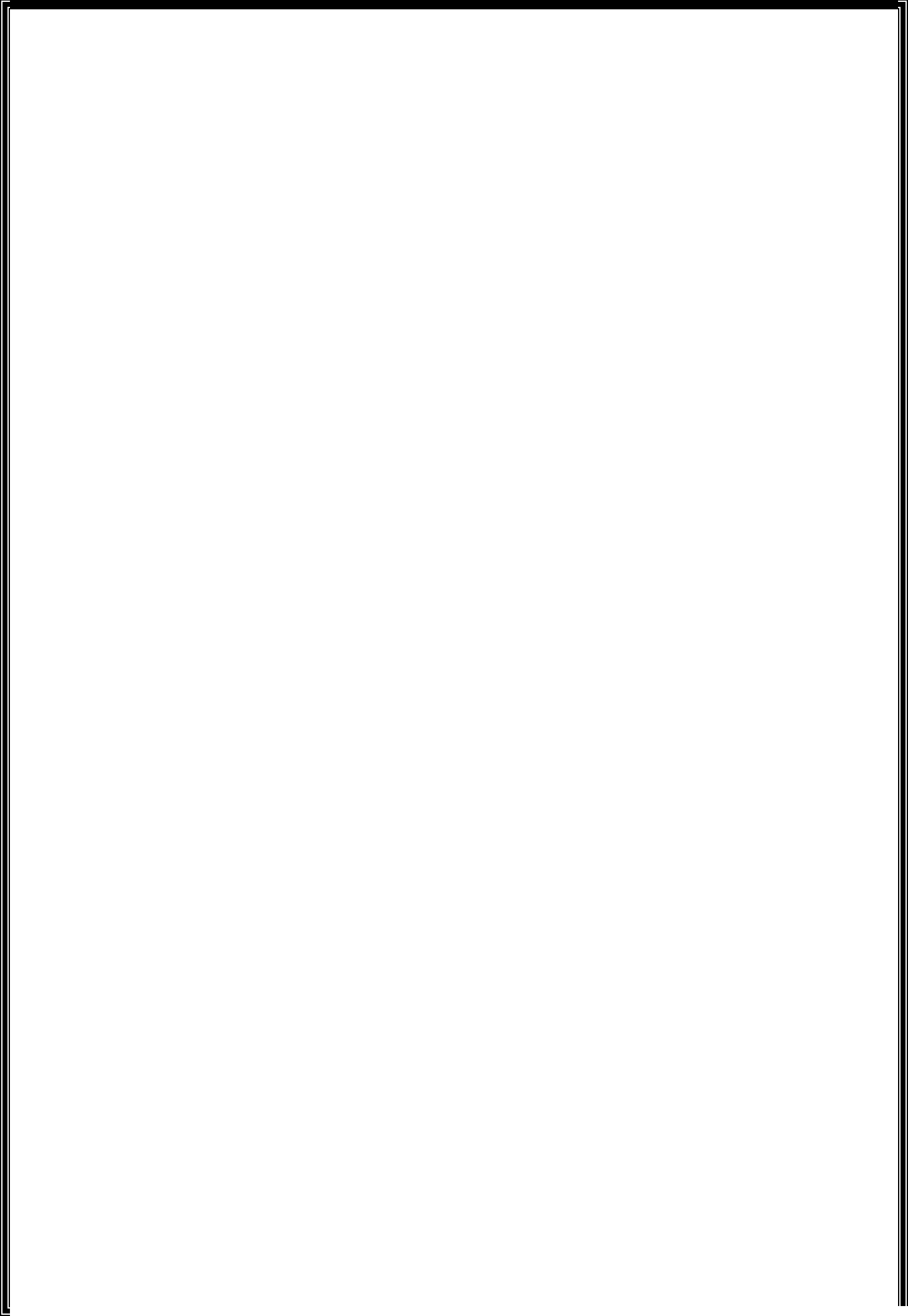
**V****ISVESVARRAYA TECHNOLOGICAL UNIVERSSITY**

**JNANNA SANGAMA,BELAGAVI – 590018**

**KARNATAKA**



**Mini Project Report**

**On**

**“LINEAR SEARCH”**

**SUBMITTED IN PAARTIAL FULFILLMENT OF THE ASSIGGNMENT**

**FOR THE Analysis & Design of Algorithms (BCS401)**

**COURSE OF IV SEMESTER**

**Submitted by**

**Name: ABHISHEK SM**

**USN: 1CG22CS003**

**Name: MALLAYYA SHARANAYYA MATH**

**USN: 1CG22CS061**

|  |  |
| --- | --- |
| **Guide:** | **HOD:** |
| **Mr.Asif Ulla Khan.M. Tech.** | **Dr. Shantala C PPhD.,** |
| **Asst. Prof., Dept. of CSE** | **Head, Dept. of CSE** |
| **CIT, Gubbi.** | **CIT, Gubbi.** |

