



**(Accredited with Grade 'A+' by NAAC)**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**LABORATORY MANUAL**

**FOR**

**22CSCP608 – SOFTWARE ENGINEERING LABORATORY**

## **LIST OF EXERCISES**

1. Write a C program for matrix multiplication to understand the causes of failures.
2. Write a C program for Binary Search - Path Testing.
3. Write a C program to derive test cases based on boundary value analysis.
4. Write a C program for cause effect graph to check whether defect is found in the program.
5. Write a C program to perform data flow testing for the given code and find out all d- use Pairs.
6. Write a C program to demonstrate the working of the looping constructs.
7. Write and test a program to count number of check boxes on the page checked and unchecked count using selenium tool.
8. Write and test a program to provide total number of objects available on the page using selenium tool.
9. Write and test a program to login a specific web page using selenium tool.
10. Write and test a program to select the number of students who have scored more than 60 in any one subject (or all subjects).
11. Write a Java script to develop a web page which calculates the GCD of 2 numbers using Selenium tool.
12. Write and test a program to update 10 student records into table into Excel file using selenium tool.

# **INTRODUCTION TO SOFTWARE TESTING**

The purpose of the laboratory is to evaluate and develop methods of testing software efficiently that aim on discovering security relevant software flaws along with considering core components of quality before the final product is deployed.

The prime goal is to make the students aware about the existing methods of software testing and considering software quality. Many of the software testing techniques available are very expensive and time consuming. Therefore, the aim of the lab is to understand, which existing testing techniques are most effective for vulnerability detection, in order to provide software engineers guidelines for the selection of testing methods using software quality methods.

## **LAB REQUIREMENTS:**

Software Detail

Dev C++,

Eclipse, Selenium Tool, Jxl, JUnit.


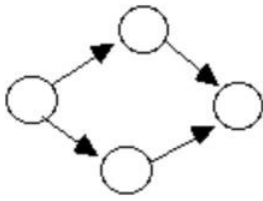
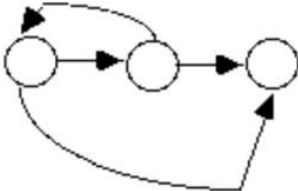
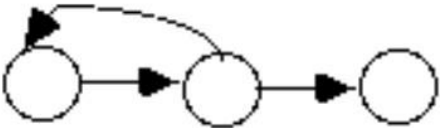
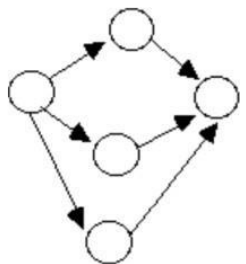
## 1) PATH TESTING:

The basis path method allows for the construction of test cases that are guaranteed to execute every statement in the program at least once. This method can be applied to detailed procedural design or source code.

### ***Method:***

- 1) Draw the flow graph corresponding to the procedural design or code.
- 2) Determine the cyclomatic complexity of the flow graph.
- 3) Determine the basis set of independent paths. (The cyclomatic complexity indicates the number of paths required.)
- 4) Determine a test case that will force the execution of each path.

### **Flow graphs:**

<b>Sequence</b>	
<b>IF</b>	
<b>WHILE</b>	
<b>REPEAT</b>	
<b>CASE</b>	

## Cyclomatic Complexity:

Cyclomatic complexity is a source code complexity measurement that is being correlated to a number of coding errors. It is calculated by developing a Control Flow Graph of the code that measures the number of linearly-independent paths through a program module.

Lower the Program's cyclomatic complexity, lower the risk to modify and easier to understand. It can be represented using the below formula:

$$\text{Cyclomatic complexity} = E - N + 2 * P$$

where,

E = number of edges in the flow graph.

N = number of nodes in the flow graph.

P = number of nodes that have exit points

### Example:

```
IF A = 10 THEN
```

```
IF B > C THEN
```

```
  A = B
```

```
ELSE
```

```
  A = C
```

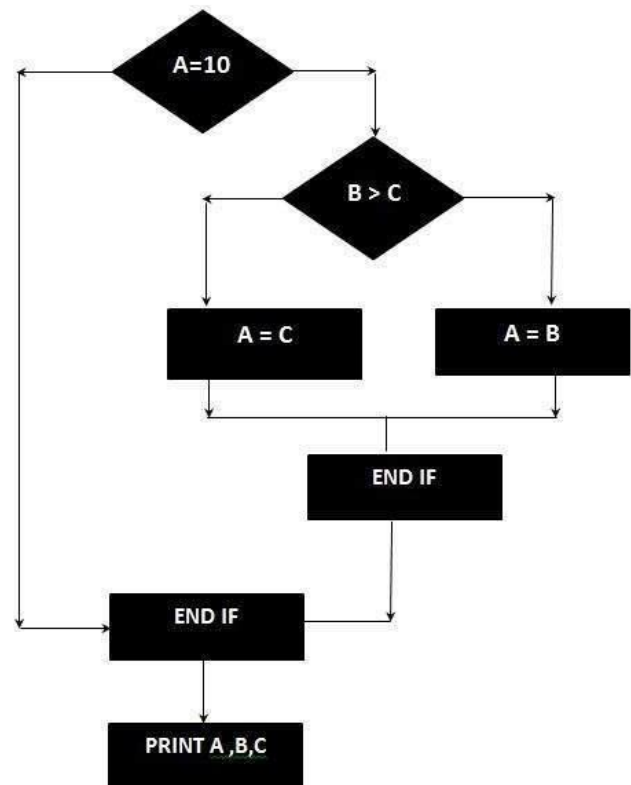
```
ENDIF
```

```
ENDIF
```

```
Print A
```

```
Print B
```

```
Print C
```



The Cyclomatic complexity is calculated using the above control flow diagram that shows seven nodes(shapes) and eight edges (lines), hence the cyclomatic complexity is  $8 - 7 + 2 = 3$ .

## 2) BOUNDARY VALUE ANALYSIS:

Boundary value analysis is complementary to equivalence partitioning. Rather than selecting arbitrary input values to partition the equivalence class, the test case designer chooses values at the extremes of the class. Furthermore, boundary value analysis also encourages test case designers to look at output conditions and design test cases for the extreme conditions in output.

### Guidelines for boundary value analysis:

- 1) If an input condition specifies a range bounded by values a and b, test cases should be designed with values a and b, and values just above and just below a and b.
- 2) If an input condition specifies a number of values, test cases should be developed that exercise the minimum and maximum numbers. Values above and below the minimum and maximum are also tested.
- 3) Apply the above guidelines to output conditions. For example, if the requirement specifies the production of a table as output, then you want to choose input conditions that produce the largest and smallest possible table.
- 4) For internal data structures, be certain to design test cases to exercise the data structure at its boundary. For example, if the software includes the maintenance of a personnel list, then you should ensure the software is tested with conditions where the list size is 0, 1 and maximum (if constrained).



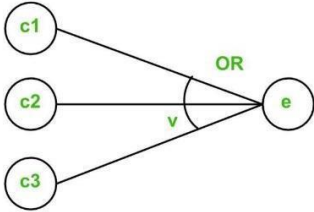
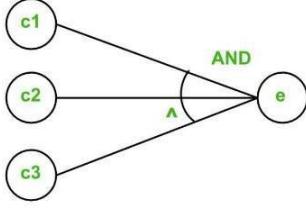
Data Item	Input Condition	Remarks
Area Code	range	Values between 200 and 999 with area codes requiring 0, 1 in second position Test Cases: 200, 910 (valid end-points) 199, 912 (value below and above)
Prefix	range	Specified value > 200 Test Cases: 201 (minimum) 999 (maximum) 200 (invalid, just below) 199, 912 (value below and above)
Prefix	range	Specified value > 200 Test Cases: 201 (minimum) 999 (maximum) 200 (invalid, just below)
Suffix	value	Four-digit length Test Cases: 0000 (minimum) 9999 (maximum)
Password	value	Six-character string Test Cases:
Commands	set	Test Cases:

### 3) CAUSE-EFFECT GRAPHS:

A weakness of the two methods is that do not consider potential combinations of input/output conditions. Cause-effect graphs connect input classes (causes) to output classes (effects) yielding a directed graph.

#### Guidelines for cause-effect graphs:

- 1) Decompose the specification into workable pieces.
- 2) Identify causes and their effects.
- 3) Create a (Boolean) cause-effect graph (special symbols are required).
- 4) Annotate the graph with constraints describing combinations of causes and /or effects that are impossible.
- 5) The graph is converted to a decision table by methodically tracing state conditions in the graph. Each column in the table represents a test case.
- 6) Decision table rules are converted to test cases.

Identify	
Not	
Or	
And	

### 4) DATA FLOW TESTING:

Data Flow Testing is a type of structural testing. It is a method that is used to find the test paths of a program according to the locations of definitions and uses of variables in the program. It has nothing to do with data flow diagrams.

It is concerned with:

- Statements where variables receive values,
- Statements where these values are used or referenced.

To illustrate the approach of data flow testing, assume that each statement in the program assigned a unique statement number. For a statement number S-

$DEF(S) = \{X \mid \text{statement } S \text{ contains the definition of } X\}$

$USE(S) = \{X \mid \text{statement } S \text{ contains the use of } X\}$

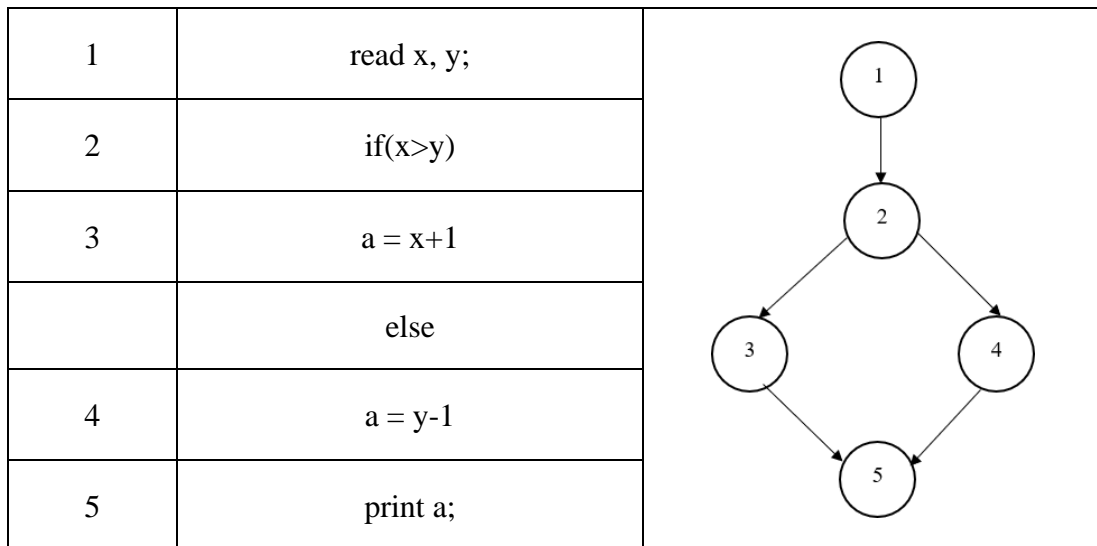
If a statement is a loop or if condition then its DEF set is empty and USE set is based on the condition of statement s.

Data Flow Testing uses the control flow graph to find the situations that can interrupt the flow of the program.

Reference or define anomalies in the flow of the data are detected at the time of associations between values and variables. These anomalies are:

- A variable is defined but not used or referenced,
- A variable is used but never defined,
- A variable is defined twice before it is used

**EXAMPLE:**



**Define/use of variables of above example:**

Variable	Defined at node	Used at node
X	1	2,3
Y	1	2,4
a	3,4	5



Data flow testing is a family of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of variables or data objects. Dataflow Testing focuses on the points at which variables receive values and the points at which these values are used.

- A node  $n$  in the program graph is a **defining** node for variable  $v$  – **DEF**( $v, n$ ) – if the value of  $v$  is defined at the statement fragment in that node.
  - ❖ *Input, assignment, procedure calls*
- A node  $n$  in the program graph is a **usage** node for variable  $v$  – **USE**( $v, n$ ) – if the value of  $v$  is used at the statement fragment in that node.
  - ❖ *Output, assignment, conditionals*
- A usage node is a **predicate use, P-use**, if variable  $v$  appears in a predicate expression.
  - ❖ *Always in nodes with outdegree  $\geq 2$*
- A usage node is a **computation use, C-use**, if variable  $v$  appears in a computation.
  - ❖ *Always in nodes with outdegree  $\leq 1$*
- A node  $n$  in the program is a **kill** node for a variable  $v$  – **KILL**( $v, n$ ) – if the variable is deallocated at the statement fragment in that node.

