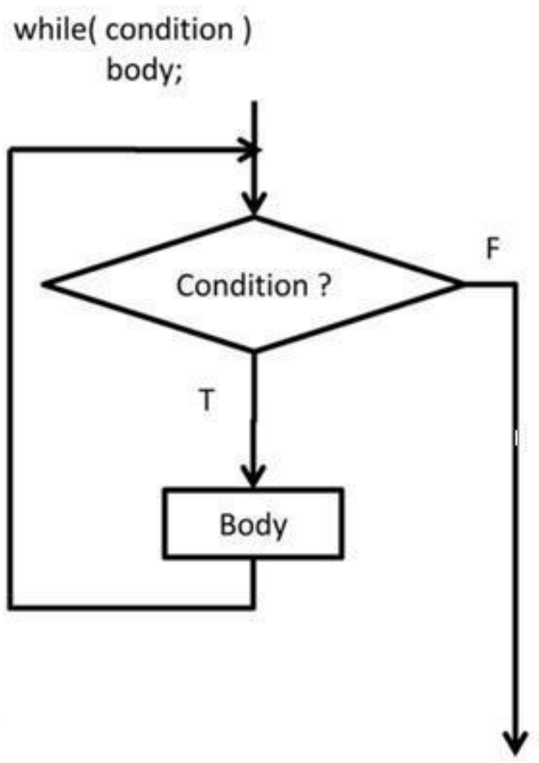
While loops in C:

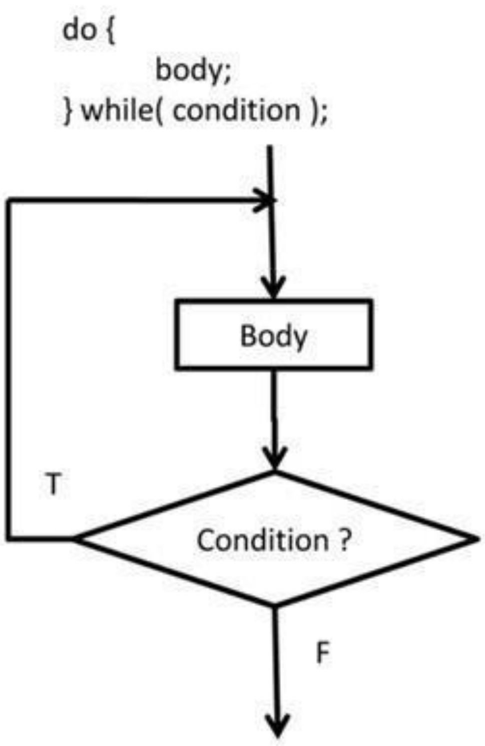
```
#include <stdio.h>
int main() {
      int i=0;
      printf("Before loop, i=%d\n", i);
      while(i<5) {
            printf("i=%d\n", i++);
      }
      printf("After loop, i=%d\n", i);
      return 0;
}
```

Do While loops in C:
Code:

```
#include <stdio.h>
int main() {
      int month;
      do {
            printf("Please enter the month of your birth > ");
            scanf("%d", &month);
      } while (month < 1 || month > 12);
      return 0;
}
```

While Vs Do-While Constructs:





OUTPUT:

While Loop:

```
Before loop, i=0
i=0
i=1
i=2
i=3
i=4
After loop, i=5
```

Do While Loop:

```
Please enter the month of your birth > 13
Please enter the month of your birth > -1
Please enter the month of your birth > 15
Please enter the month of your birth > -3
Please enter the month of your birth > 12
```

RESULT:

Thus, the C program to demonstrate the working of the looping constructs has been executed successfully.

Ex. No: 07 **CHECKBOX COUNT**
Date:

AIM:

To write and test a program to count number of check boxes on the page checked and unchecked count using selenium tool.

ALGORITHM:

1. Set up the WebDriver for the Edge browser by specifying the path to the `msedgedriver.exe`.
2. Initialize the EdgeDriver and configure it to wait implicitly for elements to load (`set timeout`).
3. Maximize the browser window to make it easier to interact with the page.
4. Navigate to the local HTML file using `driver.get()` with the correct file path (`file:///D:/new.html`).
5. Find all checkbox elements on the page using the XPath `//input[@type='checkbox']`.
6. Print the total number of checkboxes found on the page.
7. Loop through the list of checkboxes, and click every 4th checkbox `(i += 4)`.
8. Initialize two counters (`checkedCount` and `uncheckedCount`) to zero.
9. Loop through all checkboxes and check if each one is selected (`checkbox.isSelected()`).
10. If the checkbox is selected, increment the `checkedCount`; if not, increment the `uncheckedCount`.
11. After the loop, print the total number of selected checkboxes (`checkedCount`) and unselected checkboxes (`uncheckedCount`).