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEEERING**



**Channabasaveshwara Institute of Technollogy**

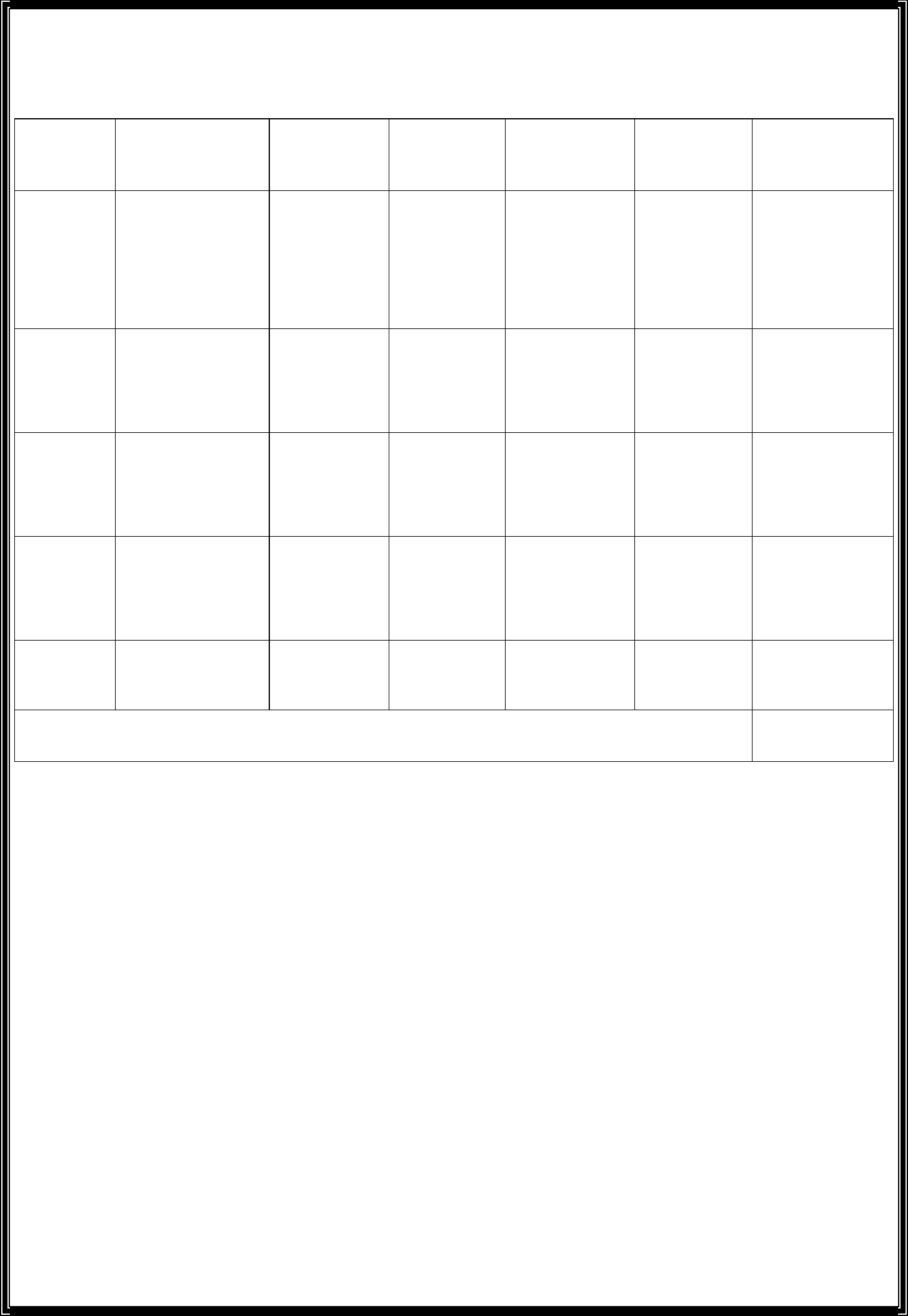


**(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi) (NAAC Acccredited & ISO 9001:2015 Certified Institution) NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka**



**2023-24**



**Rubric – B.E. Mini-Project [BCS401]**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Course** | **Rubric/Level** | **Excellent** | **Good** | **Average** | **Moderate** | **Score** |
| **outcome** |  | **(91-100%)** | **(81-90%)** | **(61-80%)** | **(40-60%)** |  |
| **CO1** | **Identification** |  |  |  |  |  |
|  | **of project** |  |  |  |  |  |
|  | **proposal** |  |  |  |  |  |
|  | **(05 Marks)** |  |  |  |  |  |
| **CO2** | **Design and** |  |  |  |  |  |
|  | **Implementation** |  |  |  |  |  |
|  | **(05 Marks)** |  |  |  |  |  |
| **CO3** | **Presentation** |  |  |  |  |  |
|  | **skill** |  |  |  |  |  |
|  | **(05 Marks)** |  |  |  |  |  |
| **CO 4** | **Individual or** |  |  |  |  |  |
|  | **in a team** |  |  |  |  |  |
|  | **development** |  |  |  |  |  |
| **CO5** | **Report** |  |  |  |  |  |
|  | **(05 Marks)** |  |  |  |  |  |
|  |  | **Total** |  |  |  |  |