#### du-path:

- A **definition-use path, du-path**, with respect to a variable  $v$  is a path whose first node is a defining node for  $v$ , and its last node is a usage node for  $v$ .

#### dc-path:

- A **du-path** with no other defining node for  $v$  is a **definition-clear path, dc-path**.

### 5) LOOP TESTING:

Simple Loops	<p>The following set of tests should be applied to simple loops, where <math>n</math> is the maximum number of allowable passes:</p> <ol style="list-style-type: none"> <li>1. Skip the loop entirely.</li> <li>2. Only one pass through the loop.</li> <li>3. Two passes through the loop.</li> <li>4. <math>m</math> passes through the loop where <math>m &lt; n</math>.</li> <li>5. <math>n-1, n, n+1</math> passes through the loop.</li> </ol>
Nested Loops	<ol style="list-style-type: none"> <li>1. Start with the innermost loop. Set all other loops to minimum values.</li> <li>2. Conduct simple loop tests for the innermost loop while holding the outer loops at their minimum iteration values.</li> <li>3. Work outward, conducting tests for the next loop, but keeping all other outer loops at this minimum iteration count.</li> <li>4. Continue until all loop have been tested.</li> </ol>
Concatenated Loops	<p>Concatenated loops can be tested using the approach defined for simple loops, if the loops are independent. If the loop counter from a loop <math>i</math> is used as the initial value for loop <math>i+1</math> then the loops are not independent. When loops are not independent use the concatenated loop strategy.</p>
Concatenated Loops	<p>Redesign the loops so they are one of the above categories.</p>

# SELENIUM

## AUTOMATION:

Automation is making a process automatic, eliminating the need for human intervention. It is a self-controlling or self-moving process. Automation Software offers automation wizards and commands of its own in addition to providing a task recording and re-play capabilities. Using these programs, you can record an IT or business task.

### Benefits of Automation:

- Speed
- Reliability
- Repeatability
- Programmability
- Reusability
- Simplifies Regression Testing
- Enables Continuous Testing

## Introduction to Selenium:

### History of Selenium:

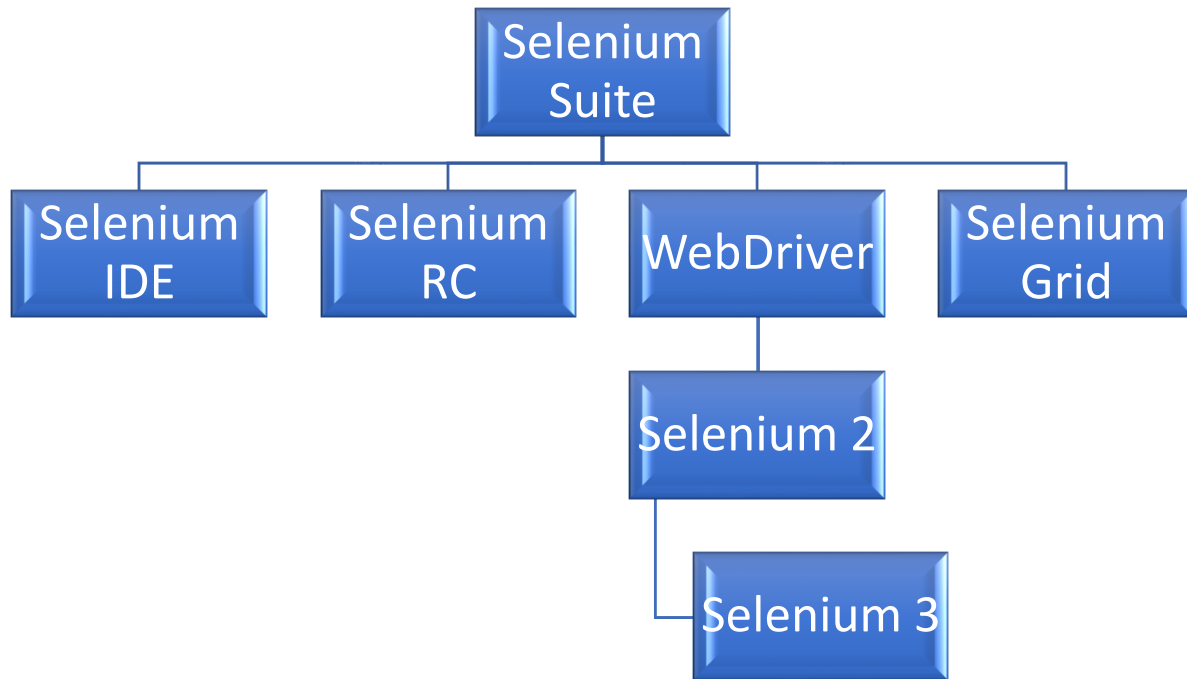
- **2004: Selenium** was created by **Jason R. Huggins** and his team at **ThoughtWorks**.
- Initially named **JavaScript Functional Tester (JSFT)**, it was developed as a tool to automate testing of web applications.
- **2006: Selenium RC (Remote Control)** was introduced, allowing developers to write tests in any programming language by controlling browsers remotely.
- **2011: Selenium WebDriver** was introduced, offering a more efficient and reliable alternative to Selenium RC, providing better support for modern web browsers and features.
- **2018: Selenium 4** was introduced, featuring better W3C WebDriver standard compliance, improved support for modern web applications, and enhanced features like improved grid capabilities and a new documentation system.

### What is Selenium?

Selenium is an acceptance testing tool for web applications. It enables users to run tests directly in the browser, which provides an accurate representation of how a user would interact with the application. Selenium is cross-platform, supporting deployment on Windows, Linux, and Macintosh systems. The tool is implemented using web technologies such as JavaScript, DHTML, and Frames, making it highly adaptable and compatible with modern web applications.

### Selenium Components:

- **Selenium Integrated Development Environment (IDE)** – A record-and-playback tool for creating tests without writing code.
- **Selenium Remote Control (RC)** – Allows users to control a browser remotely to perform tests.
- **WebDriver** – A more advanced tool for controlling browsers that provides a programming interface to interact with the browser directly.
- **Selenium Grid** – Enables parallel test execution across multiple machines and browsers, allowing for efficient and scalable testing.



## INSTALLATION OF JAVA RELATED SOFTWARES:

Software installation is a 6-step process:

- Install Java SDK
- Install Eclipse
- Install Selenium
- Install EdgeDriver
- Install Jxl V2.6
- Install JUnit V4

### Step 1: Install Java SDK (JDK)

Before installing Selenium and the necessary tools, you need to install the Java Software Development Kit (JDK).

#### 1.1 Download and Install JDK

- Go to the Oracle JDK Downloads page.
- Download the JDK version appropriate for your operating system (Windows/macOS/Linux).
- Follow the installation instructions for your operating system.

#### 1.2 Set Java Environment Variables

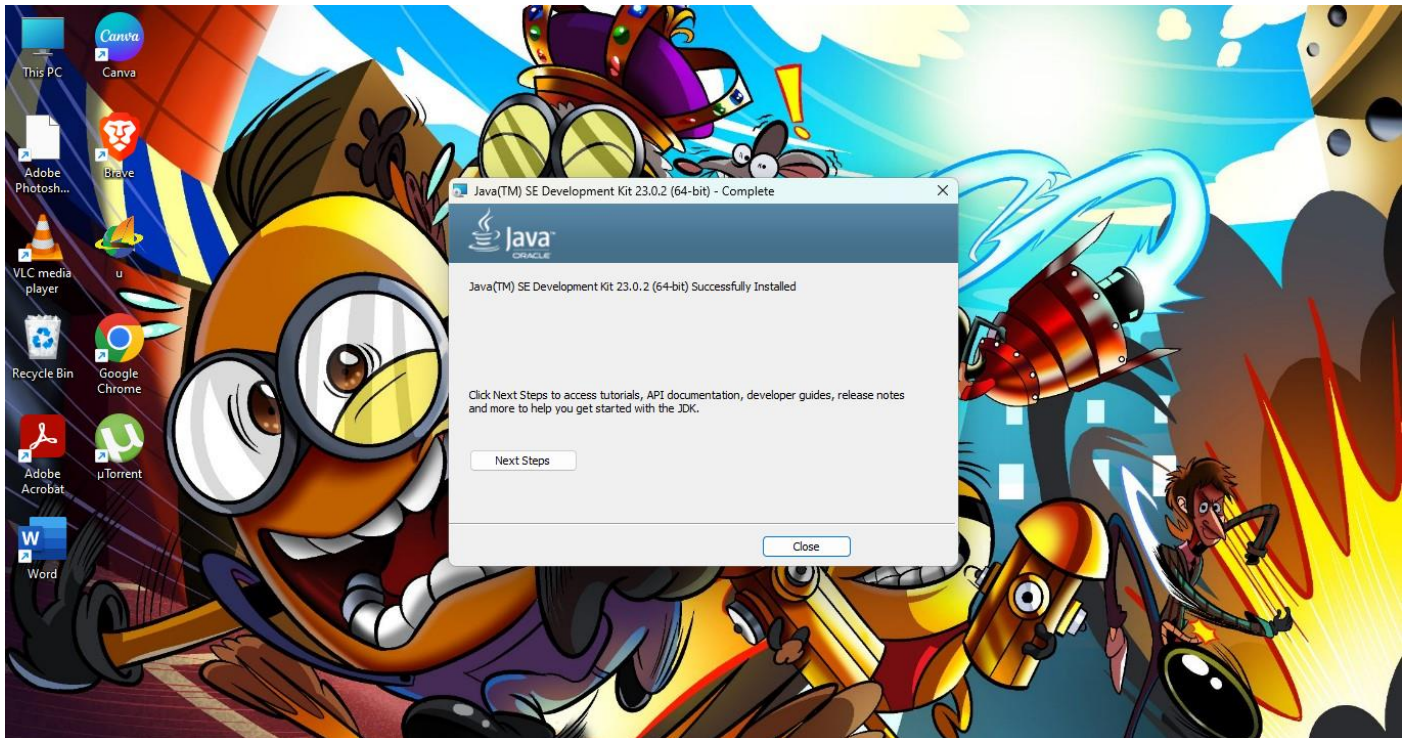
On Windows:

Set the JAVA\_HOME environment variable by going to **Control Panel > System and Security > System > Advanced system settings > Environment Variables**.

Then add:

Variable Name: JAVA\_HOME

Variable Value: Path to your JDK folder (e.g., **C:\Program Files\Java\jdk-23**).