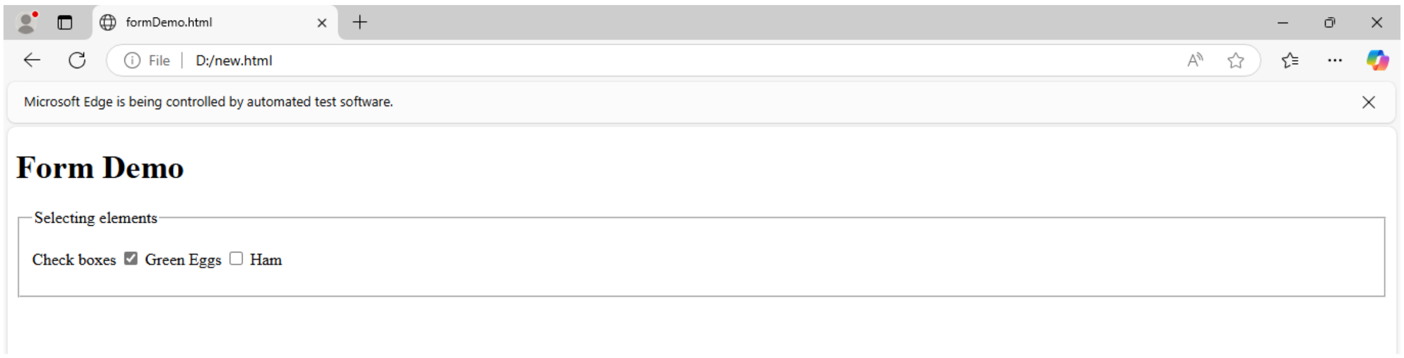
PROGRAM CODE:

```java
package demo;
import java.util.List;
import java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.edge.EdgeDriver;

public class checkbox {
    public static void main(String[] args) {
        System.setProperty("webdriver.edge.driver", "C:\\edgedriver\\msedgedriver.exe");

        EdgeDriver driver = new EdgeDriver();
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
        driver.manage().window().maximize();
        driver.get("file:///D:/new.html");

        List<WebElement>checkboxes=driver.findElements(By.xpath("//input[@type='checkbox']"));
        System.out.println("Total checkboxes: " + checkboxes.size());
        for (int i = 0; i < checkboxes.size(); i += 4) {
            checkboxes.get(i).click();
        }

        int checkedCount = 0, uncheckedCount = 0;
        for (WebElement checkbox : checkboxes) {
            if (checkbox.isSelected()) {
                checkedCount++;
            } else {
                uncheckedCount++;
            }
        }
        System.out.println("Selected checkboxes: " + checkedCount);
        System.out.println("Unselected checkboxes: " + uncheckedCount);
    }
}
```

HTML CODE:

```
<!DOCTYPE HTML>
<html lang="en">
<head>
    <title>Check Box</title>
</head>
<body>
    <h1>Check Box</h1>
    <form>
        <fieldset>
            <legend>Selecting elements</legend>
            <p>
                <label for="chkEggs">Check boxes</label>
                <input type="checkbox" id="chkEggs" value="greenEggs" />
                <label for="chkEggs">Green Eggs</label>

                <input type="checkbox" id="chkHam" value="ham" />
                <label for="chkHam">Ham</label>
            </p>
        </fieldset>
    </form>
</body>
</html>
```

OUTPUT:


formDemo.html
File | D:/new.html
Microsoft Edge is being controlled by automated test software.

Form Demo

Selecting elements

Check boxes ☑ Green Eggs ☐ Ham

```
Total checkboxes: 2
Selected checkboxes: 1
Unselected checkboxes: 1
```

RESULT:

Thus, the Program to count number of check boxes on the page, checked and unchecked using Selenium tool has been executed successfully.

Ex. No: 08 **OBJECT COUNT**
Date:

AIM:

To Write and test a program to provide total number of objects present/ available on the page.