**Course outcome:**

**CO 1: Identification of project proposal which is relevant to subject of engineering.**

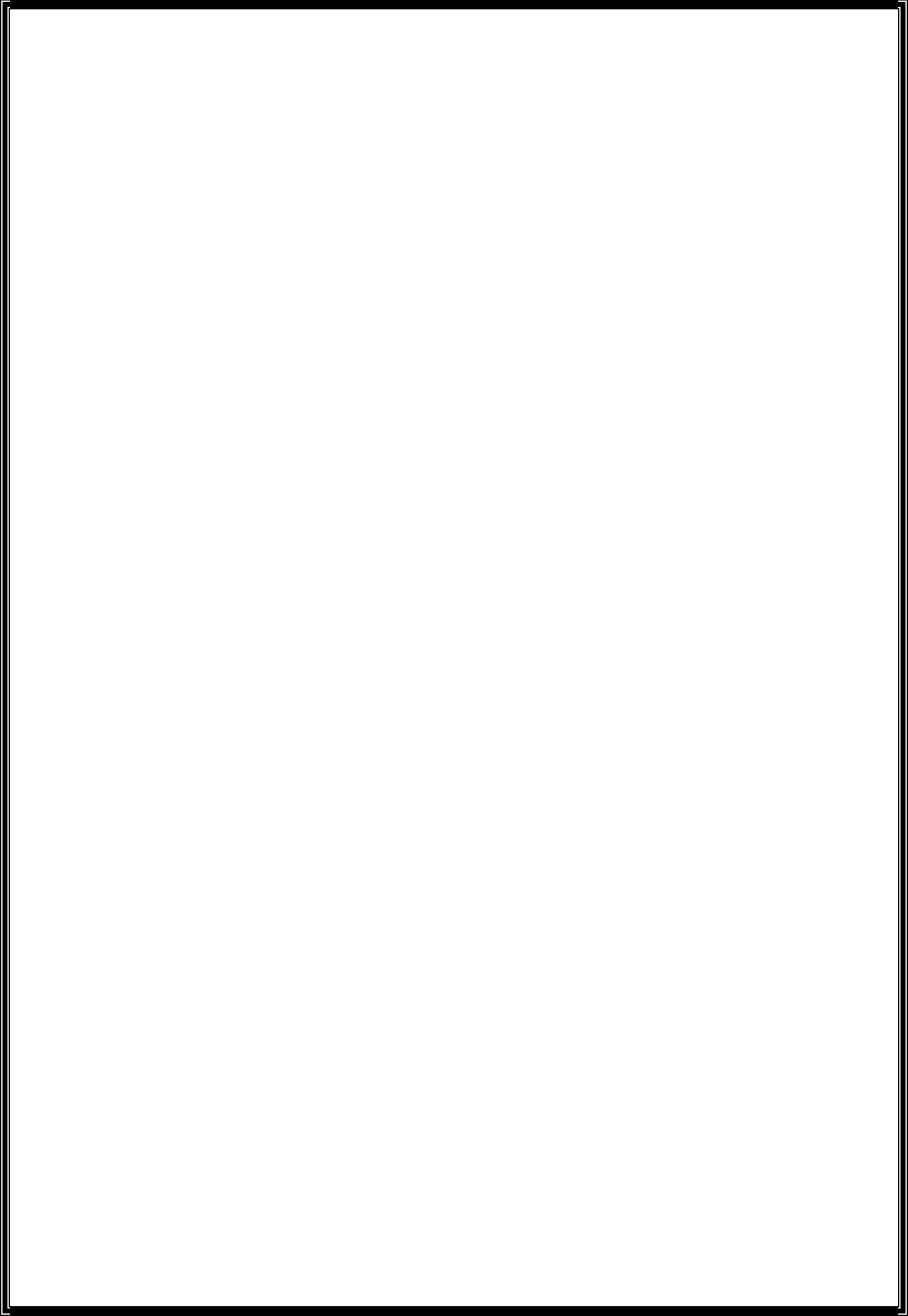
**CO 2: Design and implement proposed project methodology.**

**CO 3: Effective communication skill to assimilate their project work.**

**CO 4: Work as an individual or in a team in development of technical projects.**

**CO 5:Understanding overall project progress and performance.**

**Student Signature** **Faculty signature**

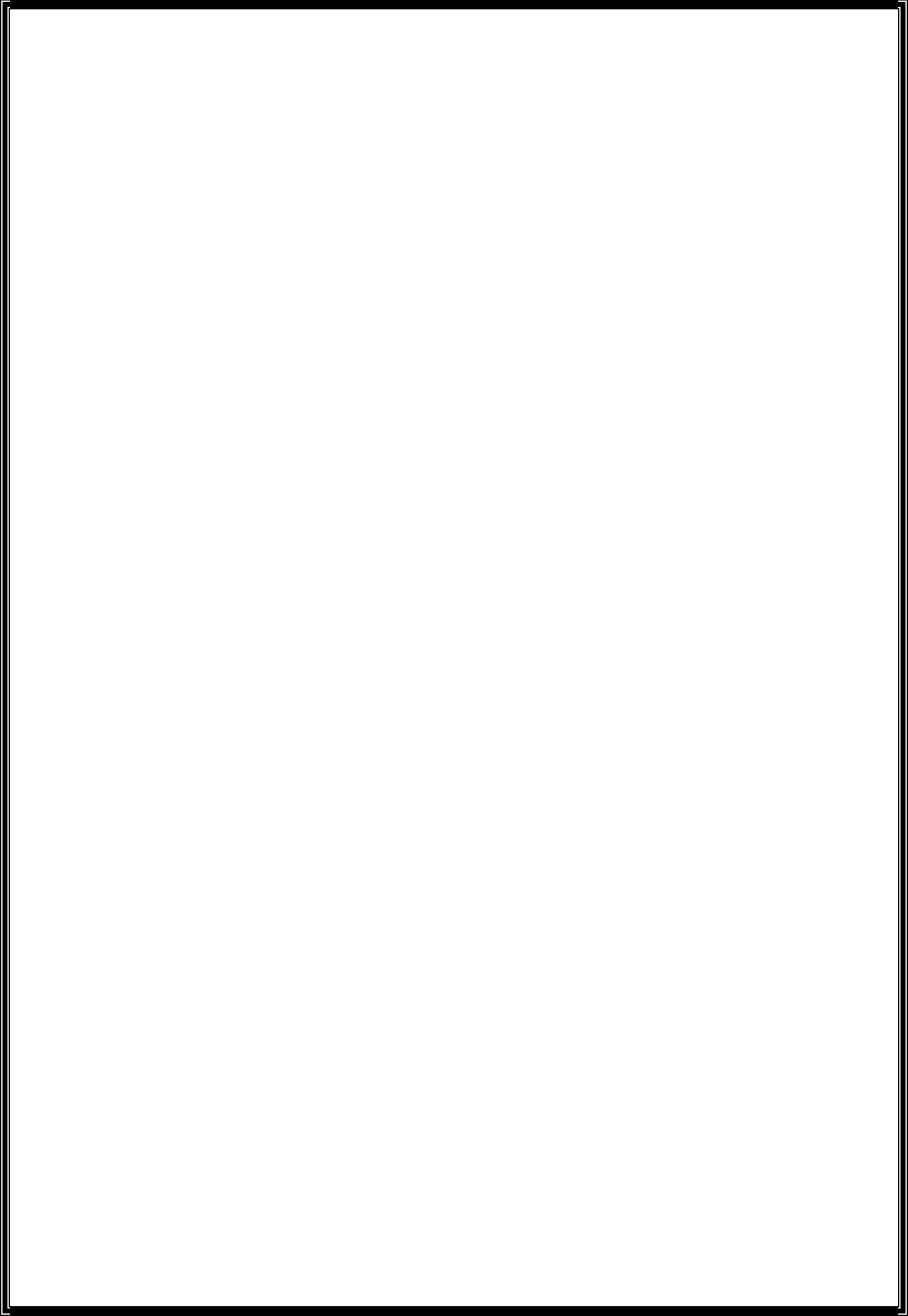
**ABSTRACT**

This Java program provides a practical demonstration of a linear search algorithm, integrated with user interaction to enhance learning and usability. The program first prompts the user to specify the number of elements in an array and then allows them to input each element. After the array is populated, the user is asked to enter the item they wish to search for within the array.