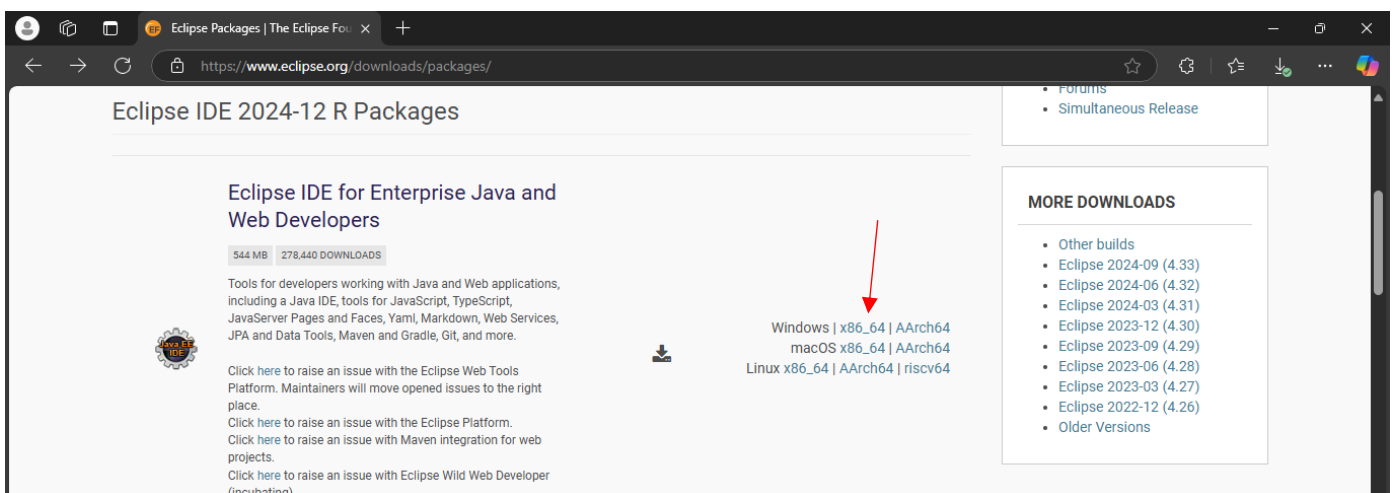


## Step 2: Install Eclipse IDE

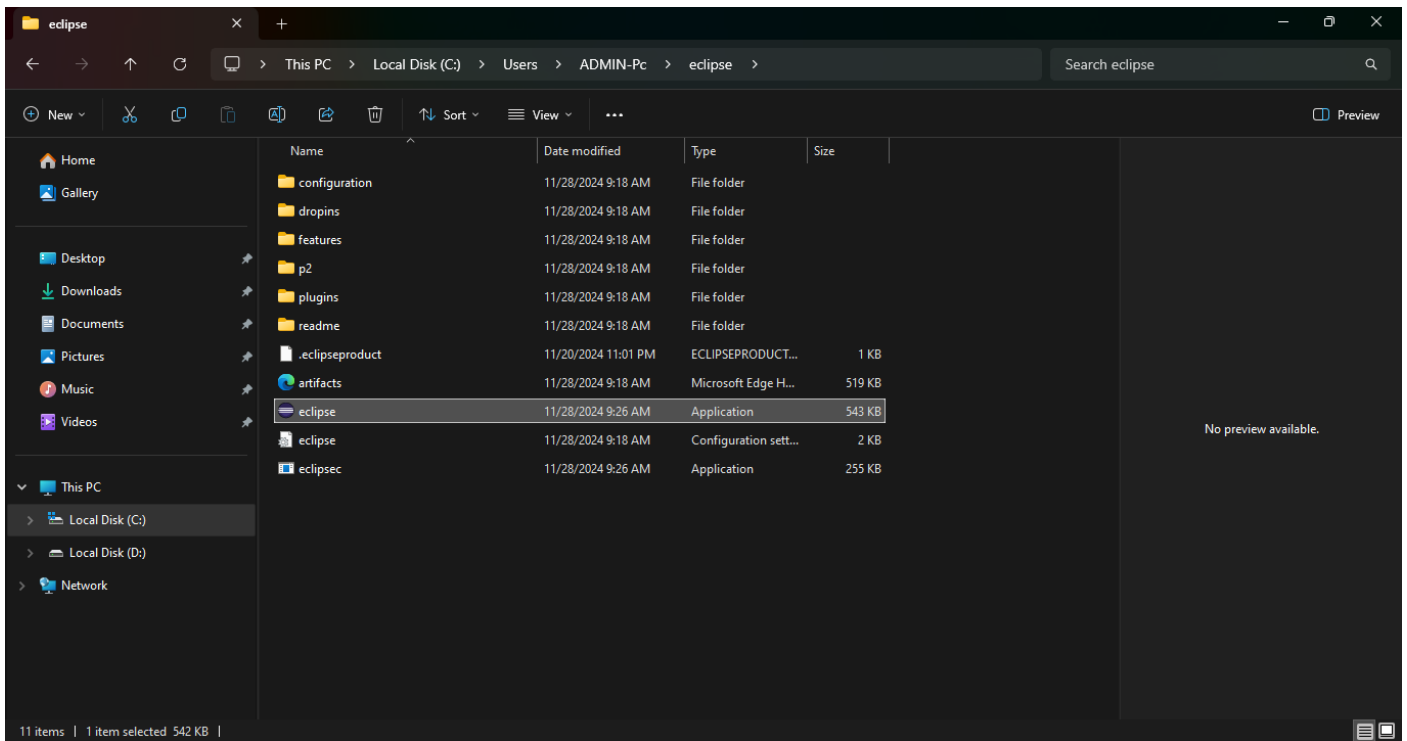
Eclipse is an Integrated Development Environment (IDE) used for Java development.

### 2.1 Download Eclipse

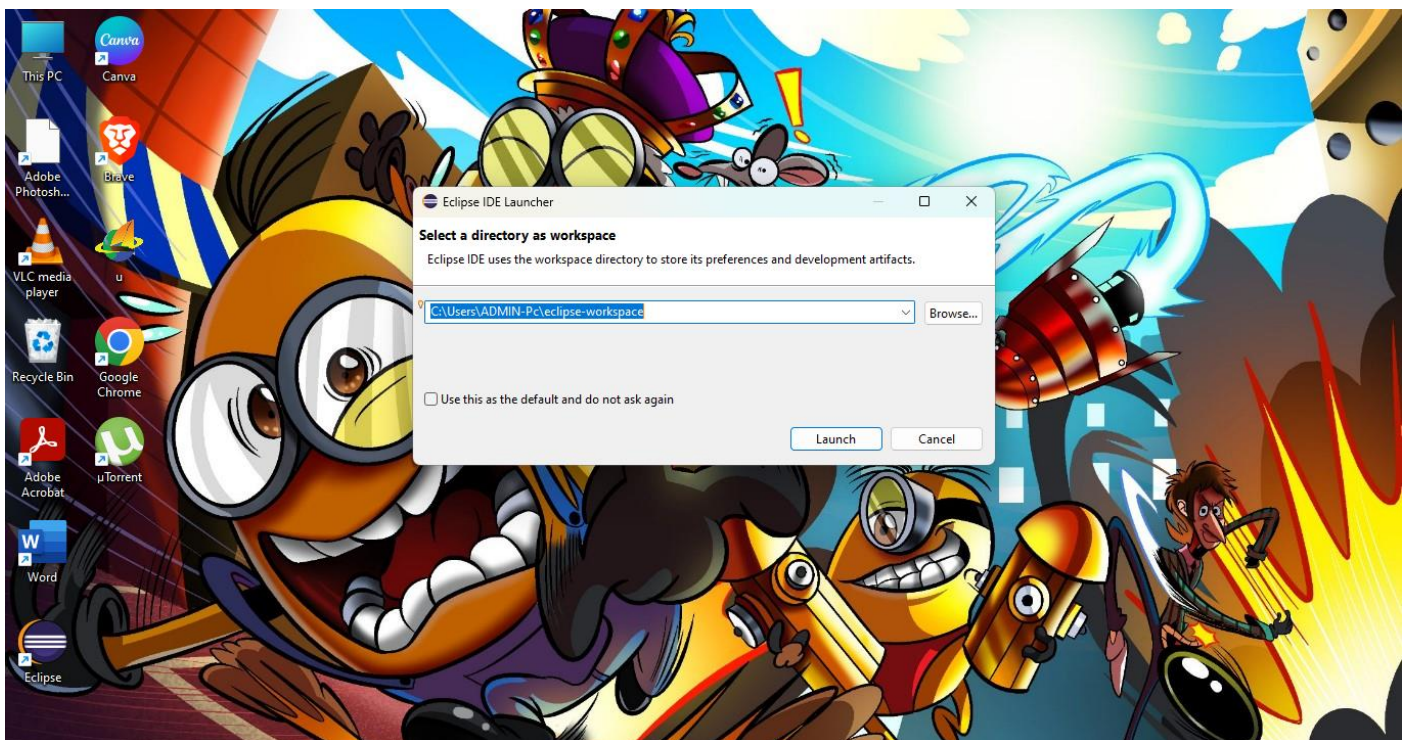
- Go to the official Eclipse download page: [Eclipse Downloads](https://www.eclipse.org/downloads/packages/).
- Download Eclipse IDE for Java Developers in the .zip file.
- Once the download is complete, extract the zipped file and move it into the Users folder. Save it as '**eclipse**' in **C:\Users\ADMIN-pc**.
- Open the '**eclipse**' folder and launch the **eclipse.exe** file.



Open the '**eclipse**' folder and launch the **eclipse.exe** file.



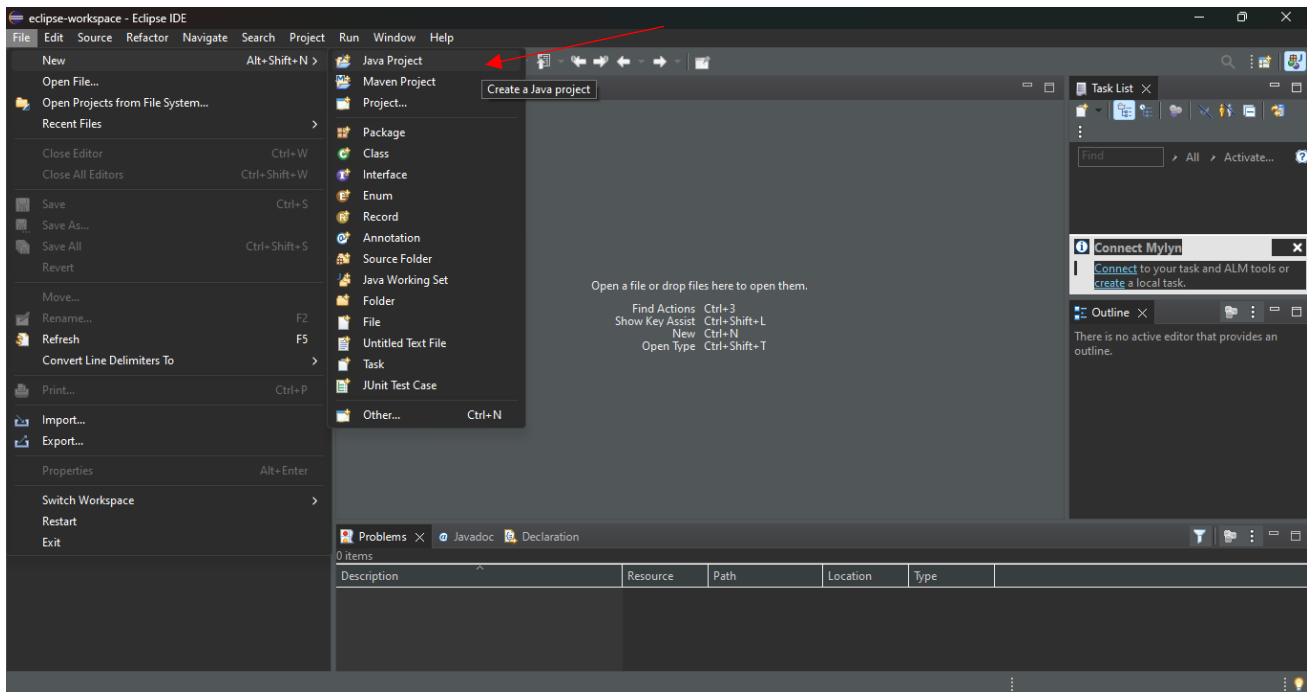
## 2.2 Launch Eclipse and create the Workspace





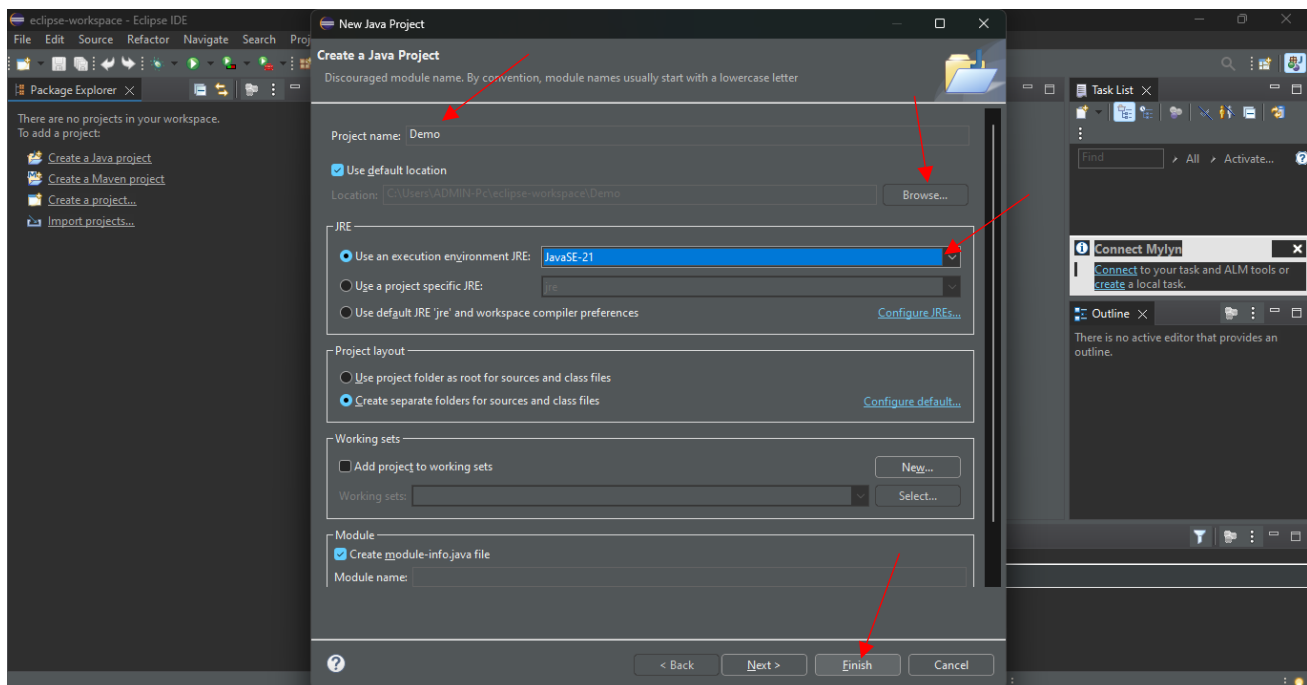
## 2.3 Create a New Java Project

Create a new project by navigating to **File > New > Java Project**. Name of the project yourwise.