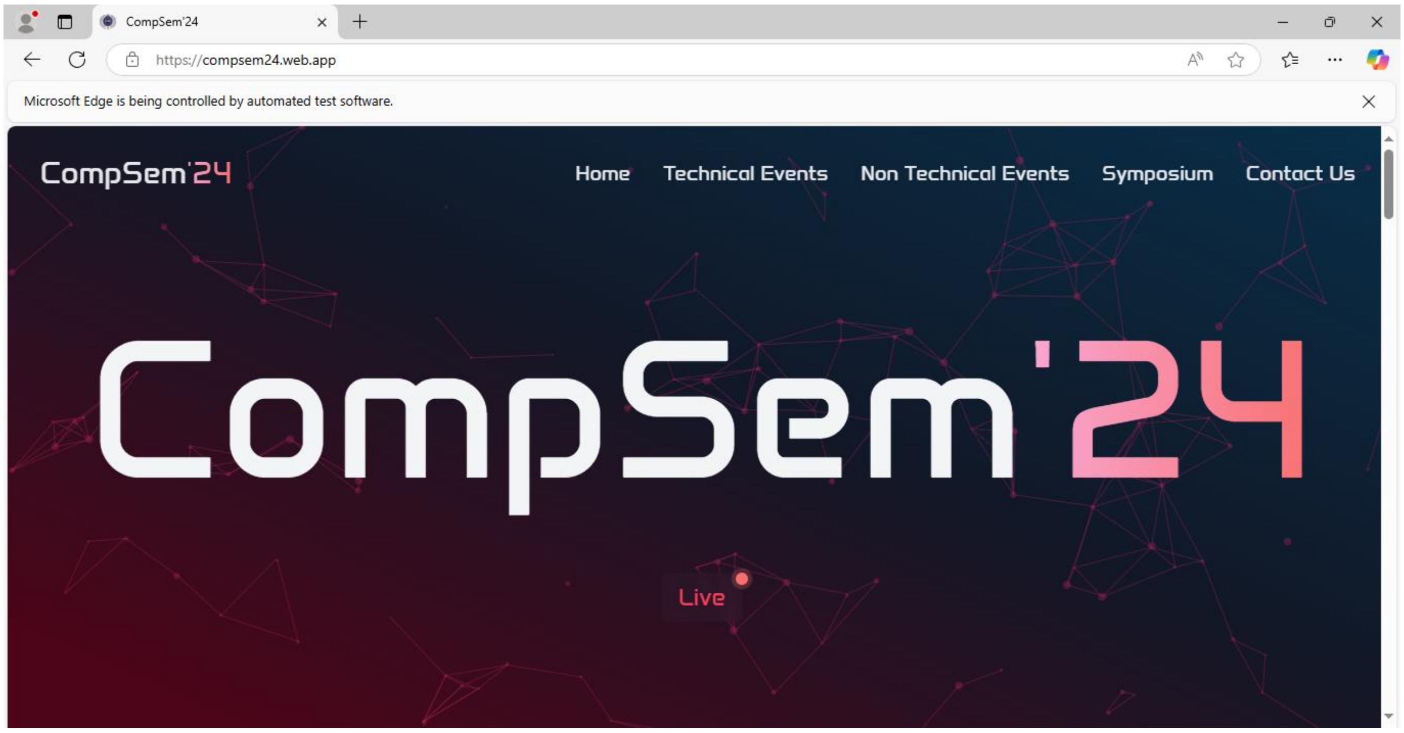
ALGORITHM:

1. Set up WebDriver: Set the system property for msedgedriver.exe and initialize the EdgeDriver.
2. Configure WebDriver: Set an implicit wait for elements to load and maximize the browser window.
3. Navigate to the Web Page: Open the website using `driver.get()`.
4. Find Paragraph Elements: Use `findElements` with `XPath` `//p` to get all `<p>` elements on the page.
5. Count Paragraph Elements: Get the size of the list of paragraphs.
6. Display the Count: Print the total number of `<p>` elements.

PROGRAM:

```
package demo;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.edge.EdgeDriver;
import java.time.Duration;
import java.util.List;

public class objectcount {
    public static void main(String[] args) {
      System.setProperty("webdriver.edge.driver", "C:\\edgedriver\\msedgedriver.exe");
      EdgeDriver driver = new EdgeDriver();

        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
        driver.manage().window().maximize();

        driver.get("https://compsem24.web.app/");

        List<WebElement> paragraphs = driver.findElements(By.xpath("//p"));
        int num = paragraphs.size();

        System.out.println("The number of <p> elements present are: " + num);
    }
}
```

OUTPUT:



```
The number of <p> elements present are: 31
```

RESULT:

Thus, the Program to provide total number of objects present/available on the page has been executed successfully.