Using a Scanner object to handle user inputs, the program dynamically constructs the array based on user-provided data. The core functionality of the program is carried out by the linearSearchVersion2 method, which iterates through the array using a while loop. This method checks each element sequentially and prints informative messages indicating the current index and value being checked. If the item is found, the search stops, and the index of the found item is returned. If the item is not found, the method returns -1.

The main method of the program coordinates the user input, array initialization, and search operations, ensuring a smooth flow from data entry to search results. The search outcome, whether the item is found or not, is clearly communicated to the user, enhancing the program’s educational value.

This program not only demonstrates the principles of linear search but also emphasizes the importance of handling user inputs and array manipulations in Java. It serves as a foundational example for beginners to grasp the concept of search algorithms, their implementation, and practical applications in real-world scenarios.

**CHAPTER 1:**

**INTRODUCTION**

This Java program serves as an educational tool for understanding and implementing a linear search algorithm. Linear search is one of the simplest and most intuitive search techniques, used for finding a specific element within a list or array. This program guides the user through the process of inputting an array, specifying a search item, and then locating that item using linear search.

**Key Features of the Program**

**1. User Interaction:**

- Array Input: The program starts by prompting the user to enter the number of elements in the array. This allows the program to dynamically allocate the appropriate amount of memory for the array.

- Element Input: The user is then asked to input each element of the array individually. This step ensures that the array is populated with user-defined values, making the search process relevant to the user's data.

- Search Item Input: After the array is populated, the user specifies the item they wish to search for within the array.

**2. Linear Search Algorithm:**

- Initialization: The algorithm initializes variables to track the current index, a flag to indicate if the item has been found, and a variable to store the found index.

- Iteration: The algorithm iterates through each element of the array using a while loop. At each step, it compares the current array element to the search item.

- Condition Checking: If the current element matches the search item, the algorithm sets the found index, raises the flag to stop further iterations, and outputs a message indicating the item has been found.

- Completion: If the loop completes without finding the item, the program outputs a message indicating that the item was not found in the array.

**3. Output:**

- The program provides real-time feedback at each step of the search process. It prints the current index and the value being checked, and it also informs the user when the item is found or not found.

**Detailed Code Walkthrough**

**1. Main Method:**

- The main method is the entry point of the program. It handles user input and invokes the linear search method.

- It creates a Scanner object to read input from the user.

- It prompts the user to enter the number of elements in the array and reads the input.

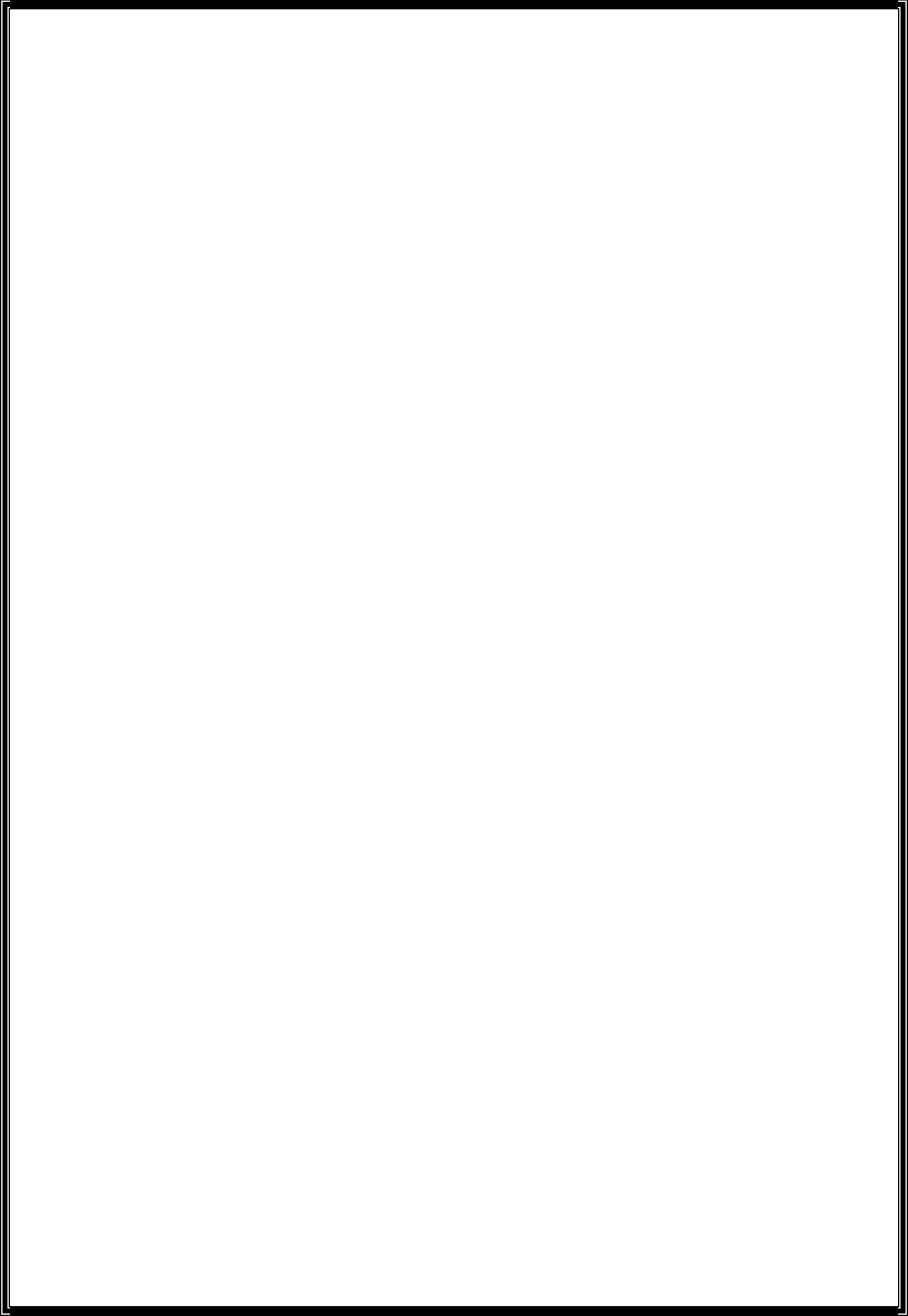
- It initializes the array based on the user-defined size.