A new pop-up window will open. Enter the details as follows:

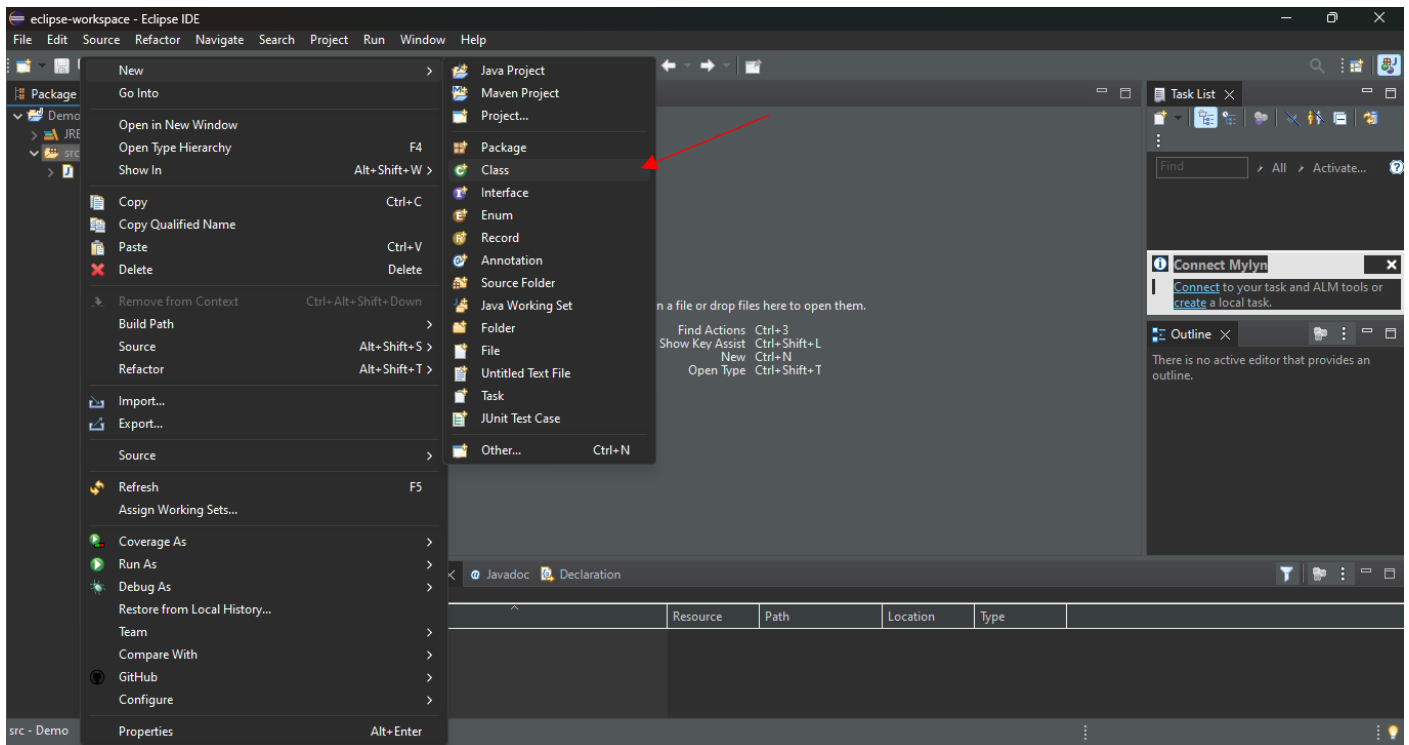
- Project Name
- Location to save the project
- Select an execution JRE
- Select the project layout option
- Click the Finish button



Now your Java Project was created.

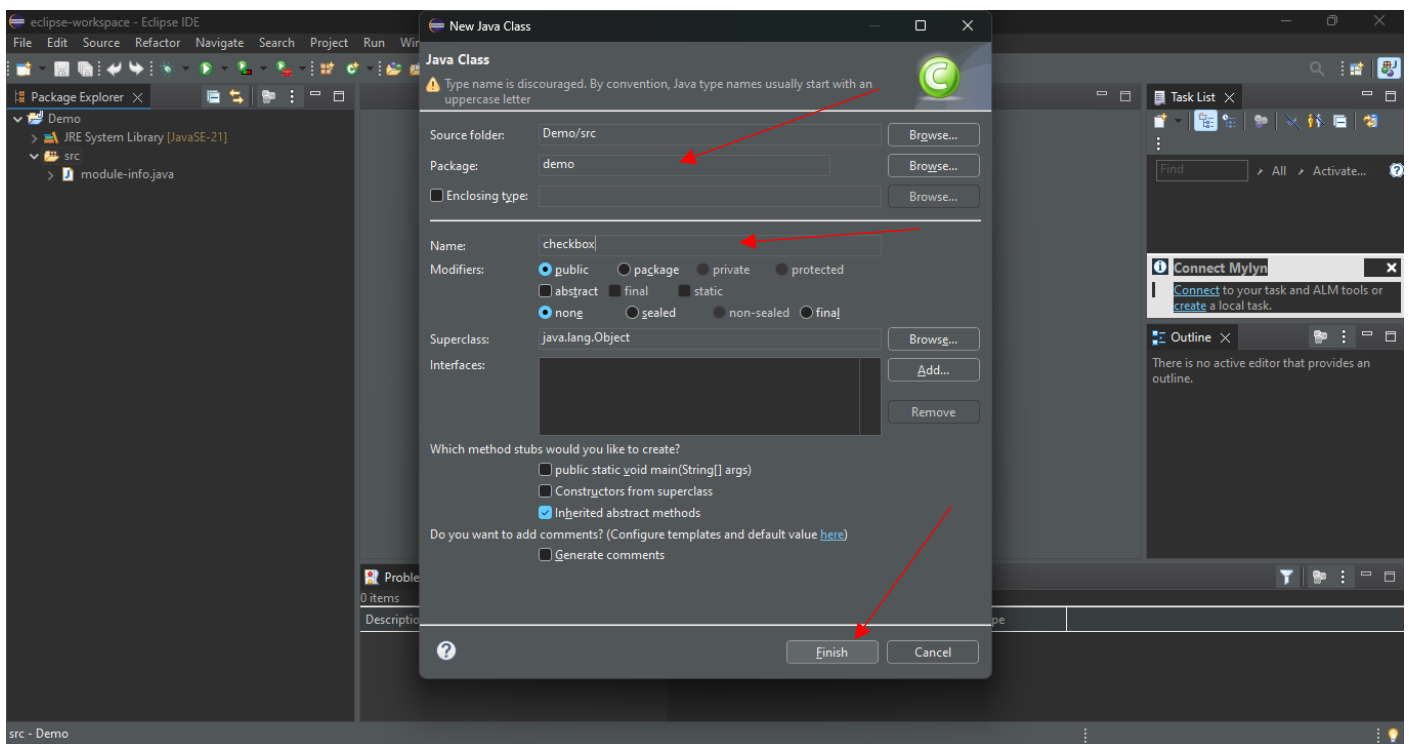
## 2.4 Create a New Java Class

Create a new project by navigating to **File > New > Class**. Name of the Class yourwise.



A new pop-up window will open. Enter the details as follows:

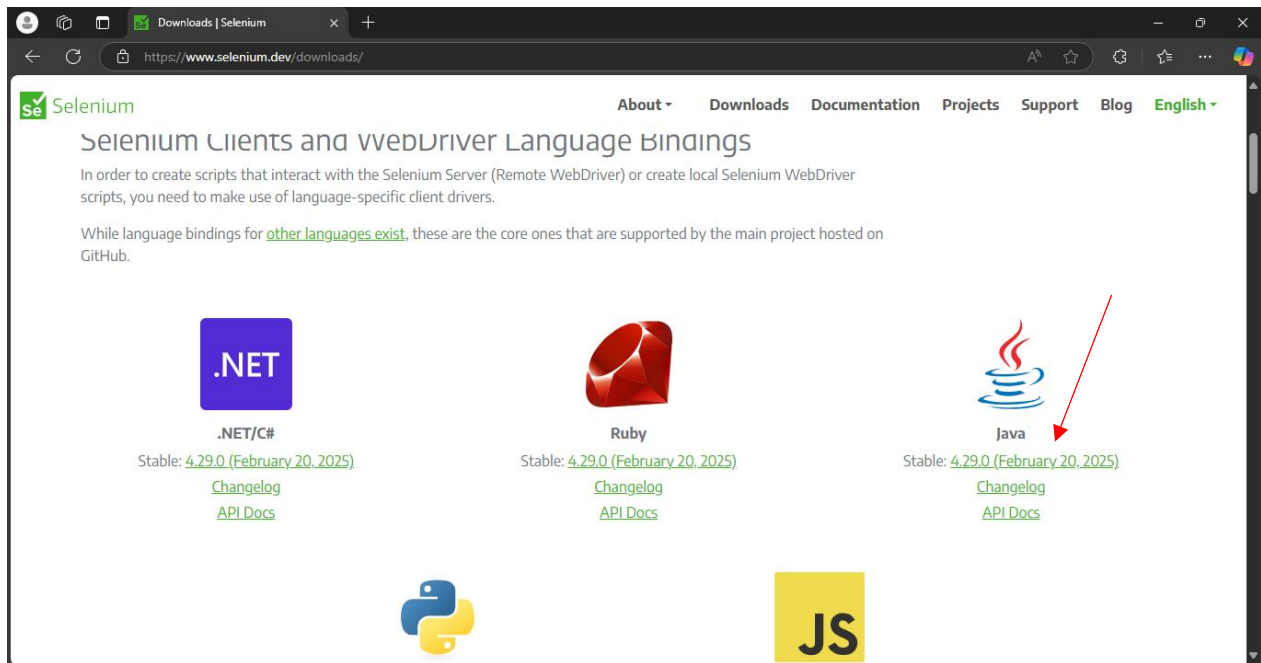
- Package Name
- Name of the Class
- Click the Finish button