Ex. No: 09 **WEBPAGE LOGIN AUTOMATION**
Date:

AIM:

To write and test a program to login a specific web page using selenium tool.

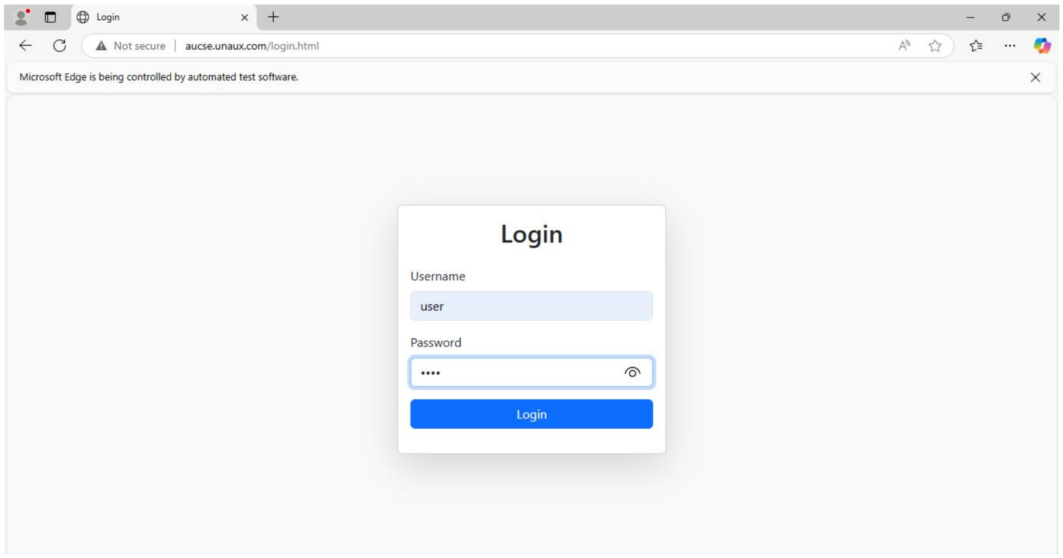
ALGORITHM:

1. Set Up WebDriver: Set the path for the EdgeDriver and initialize the WebDriver.
2. Maximize Window: Maximize the browser window.
3. Set Implicit Wait: Set an implicit wait time of 10 seconds.
4. Open Login Page: Navigate to the login page URL (`http://aucse.unaux.com/login.html`).
5. Locate Elements: Find the username, password input fields, and login button.
6. Enter Credentials: Input the username (`"user"`) and password (`"pass"`).
7. Click Login: Click the login button to submit the credentials.
8. Wait for Page to Load: Pause for 5 seconds to allow the page to load.
9. Verify Login: Check if the current URL contains "`index.html`" to determine if login was successful.
10. Print Result: Print "`Test Passed`" if login is successful, or "`Test Failed`" if not.

PROGRAM CODE:

```java
package demo;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.edge.EdgeDriver;
import java.time.Duration;

public class login {
    public static void main(String[] args) {
        System.setProperty("webdriver.edge.driver", "C:\\edgedriver\\msedgedriver.exe");
        WebDriver driver = new EdgeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

        driver.get("http://aucse.unaux.com/login.html");

        WebElement username = driver.findElement(By.id("username"));
        WebElement password = driver.findElement(By.id("password"));
        WebElement loginButton = driver.findElement(By.cssSelector("button[type='submit']"));
        username.sendKeys("user");
        password.sendKeys("pass");

        loginButton.click();
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        if (driver.getCurrentUrl().contains("index.html")) {
            System.out.println("Test Passed: Login successful!");
        } else {
            System.out.println("Test Failed: Login unsuccessful.");
        }
    }
}
```

OUTPUT:


Login

aucse.unaux.com/login.html

Microsoft Edge is being controlled by automated test software.