- It prompts the user to input each element of the array.

- It prints the array to the console for user verification.

- It prompts the user to enter the item to search for and reads the input.

- It calls the linearSearchVersion2 method and handles the result by printing appropriate messages

**CHAPTER 2:**

**PROBLEM STATEMENT**

**Requirements:**

**1. User Input:**

- The program should prompt the user to enter the number of elements in the array.

- The program should allow the user to input each element of the array.

**2. Display Array:**

- After taking the input, the program should display the entire array.

**3. Search Item:**

- The program should prompt the user to enter the item to search for within the array.

**4. Linear Search Implementation:**

- Implement a linear search algorithm that iterates through the array elements to find the specified item.

- The search should stop once the item is found, and the program should output the index of the found item.

- If the item is not found, the program should output an appropriate message.

**5. Output:**

- Display the index of the found item, or a message indicating that the item was not found in the array.

**Constraints:**

- The array should contain integers.

- The array size and elements are provided by the user at runtime.

**Example:**

1. User inputs the number of elements: 7

2. User inputs the array elements: 11, 25, 10, 29, 15, 13, 18

3. Program displays: [11, 25, 10, 29, 15, 13, 18]

4. User inputs the search item: 15

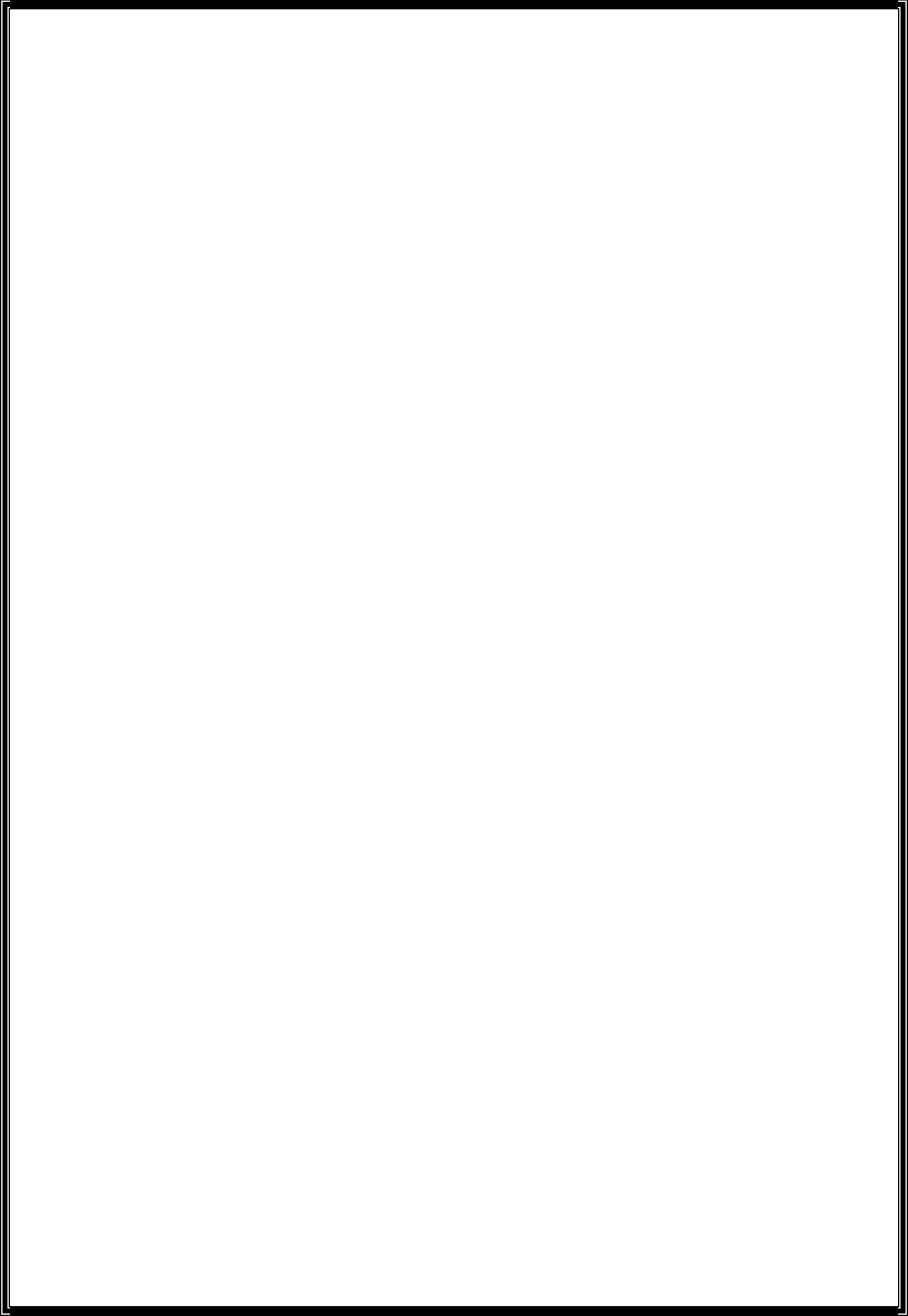
5. Program outputs: 15 was found at index 4

**Notes:**

- Ensure proper handling of user inputs.