## Step 3: Install Selenium

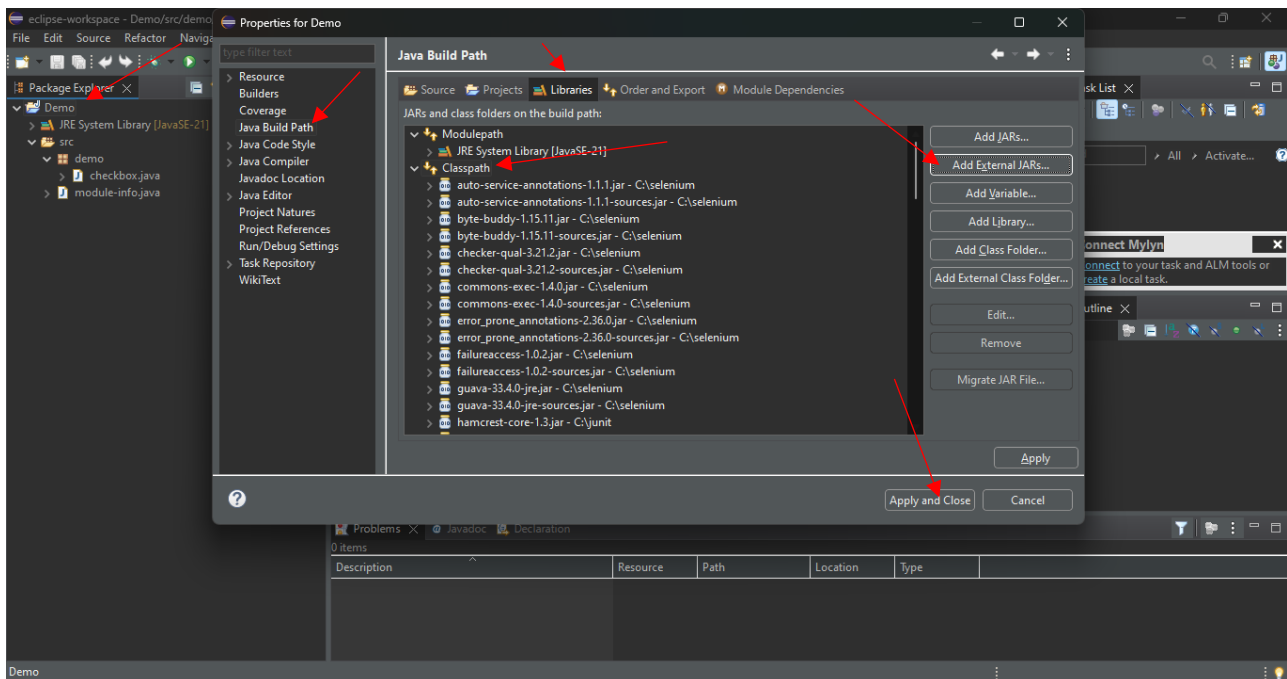
### 3.1 Download Selenium WebDriver for Java

- Visit the official Selenium website: [Selenium Downloads](https://www.selenium.dev/downloads/).
- Under Selenium Client & WebDriver Language Bindings, select Java and download the latest version of Selenium as a **.zip** file.
- Extract the **.zip** file to a folder on your computer (e.g., **C:\selenium**).



### 3.2 Add Selenium to Eclipse

- Right-click on your project and select **Properties**.
- Go to **Java Build Path > Libraries > Click the Classpath > Add External JARs**.
- Navigate to the folder where you extracted Selenium and add all the .jar files from the libs folder.



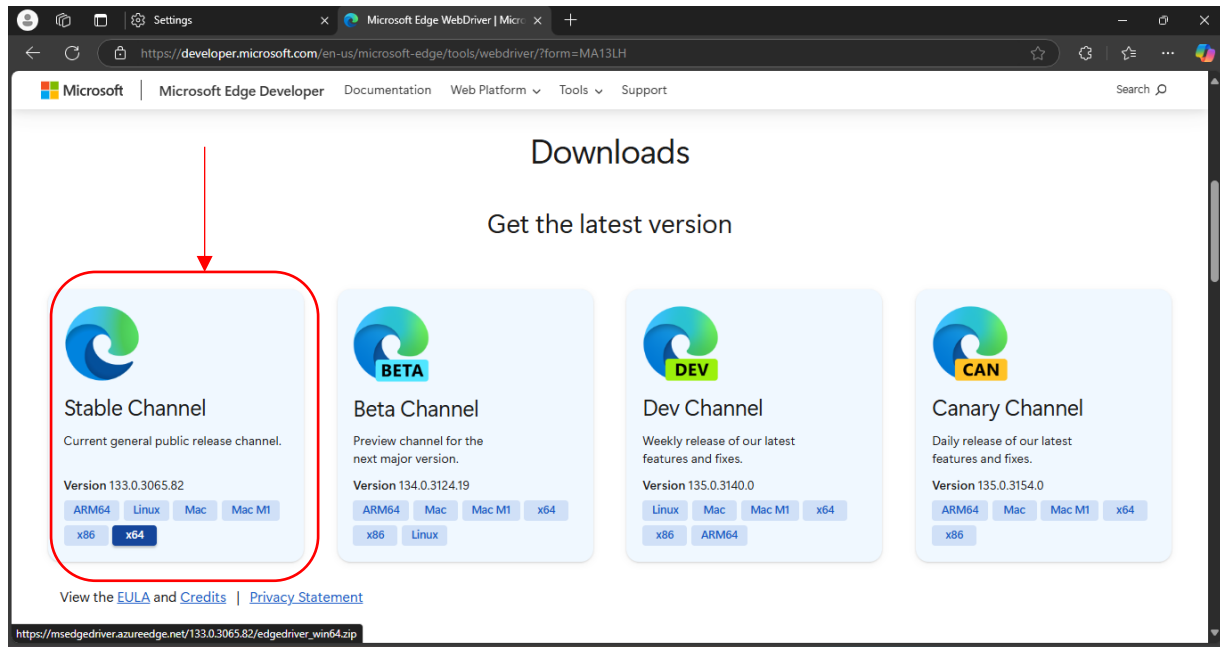


## Step 4: Install EdgeDriver

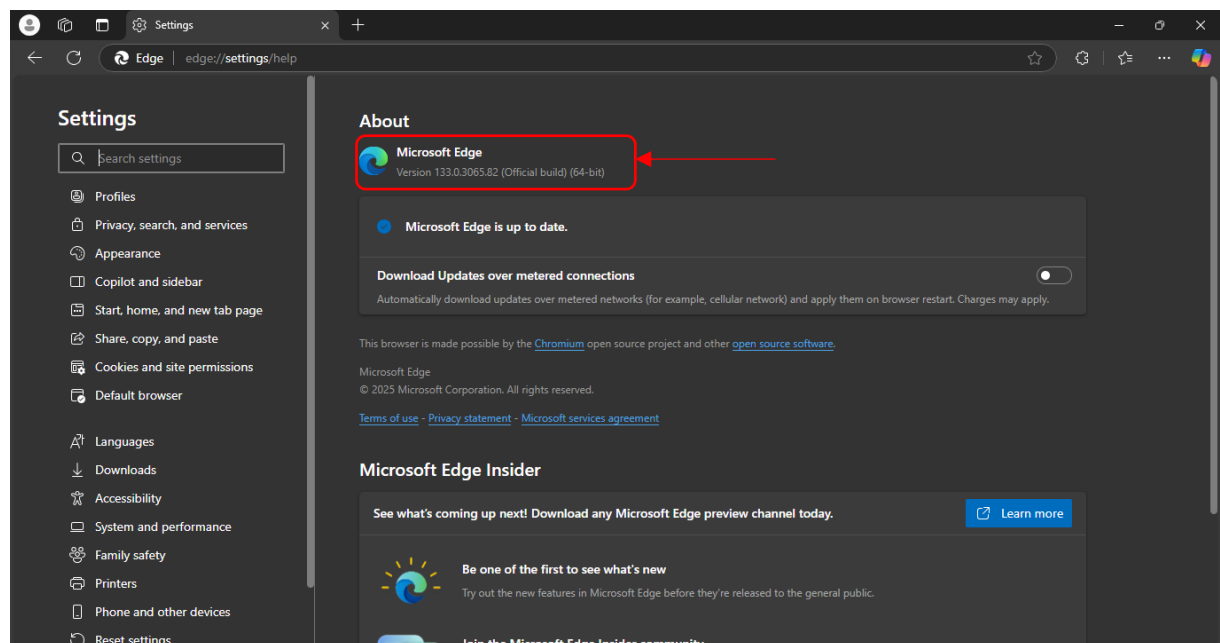
To run Selenium on Microsoft Edge, you need to install EdgeDriver.

### 4.1 Download EdgeDriver

- Go to the official Microsoft Edge WebDriver download page: [EdgeDriver Download](https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/).



- Select the version of **EdgeDriver** that matches your version of Microsoft Edge (check your browser version in Edge by going to **Settings > About Microsoft Edge**).



- Download the appropriate version for your operating system.
- If you have any Error Message while Execute the Program, check the Browser and Driver in the same version. Else Update the both up to date.
- This same process for the other Browser also like Google Chrome, Firefox, Safari.

## Step 5: Install JXL 2.6 (Java Excel API)

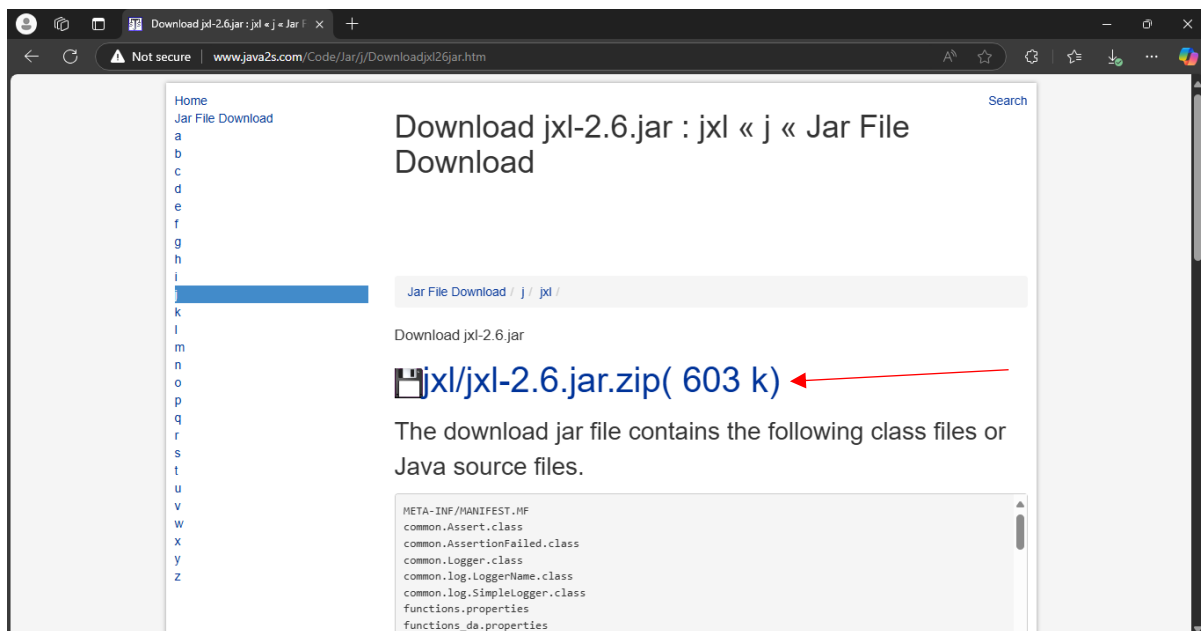
JXL (Java Excel API) allows you to work with Excel files in Java.

### 5.1 Download JXL 2.6

- Visit the official JXL download page: [JXL 2.6](#).
- Download the **JXL.jar** file.

### 5.2 Add JXL to Eclipse

- Open Eclipse.
- Right-click on your project and select **Properties**.
- Go to **Java Build Path > Libraries > Add External JARs**.
- Select the **JXL.jar** file you downloaded and click OK.



## Step 6: Install JUnit 4

JUnit 4 is a popular testing framework for Java, often used with Selenium for writing automated tests.