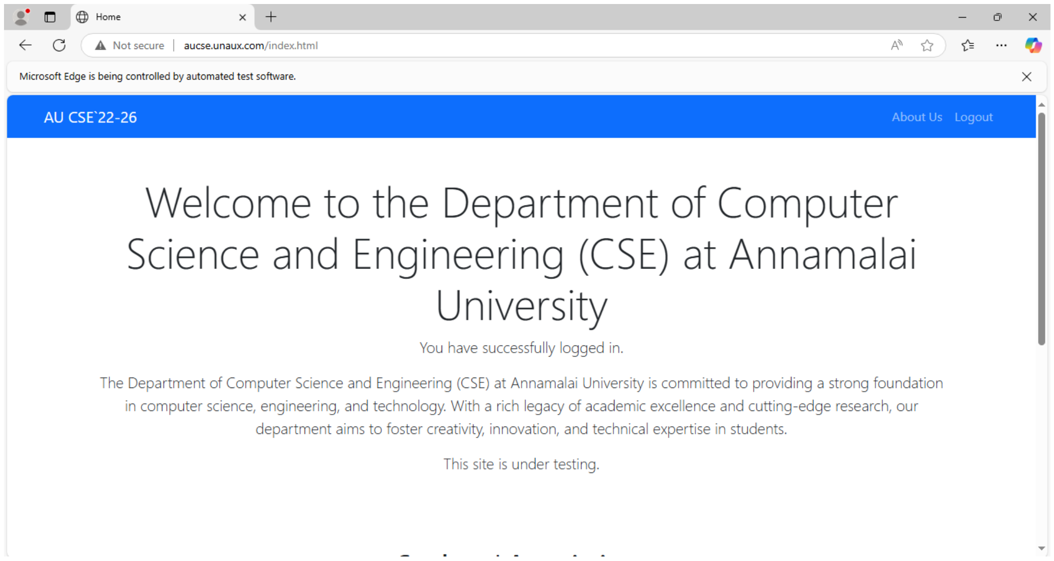
Login

Username

user

Password

••••


Home

aucse.unaux.com/index.html

Microsoft Edge is being controlled by automated test software.

AU CSE`22-26

Welcome to the Department of Computer Science and Engineering (CSE) at Annamalai University

You have successfully logged in.

The Department of Computer Science and Engineering (CSE) at Annamalai University is committed to providing a strong foundation in computer science, engineering, and technology. With a rich legacy of academic excellence and cutting-edge research, our department aims to foster creativity, innovation, and technical expertise in students.

This site is under testing.

```
Test Passed: Login successful!
```

RESULT:

Thus, the Program to login a specific web page using Selenium tool has been executed successfully.

Ex. No: 10 **STUDENTS SCORE FILTER**
Date:

AIM:

To write and test a program to select the number of students who have scored more than 60 in any one subject (or all subjects).

ALGORITHM:

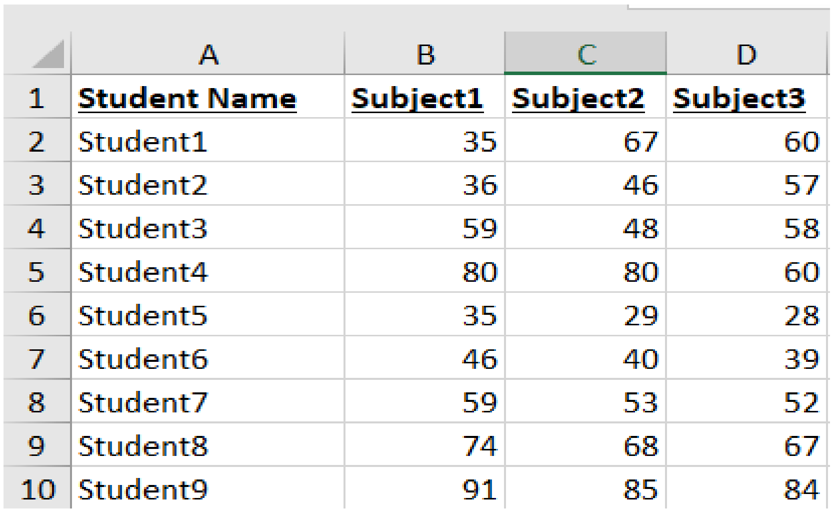
1. Set Input File: Specify the path to the Excel file.
2. Check File Existence: Ensure the file exists. If not, display an error and stop.
3. Open Workbook: Open the Excel file using `Workbook.getWorkbook().`
4. Read First Sheet: Get the first sheet of the workbook.
5. Loop Through Rows: For each row, initialize a flag `studentHasHighScore` as false.
6. Loop Through Columns: For each cell in the row, check if it contains a number.
7. Check for High Score: If the score is greater than 60, set the flag `studentHasHighScore` to true.
8. Count High Scorers: If the flag is true, increment the counter for high-scoring students.
9. Display Result: Print the total count of students who scored more than `60`.
10. Handle Errors: Handle any exceptions or warnings (e.g., non-numeric values).