- Provide clear and concise prompts and output messages.

.**.**

**CHAPTER 3:**

**ALGORITHM**

**The program reads user inputs to create and populate an array, then performs a linear search to find a specified item. It prints each search step, displays the result, and demonstrates basic user input handling and search algorithm implementation in Java.**

**Algorithm:**

**STEP 1:** Start by initializing an index variable current to 0 and a result variable foundIndex to -1 to track the index of the search item if found.

**STEP 2:** Iteratethrough each element in the array using a loop that continues while current is less than the length of the array.

**STEP 3**:In each iteration, compare the element at the current index with the searchItem.

**STEP 4:**If the element matches the searchItem, set foundIndex to current, and break out of the loop since the item has been found.

**STEP 5:**If the element does not match, increment current by 1 to move to the next element in the array.

**STEP 6:**After exiting the loop, check the value of foundIndex.

**STEP 7**:If foundIndex is still -1, the searchItem was not found in the array, and you should return -1.

**STEP 8**:If foundIndex is not -1, return the value of foundIndex as it represents the index where the searchItem was found.

**Pseudo Code:**

CREATE scanner object

PRINT "Enter the number of elements in the array: "

READ integer n

INITIALIZE array testItems of size n

PRINT "Enter the elements of the array:"

FOR i FROM 0 TO n-1

READ integer input

SET testItems[i] = input

END FOR

PRINT "### Linear search version 2 (while loop) ###"

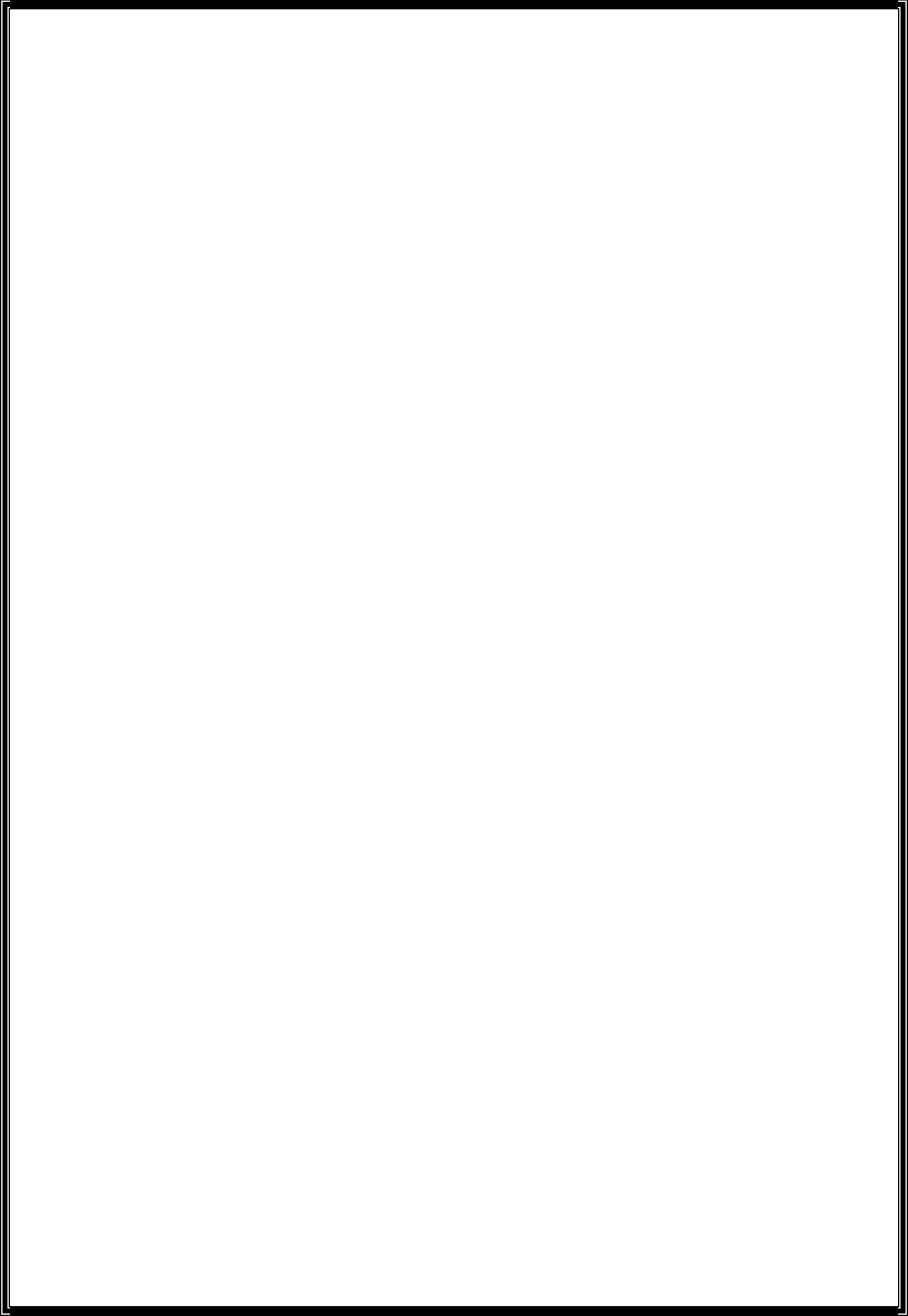
PRINT testItems

PRINT "Enter the item to find: "