### 6.1 Download JUnit 4

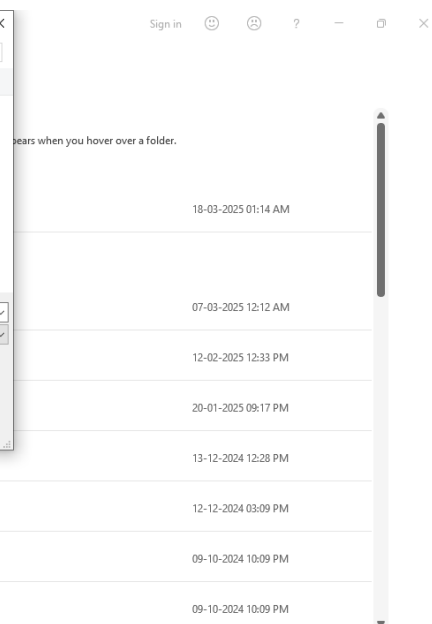
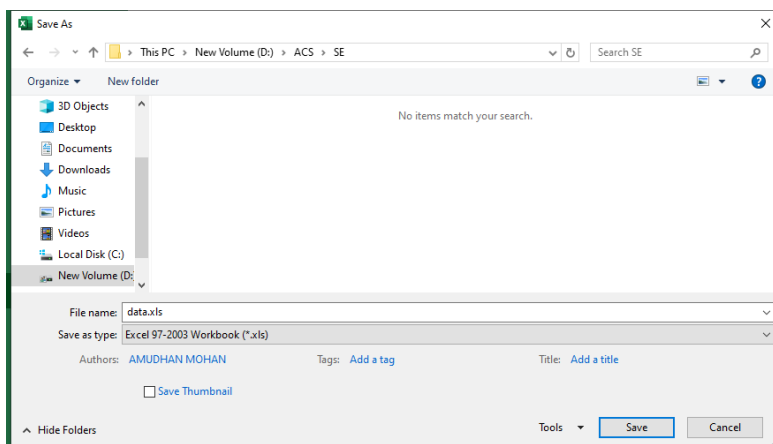
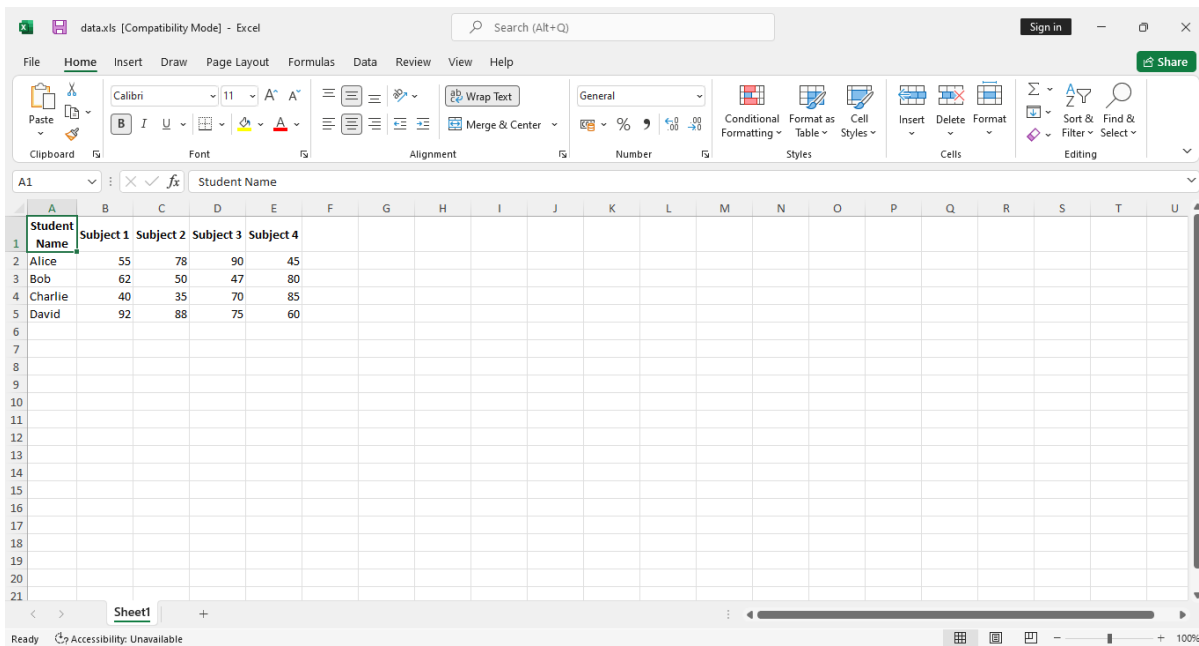
- Visit the official JUnit GitHub Wiki page: [JUnit 4 Download](#)
- Download the Plain-old JAR (junit.jar, hamcrest-core.jar) files.

### 6.2 Add JUnit to Eclipse

- Open Eclipse.
- Right-click on your project and select Properties.
- Go to Java Build Path > Libraries > Add External JARs.
- Select the junit.jar, hamcrest-core.jar files and click OK.

## Create an Excel File:

- 1) **Open the Excel Software** - Ensure that Excel is installed and updated on your device.
- 2) **Start a Blank Worksheet** - After launching Excel, click on **Blank workbook** to open a new sheet.
- 3) **Enter Data into the Sheet** - Type your data directly into the cells. If there are issues during data entry, like formatting errors or unresponsive cells, let me know.
- 4) **Save the File** - Here are the detailed steps:
  - Go to the menu: File > Save As.
  - Enter a filename in the "**data**" field.
  - Select a path where you want to save the file.
  - Choose the format: **Excel 97-2003 Workbook (.xls)\***.
  - Click Save.



**AIM:**

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

**ALGORITHM:**

- 1) Input Dimensions:

Read the dimensions of matrices **A** (**p x q**) and **B** (**r x s**).

- 2) Check Compatibility:

If **q** **!=** **r**, print "Matrix multiplication is not possible" and stop.

- 3) Input Matrices:

Read elements of matrices A and B.

- 4) Matrix Multiplication:

For each element **C[i][j]** in the result matrix:

**C[i][j] += A[i][k] \* B[k][j].**

- 5) Output Result:

Print the result matrix C.

**Possible reasons for Matrix multiplication failure:**

Failure case: 1: The main reason is for failure is, the number of columns in first matrix is not equal to the number of rows in second matrix.

Failure case:2: One or more values entered is/are not an integer.

**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

**AIM:**

2. To write a C program for binary search – Path Testing

**ALGORITHM:****Binary Search:**

Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

**Path Testing:**

- The objective of path testing is to ensure that the set of test cases is such that each path through the program is executed at least once.
- The starting point for path testing is a program flow graph that shows nodes representing program decisions and arcs representing the flow of control.
- Statements with conditions are therefore nodes in the flow graph.
- Cyclomatic complexity = Number of edges - Number of nodes +2.

**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);
    }
}
}

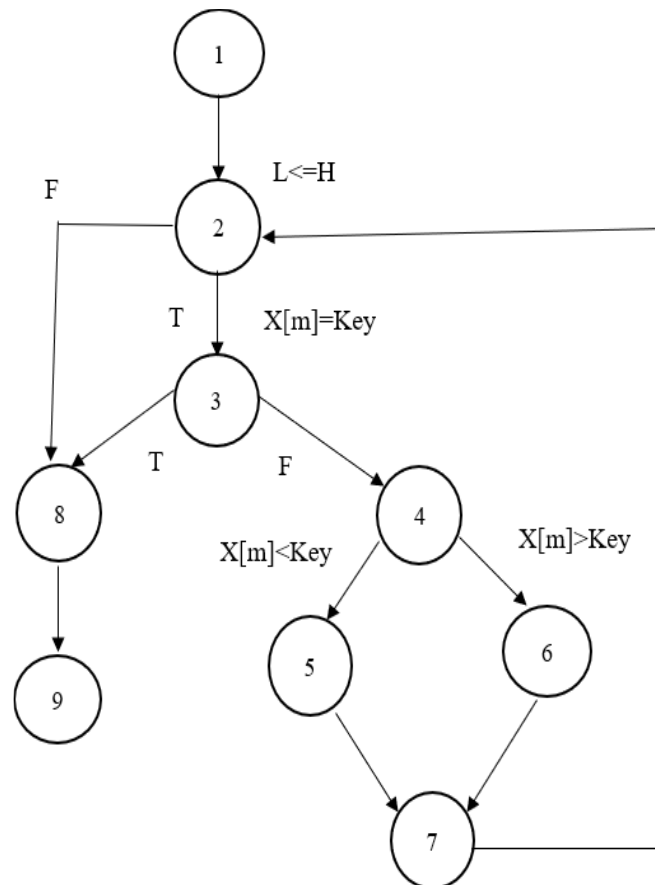
```

#### **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:



### Independent Paths:

#Edges = 11

#Nodes = 9

#P = 1

$V(G) = E - N + 2P$

$= 11 - 9 + 2$

$= 4$

**P1:** 1-2-3-8-9

**P2:** 1-2-3-4-5-7-2

**P3:** 1-2-3-4-6-7-2

**P4:** 1-2-8-9

**Pre-Conditions / Issues:**

- Array has elements in Ascending order: T/F
- Key element is in the Array: T/F
- Array has ODD number of elements: T/F

**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 $\in$ 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 $\notin$ X[ ]

## AIM:

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

## ALGORITHM:

**Our objective is to design the boundary value test cases:**

Boundary value analysis is a software testing technique in which tests are designed to include representatives of boundary values in a range. A boundary value analysis has a total of  $4*n+1$  distinct test cases, where  $n$  is the number of variables in a problem.

Here we have to consider all the three variables and design all the distinct possible test cases. We will have a total of 13 test cases as  $n = 3$ .

**Quadratic equation will be of type:  $ax^2+bx+c=0$**

Roots are **real** if  $(b^2 - 4ac) > 0$

Roots are **imaginary** if  $(b^2 - 4ac) < 0$

Roots are **equal** if  $(b^2 - 4ac) = 0$

Equation is **not quadratic** if  $a = 0$

## Designing the test cases:

For each variable we consider below 5 cases:

$$a_{\min} = 0$$

$$a_{\min+1} = 1$$

$$a_{\text{nominal}} = 50$$

$$a_{\max-1} = 99$$

$$a_{\max} = 100$$

When we are considering these 5 cases for a variable, rest of the variables have the nominal values, like in the above case where the value of 'a' is varying from 0 to 100, the value of 'b' and 'c' will be taken as the nominal or average value. Similarly, when the values of variable 'b' are changing from 0 to 100, the values of 'a' and 'c' will be nominal or average i.e., 50.

## PROGRAM:

```
#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

**AIM:**

4. Write a C program for cause effect graph to check whether defect is found in the program.

**ALGORITHM:**

Cause-Effect graph is a testing technique that aids in choosing test cases that logically relate Causes (inputs) to Effects (outputs) to produce test cases.