PROGRAM:

```
package demo;
import java.io.File;
import java.io.IOException;
import jxl.Cell;
import jxl.CellType;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;

public class student_excel_read {
    private String inputFile;

    public void setInputFile(String inputFile) {
        this.inputFile = inputFile;
    }
    public void read() throws IOException {
        File inputWorkbook = new File(inputFile);
        if (!inputWorkbook.exists()) {
            System.out.println("Error: File not found!");
            return;
        }

        Workbook w;
        int count = 0;
        try {
            w = Workbook.getWorkbook(inputWorkbook);
            Sheet sheet = w.getSheet(0);

            for (int j = 0; j < sheet.getRows(); j++) {
                boolean studentHasHighScore = false;

                for (int i = 0; i < sheet.getColumns(); i++) {
                    Cell cell = sheet.getCell(i, j);
                    if (cell.getType() == CellType.NUMBER) {
                        try {
                            int score = Integer.parseInt(cell.getContents());
                            if (score > 60) {
                                studentHasHighScore = true;
                            }
                        } catch (NumberFormatException e) {
                                System.out.println("Warning: Non-numeric value in row " + j +
                                                                          ", column " + i);
                        }
                    }
                }
                if (studentHasHighScore) {
                    count++;
                }
            }
            System.out.println("Total students who scored more than 60 in one or more subjects:
                                                                                  " + count);
        } catch (BiffException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) throws IOException {
        student_excel_read test = new student_excel_read();
        test.setInputFile("D:\\data.xls");
        test.read();
    }
}
```

OUTPUT:

INPUT FILE:



	A	B	C	D
1	Student Name	Subject1	Subject2	Subject3
2	Student1	35	67	60
3	Student2	36	46	57
4	Student3	59	48	58
5	Student4	80	80	60
6	Student5	35	29	28
7	Student6	46	40	39
8	Student7	59	53	52
9	Student8	74	68	67
10	Student9	91	85	84

AFTER EXECUTION:

```
Total students who scored more than 60 in one or more subjects: 4
```

RESULT:

Thus, the Program to select the number of students who have scored more than 60 in one or more subjects has been executed successfully.

Ex. No: 11 **GCD CALCULATOR**
Date:

AIM:

To write a Java script to develop a web page which calculates the GCD of 2 numbers using Selenium tool.

ALGORITHM:

1. Set up WebDriver: Set the path for msedgedriver.exe and initialize the EdgeDriver.
2. Open Browser: Maximize the browser window.
3. Navigate to Web Page: Open the local HTML file `gcd.html`.
4. Locate Input Fields: Find the input fields for the two numbers (`num1 and num2`).
5. Locate Calculate Button: Find the "Calculate GCD" button.
6. Input Numbers: Enter 5 in the first input field and `6` in the second input field.
7. Click Calculate Button: Click the "Calculate GCD" button to calculate the GCD.
8. Wait for Result: Pause execution for 2 seconds using `Thread.sleep()`.
9. Get and Display Result: Find the result element and print the calculated GCD.
10. Handle Exception: Catch and print any `InterruptedException`.