READ integer itemToFind

PRINT "The search item is " + itemToFind

// Call linear search function

**** SET index = linearSearchVersion2(testItems, itemToFind)

IF index == -1 THEN

PRINT itemToFind + " was not found in the array"

ELSE

PRINT itemToFind + " was found at index " + index

END IF

CLOSE scanner object

END

// Linear Search Method

FUNCTION linearSearchVersion2(items, searchItem)

SET foundIndex = -1

SET current = 0

SET found = FALSE

WHILE NOT found AND current < LENGTH of items

PRINT "Checking index " + current + " with value " + items[current]

IF items[current] == searchItem THEN

SET foundIndex = current

SET found = TRUE

PRINT "Item found! Stopping the search"

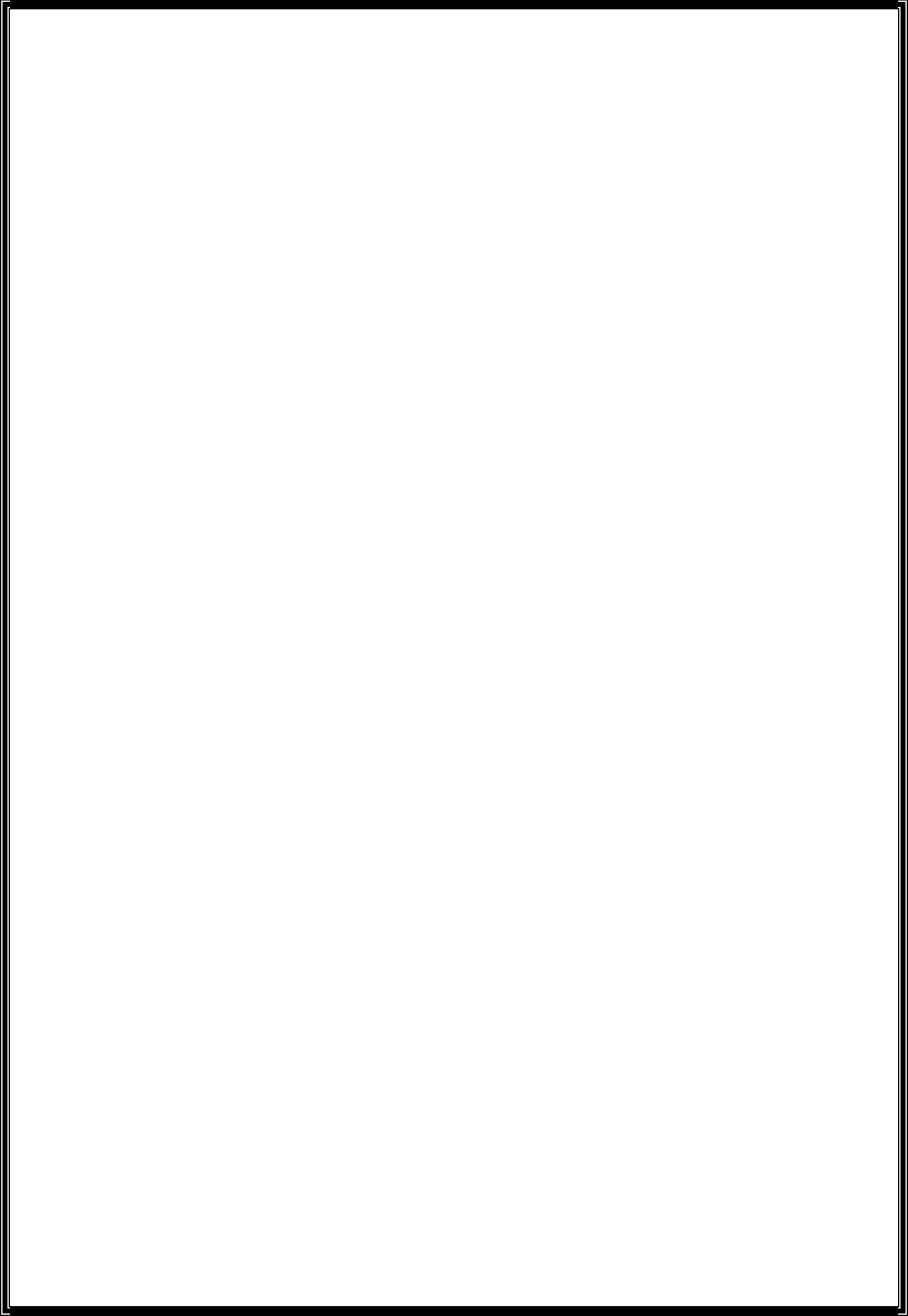
END IF

INCREMENT current by 1

END WHILE

RETURN foundIndex

END FUNCTION

**CHAPTER 4:**

**IMPLEMENTATION**

The program reads user inputs to create and populate an array, then performs a linear search to find a specified item. It prints each search step, displays the result, and demonstrates basic user input handling and search algorithm implementation in Java.