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

**STEP: 2:** Build up a cause-effect graph

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

**STEP: 4:** 11 test cases according to the 11 rules.

**PROGRAM:**

```
#include <stdio.h>

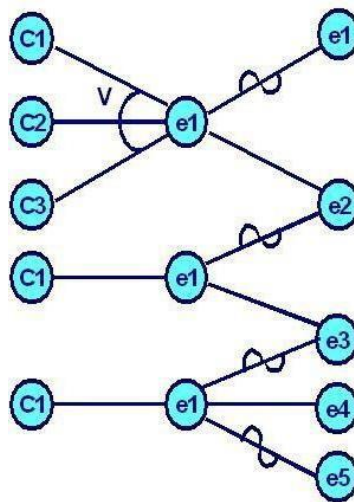
void main() {
    int a, b, c;
    printf("Enter the values of a, b, and c: ");
    if (scanf("%d %d %d", &a, &b, &c) != 3) {
        printf("\nImpossible\n");
        return;
    }
    if (((a + b) > c) && ((b + c) > a) && ((c + a) > b)) {
        if ((a == b) && (b == c)) {
            printf("\nIt 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			



**STEP: 4:** 11 test cases according to the 11 rules.

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

**AIM:**

5. Write a C program to perform data flow testing for the given code and find out all d- use Pairs.

**ALGORITHM:****Dynamic Data Flow Testing:**

Dynamic data-flow testing is performed with the intention to uncover possible bugs in data usage during the execution of the code. The test cases are designed in such a way that every definition of data variable to each of its use is traced and every use is traced to each of its definition. Various strategies are employed for the creation of test cases. All these strategies are defined below.

**All-du Paths (ADUP):**

It states that every du-path from every definition of every variable to every use of that definition should be exercised under some test. It is the strongest data flow testing strategy since it is a superset of all other data flow testing strategies. Moreover, this strategy requires the maximum number of paths for testing.

**DU-PATHS:**

In a path segment if the variable is defined earlier and the last link has a computational use of that variable then that path is termed as du path.

A sub-path in the flow is defined to go from a point where a variable is “defined”, to a point where it is “referenced, that is, where it is “used” – whatever kind of usage it is. Such a sub- path is called a “definition-use pair” or “du-pair”. The pair is made up of a “definition” of a variable and a “use” of the variable,

**DC-PATHS:**

- A path  $(i, n_1, \dots, n_m, j)$  is called a *definition-clear path* with respect to  $x$  from node  $i$  to node  $j$  if it contains no definitions of variable  $x$  in nodes  $(n_1, \dots, n_m, j)$  .
- The family of data flow criteria requires that the test data execute definition-clear paths from each node containing a definition of a variable to specified nodes containing *c- use* and edges containing *p-use* of that variable.

**PROGRAM CODE:**

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

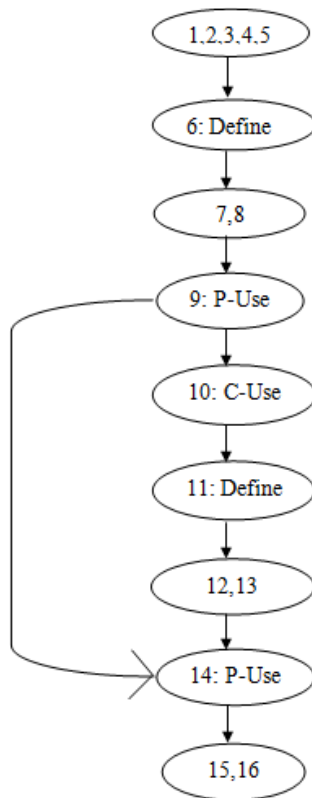
void main() {
    int a, b, t;
    system("cls");
    printf("Enter the first number ");
    scanf("%d",&a);
    printf("Enter the second number ");
    scanf("%d", &b);
```

```

if (a<b) {
    t=a;
    a=b;
    b=t;
}
printf("%d %d",a,b);
getch();
}

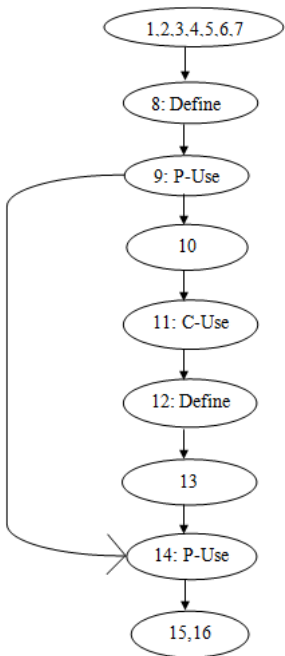
```

**Variable ‘a’ Data Flow Graph:**



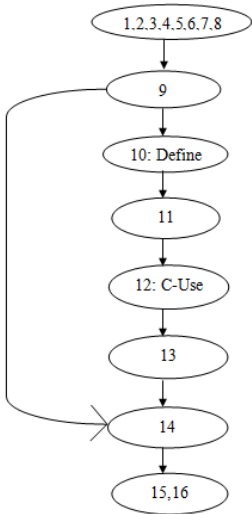
DU-PATHS	DC-PATHS

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		

**AIM:**

6. Write a C program to demonstrate the working of the looping constructs.

**ALGORITHM:****1. While Loops:****Syntax:**

```
while (condition)
    body;
```

**2. Do While loops****Syntax:**

```
do {
    body;
} while (condition);
```

**PROGRAM:**

```
#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;
}
```

**Output:**

Before loop, i=0

i=0

i=1

i=2

i=3

i=4

After loop, i=5

## Do While loops in C:

### Code:

```
#include <stdio.h>

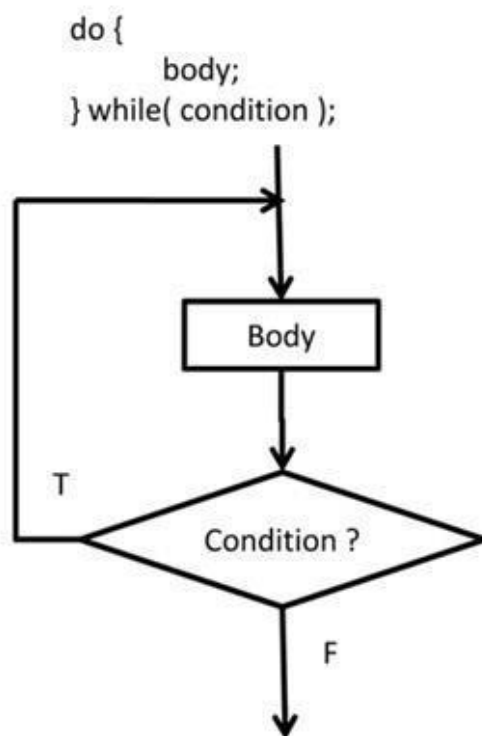
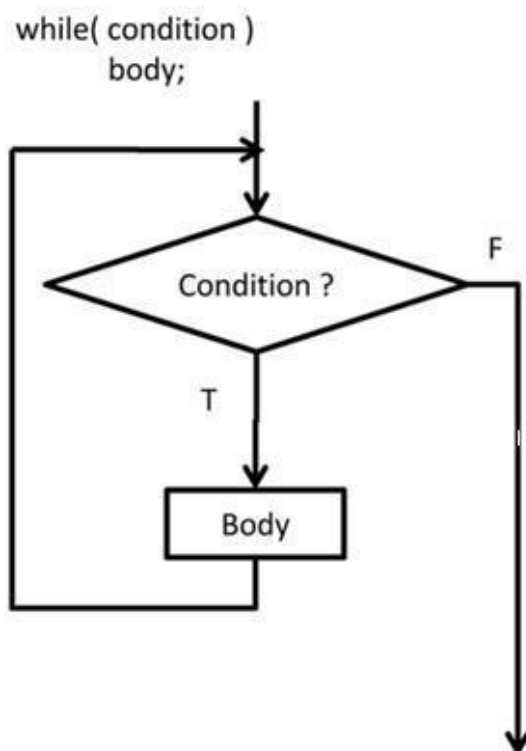
int main() {
    int month;
    do {
        printf("Please enter the month of your birth > ");
        scanf("%d", &month);
    } while (month < 1 || month > 12);
    return 0;
}
```

### OUTPUT:

Code runs till a month between 1-12 is entered

```
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
```

### While Vs Do-While Constructs:



**AIM:**

7. 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:**

```
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>
    <meta charset="UTF-8" />
</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" />
            </p>
        </fieldset>
    </form>

```



```
<label for="chkEggs">Green Eggs</label>

<input type="checkbox" id="chkHam" value="ham" />
<label for="chkHam">Ham</label>

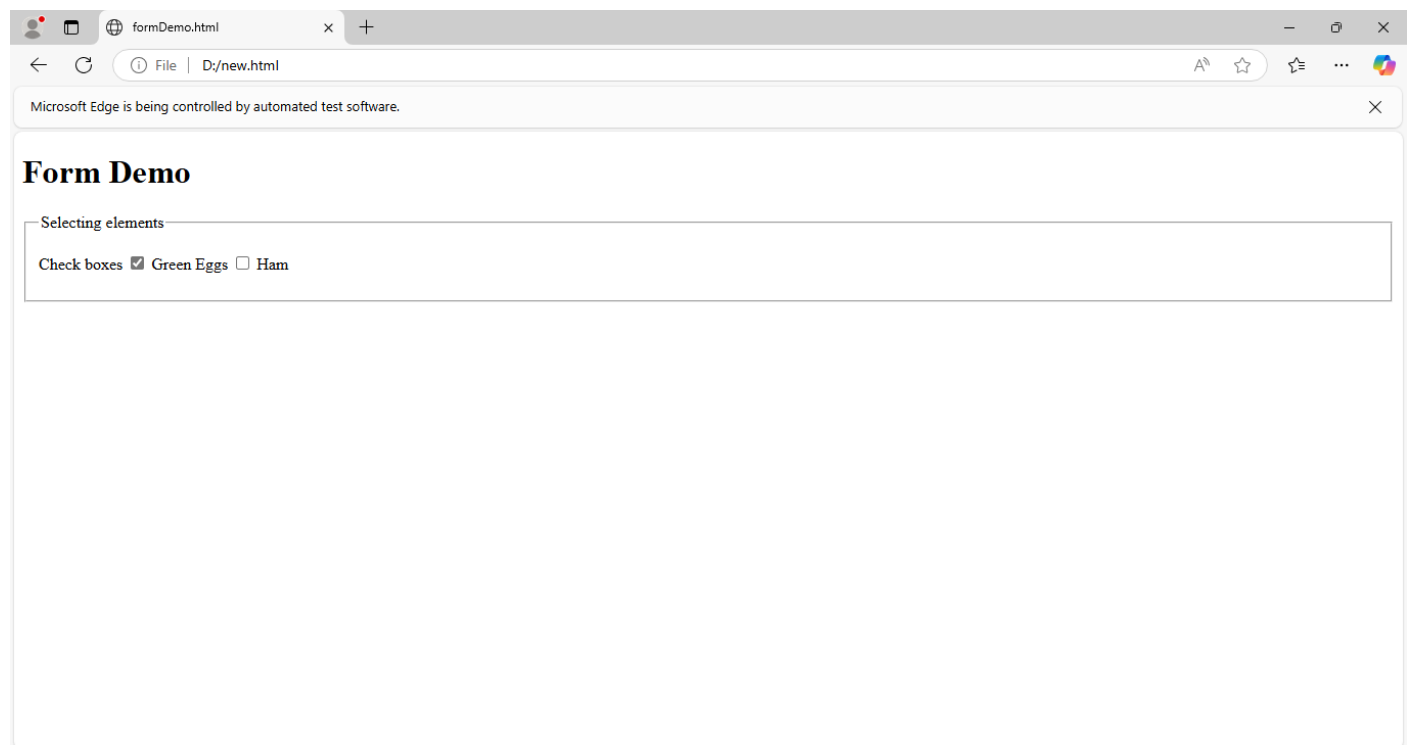
</p>
</fieldset>
</form>
</body>
</html>
```

## OUTPUT:

**Total checkboxes: 2**

**Selected checkboxes: 1**

**Unselected checkboxes: 1**



**AIM:**

8. 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();

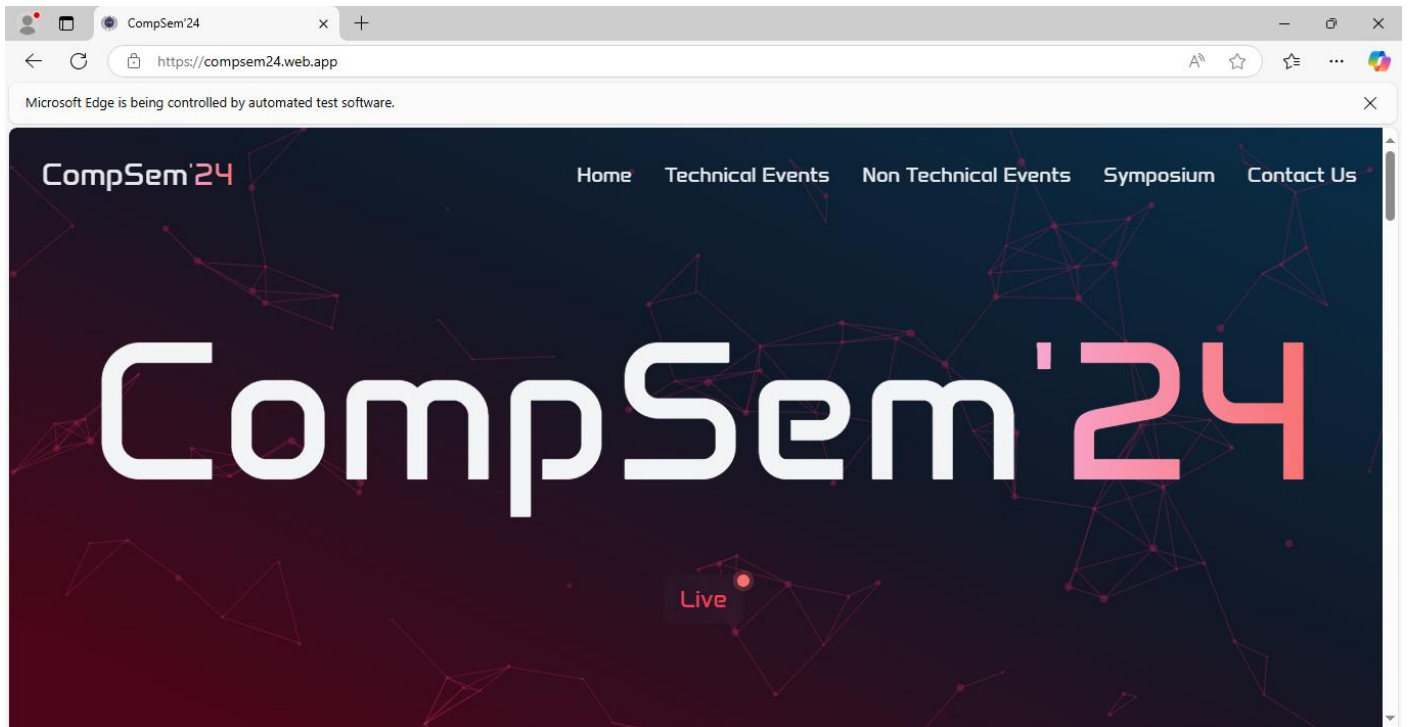
        driver.get("https://compsem24.web.app/");