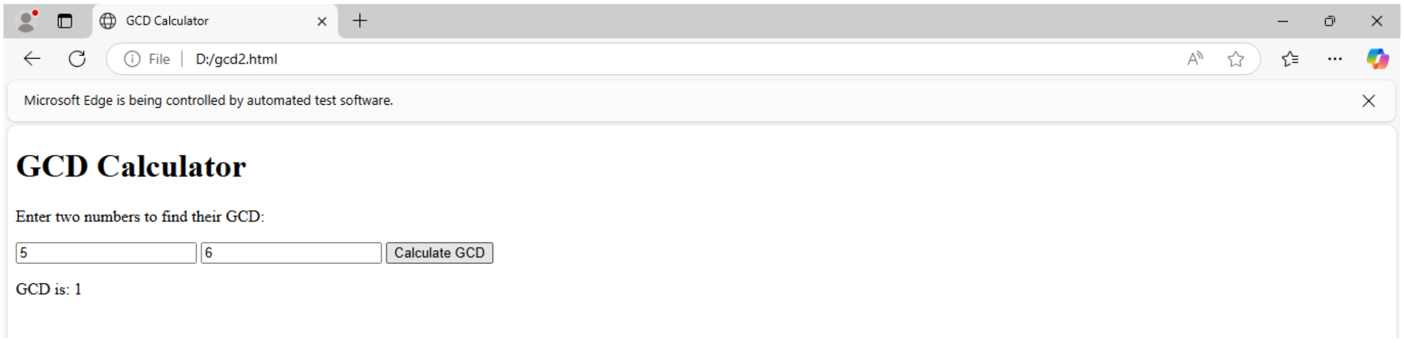
PROGRAM:

```
package demo;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.edge.EdgeDriver;

public class gcd {
    public static void main(String[] args) {
        System.setProperty("webdriver.edge.driver","C:\\edgedriver\\msedgedriver.exe");
        WebDriver driver = new EdgeDriver();
        driver.manage().window().maximize();

        try {
            driver.get("file:///D:/gcd.html");

            WebElement num1Input = driver.findElement(By.id("num1"));
            WebElement num2Input = driver.findElement(By.id("num2"));
            WebElement calculateButton =
                      driver.findElement(By.xpath("//button[text()='Calculate GCD']"));

            num1Input.sendKeys("5");
            num2Input.sendKeys("6");

            calculateButton.click();
            Thread.sleep(2000);

            WebElement result = driver.findElement(By.id("result"));
            System.out.println("Test Result: " + result.getText());
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

HTML CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>GCD Calculator</title>
<script>
    function calculateGCD() {
        var num1 = parseInt(document.getElementById("num1").value);
        var num2 = parseInt(document.getElementById("num2").value);

        while (num2 !== 0) {
            var temp = num2;
            num2 = num1 % num2;
            num1 = temp;
        }

        document.getElementById("result").innerHTML = "GCD is: " + num1;
    }
</script>
</head>
<body>
    <h1>GCD Calculator</h1>
    <p>Enter two numbers to find their GCD:</p>
    <input type="number" id="num1" placeholder="Enter first number" required>
    <input type="number" id="num2" placeholder="Enter second number" required>
    <button onclick="calculateGCD()">Calculate GCD</button>
    <p id="result"></p>
</body>
</html>
```

OUTPUT:



Microsoft Edge is being controlled by automated test software.

GCD Calculator

Enter two numbers to find their GCD:

5 6

GCD is: 1

```
Test Result: GCD is: 1
```

RESULT:

Thus, the Program to develop a web page using JavaScript which calculates the GCD of 2 numbers using Selenium tool has been executed successfully.

Ex. No: 12 **EXCEL TABLE UPDATION - STUDENT RECORDS**
Date:

AIM:

To write and test a program to update 10 student records into table into Excel file using selenium tool.

ALGORITHM:

1. Set Output File: Define the location where the Excel file will be saved.
2. Initialize Workbook: Create a new workbook and a sheet named "Report".
3. Define Formatting: Set up fonts and cell formatting for text and numbers.
4. Add Headers: Insert column headers (e.g., "Student Name", "Subject 1", "Subject 2", "Subject 3").
5. Add Data:
   - Loop through 10 rows (for 10 students).
   - For each row, add the student name and random subject scores (calculated as i*i + X).
6. Write to File: Write the data to the Excel file.
7. Close Workbook: Save and close the workbook.
8. Display Success Message: Print a message indicating the file was created.

PROGRAM CODE:

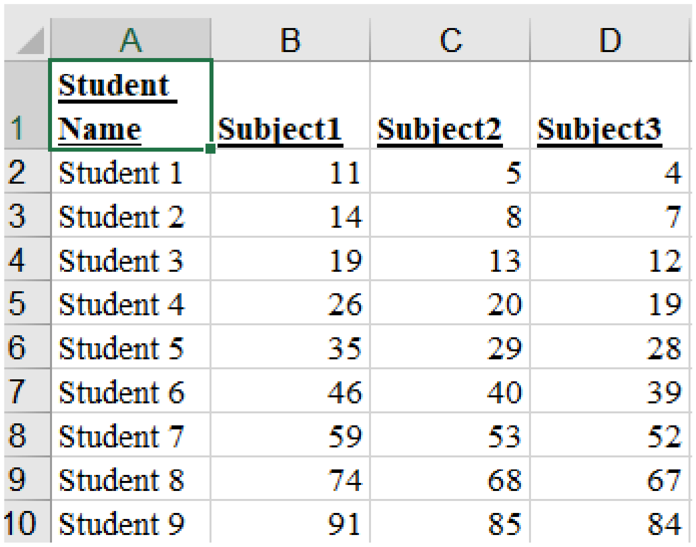
```
package demo;
import java.io.File;
import java.io.IOException;
import java.util.Locale;
import jxl.Workbook;
import jxl.WorkbookSettings;
import jxl.format.UnderlineStyle;
import jxl.write.Label;
import jxl.write.Number;
import jxl.write.WritableCellFormat;
import jxl.write.WritableFont;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;
import jxl.write.WriteException;
import jxl.write.biff.RowsExceededException;

public class student_excel_write {
    private WritableCellFormat timesBoldUnderline;
    private WritableCellFormat times;
    private String inputFile;

    public void setOutputFile(String inputFile) {
        this.inputFile = inputFile;
    }

    public void write() throws IOException, WriteException {
        File file = new File(inputFile);
        WorkbookSettings wbSettings = new WorkbookSettings();
        wbSettings.setLocale(new Locale("en", "EN"));

        WritableWorkbook workbook = Workbook.createWorkbook(file, wbSettings);
        WritableSheet excelSheet = workbook.createSheet("Report", 0);

        createLabel(excelSheet);
        createContent(excelSheet);
        workbook.write();
        workbook.close();
    }

    private void createLabel(WritableSheet sheet) throws WriteException {
        WritableFont times10pt = new WritableFont(WritableFont.TIMES, 10);
        times = new WritableCellFormat(times10pt);
        times.setWrap(true);

        WritableFont times10ptBoldUnderline = new WritableFont(
                                WritableFont.TIMES, 10, WritableFont.BOLD, false,
                                  UnderlineStyle.SINGLE);
        timesBoldUnderline = new WritableCellFormat(times10ptBoldUnderline);
        timesBoldUnderline.setWrap(true);

        // Adding Headers
        addCaption(sheet, 0, 0, "Student Name");
        addCaption(sheet, 1, 0, "Subject 1");
        addCaption(sheet, 2, 0, "Subject 2");
        addCaption(sheet, 3, 0, "Subject 3");
    }

    private void createContent(WritableSheet sheet) throws WriteException {
        for (int i = 1; i <= 10; i++) {
            addLabel(sheet, 0, i, "Student " + i);
            addNumber(sheet, 1, i, ((i * i) + 10));
            addNumber(sheet, 2, i, ((i * i) + 4));
            addNumber(sheet, 3, i, ((i * i) + 3));
        }
    }

    private void addCaption(WritableSheet sheet, int column, int row, String text) throws
                                                                      WriteException {
        Label label = new Label(column, row, text, timesBoldUnderline);
        sheet.addCell(label);
    }

    private void addNumber(WritableSheet sheet, int column, int row, Integer value) throws
                                                                      WriteException {
        Number number = new Number(column, row, value, times);
        sheet.addCell(number);
    }

    private void addLabel(WritableSheet sheet, int column, int row, String text) throws
                                                                      WriteException {
        Label label = new Label(column, row, text, times);
        sheet.addCell(label);
    }

    public static void main(String[] args) throws WriteException, IOException {
        student_excel_write test = new student_excel_write();
        test.setOutputFile("D:\\Student.xls");
        test.write();
        System.out.println("File generated: D:\\Student.xls");
    }
}
```

OUTPUT:

AFTER EXECUTION:

```
File generated: D:\Student.xls
```

EXCEL FILE AFTER UPDATE OPERATION:



	A	B	C	D
1	Student Name	Subject1	Subject2	Subject3
2	Student 1	11	5	4
3	Student 2	14	8	7
4	Student 3	19	13	12
5	Student 4	26	20	19
6	Student 5	35	29	28
7	Student 6	46	40	39
8	Student 7	59	53	52
9	Student 8	74	68	67
10	Student 9	91	85	84

RESULT:

Thus, the Program to update 10 student records into a table in an Excel file using Selenium tool has been executed successfully.