        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



**AIM:**

9. 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:**

```
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");

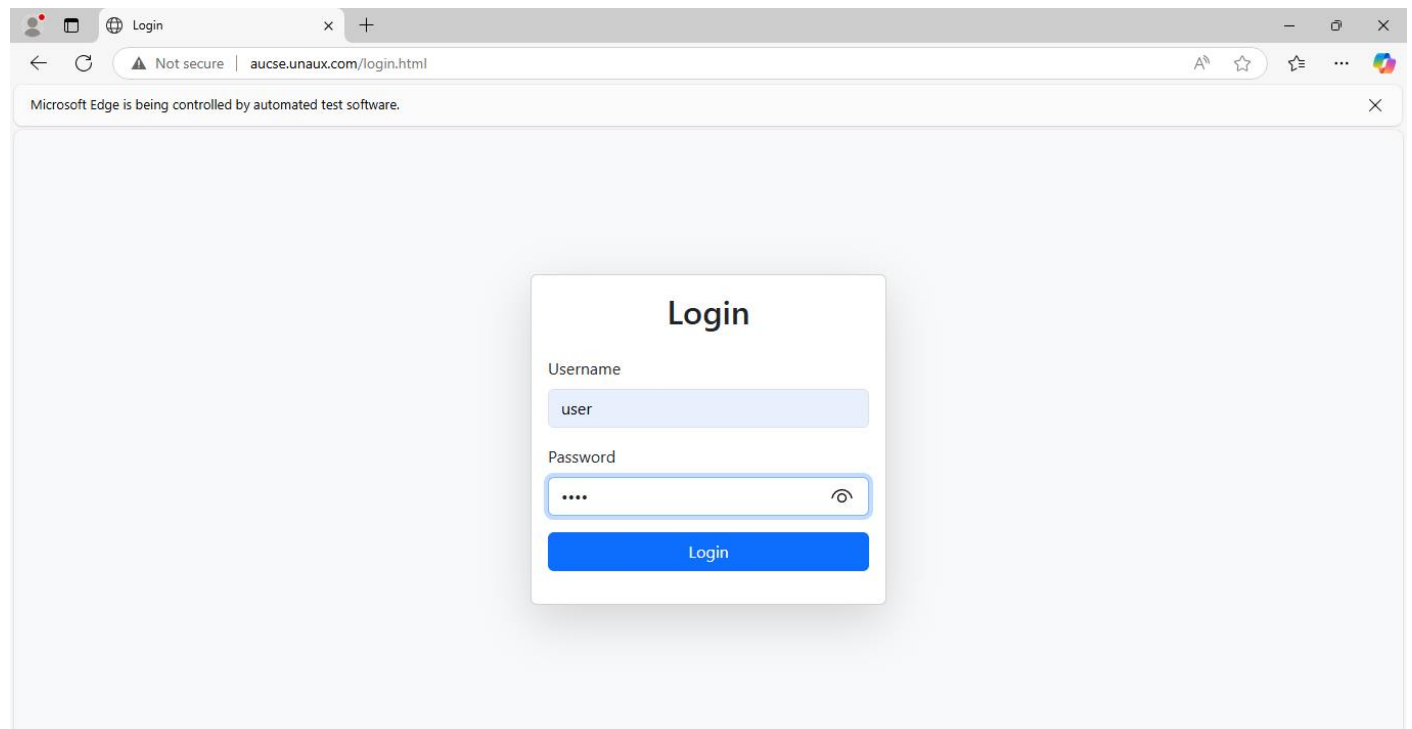
        WebElement username = driver.findElement(By.id("username"));
        WebElement password = driver.findElement(By.id("password"));

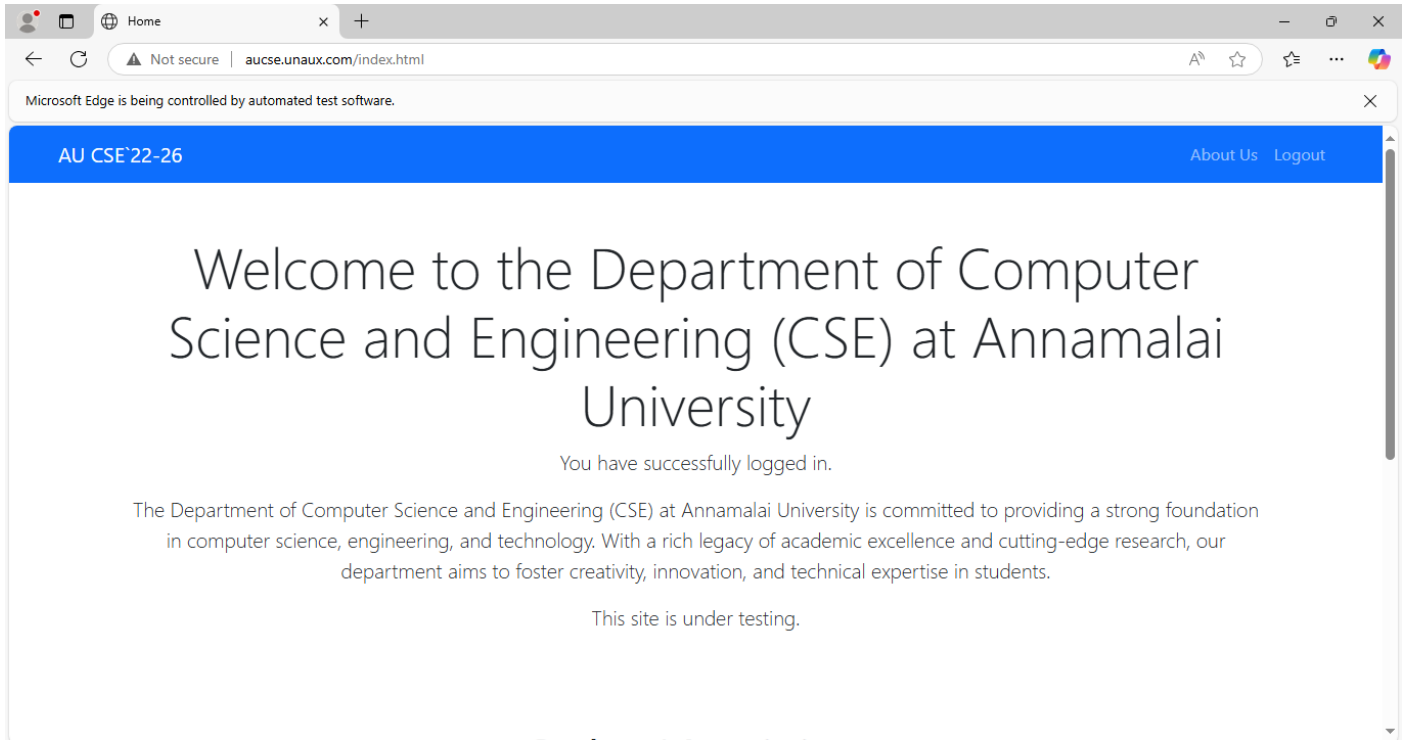
        WebElement loginButton = driver.findElement(By.cssSelector("button[type='submit']"));
```

```
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:





Test Passed: Login successful!

**AIM:**

10. 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	<u>Student Name</u>	<u>Subject1</u>	<u>Subject2</u>	<u>Subject3</u>
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

**AIM:**

11. 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:

```

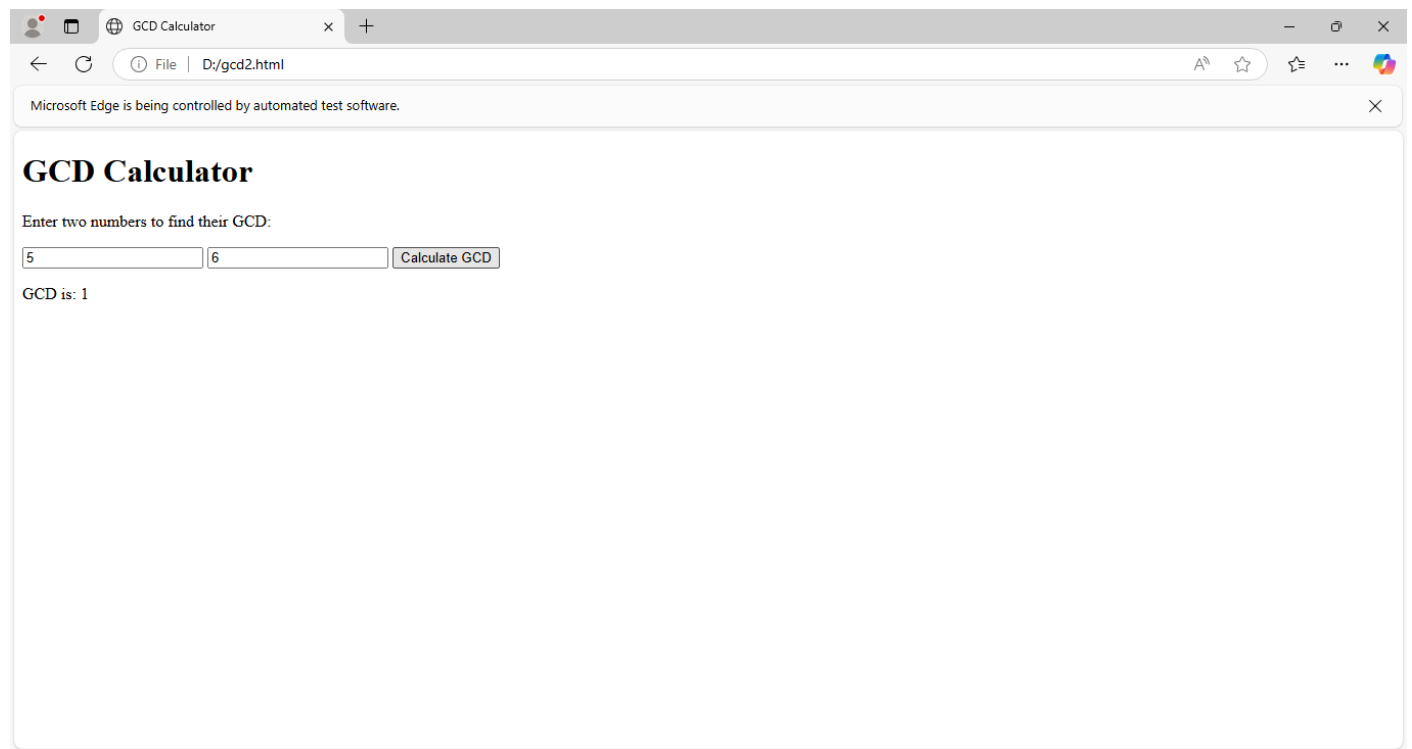
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>GCD Calculator</title>
<script>
    function calculateGCD() {
        var num1 = parseInt(document.getElementById("num1").value);
        var num2 = 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:



Test Result: GCD is: 1

**AIM:**

12. 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	<u>Student</u> <u>Name</u>	<u>Subject1</u>	<u>Subject2</u>	<u>Subject3</u>
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
11				