**CODE :**

**import java.util.Arrays;**

**import java.util.Scanner;**

**class SearchingAlgorithms {**

**// The main method is the entry point for all Java programs**

**public static void main(String[] args) {**

**// Create a Scanner object to read input from the user**

**Scanner scanner = new Scanner(System.in);**

**// Ask the user for the number of elements in the array**

**System.out.print("Enter the number of elements in the array: ");**

**int n = scanner.nextInt();**

**// Initialize the array with the user-defined size**

**int[] testItems = new int[n];**

**// Ask the user to input each element of the array**

**System.out.println("Enter the elements of the array:");**

**for (int i = 0; i < n; i++) {**

**testItems[i] = scanner.nextInt();**

**}**

**System.out.println("### Linear search version 2 (while loop) ###");**

**System.out.println(Arrays.toString(testItems));**

**// Ask the user for the item to find**

**System.out.print("Enter the item to find: ");**

**int itemToFind = scanner.nextInt();**

**System.out.println("\nThe search item is " + itemToFind + "\n");**

**// Search for the value and return the found index**

**int index = linearSearchVersion2(testItems, itemToFind);**

**if (index == -1) {**

**System.out.println("\n" + itemToFind + " was not found in the array");**

**} else {**

**System.out.println("\n" + itemToFind + " was found at index " + index);**

**}**

**// Close the scanner**

**scanner.close();**

**}**

**// A linear search algorithm that stops iterating if the item is found**

**public static int linearSearchVersion2(int[] items, int searchItem) {**

**// Initialise the variables**

**int foundIndex = -1;**

**int current = 0;**

**boolean found = false;**

**// Repeat while the end of the array has not been reached**

**// and the search item has not been found**

**while (!found && current < items.length) {**

**System.out.println("Checking index " + current + " with value " + items[current]);**

**// Compare the item at the current index to the search item**

**if (items[current] == searchItem) {**

**// If the item has been found, store the current index**

**foundIndex = current;**

**found = true; // Raise the flag to stop the loop**

**System.out.println("Item found! Stopping the search");**

**}**

**current = current + 1; // Go to the next index in the array**

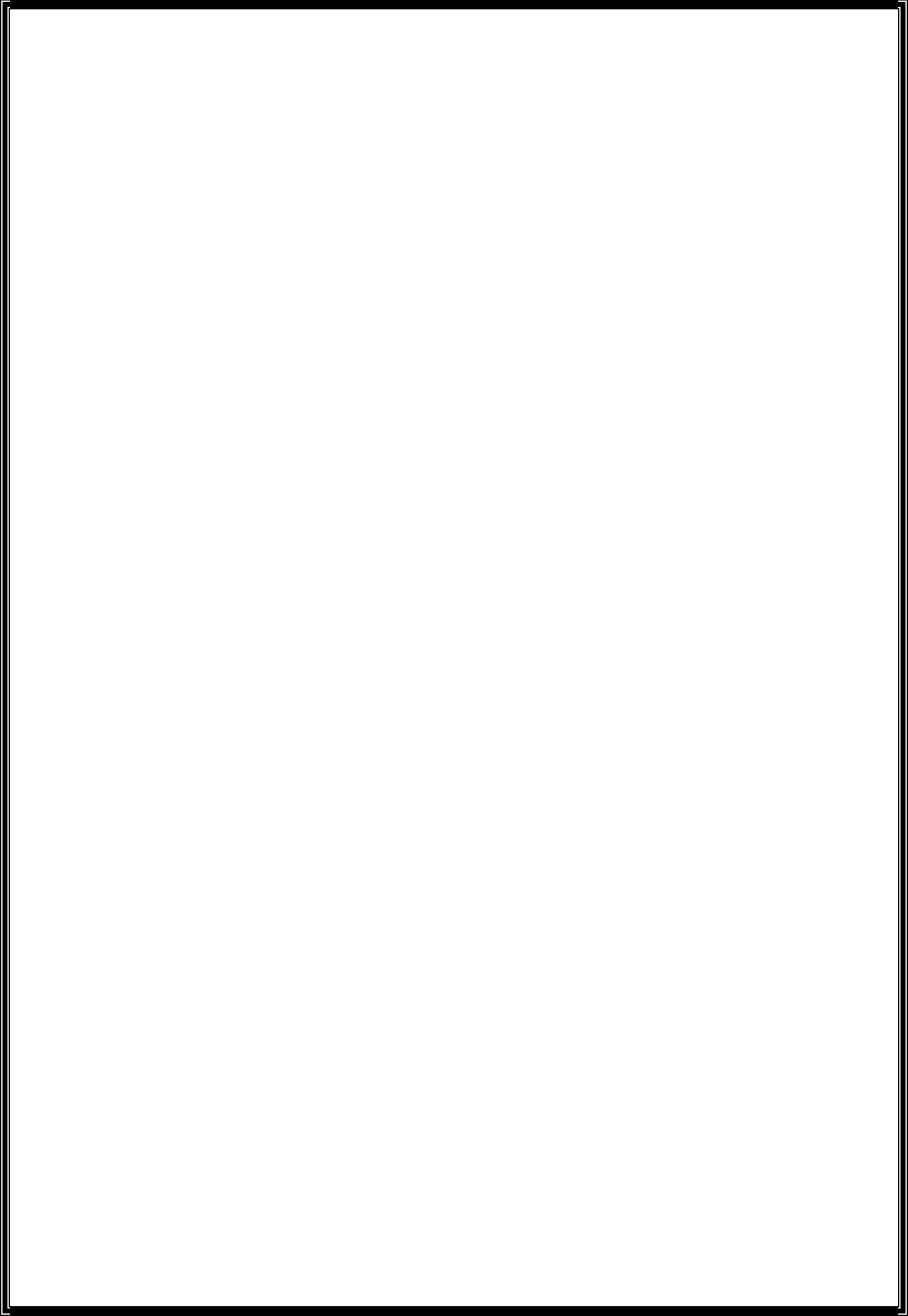
**}**

**// Return the index of the searchItem or -1 if not found**

**return foundIndex;**

**}**

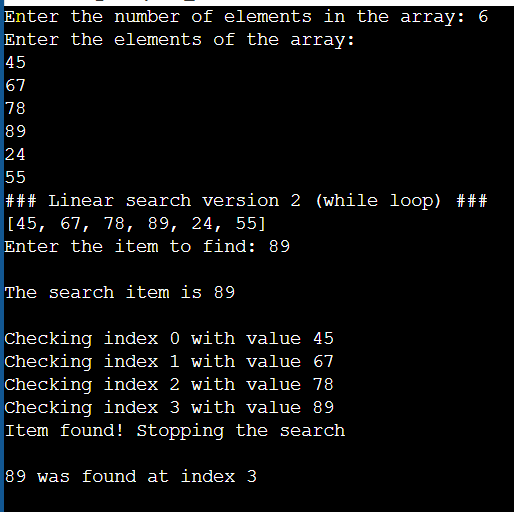
**}**

****

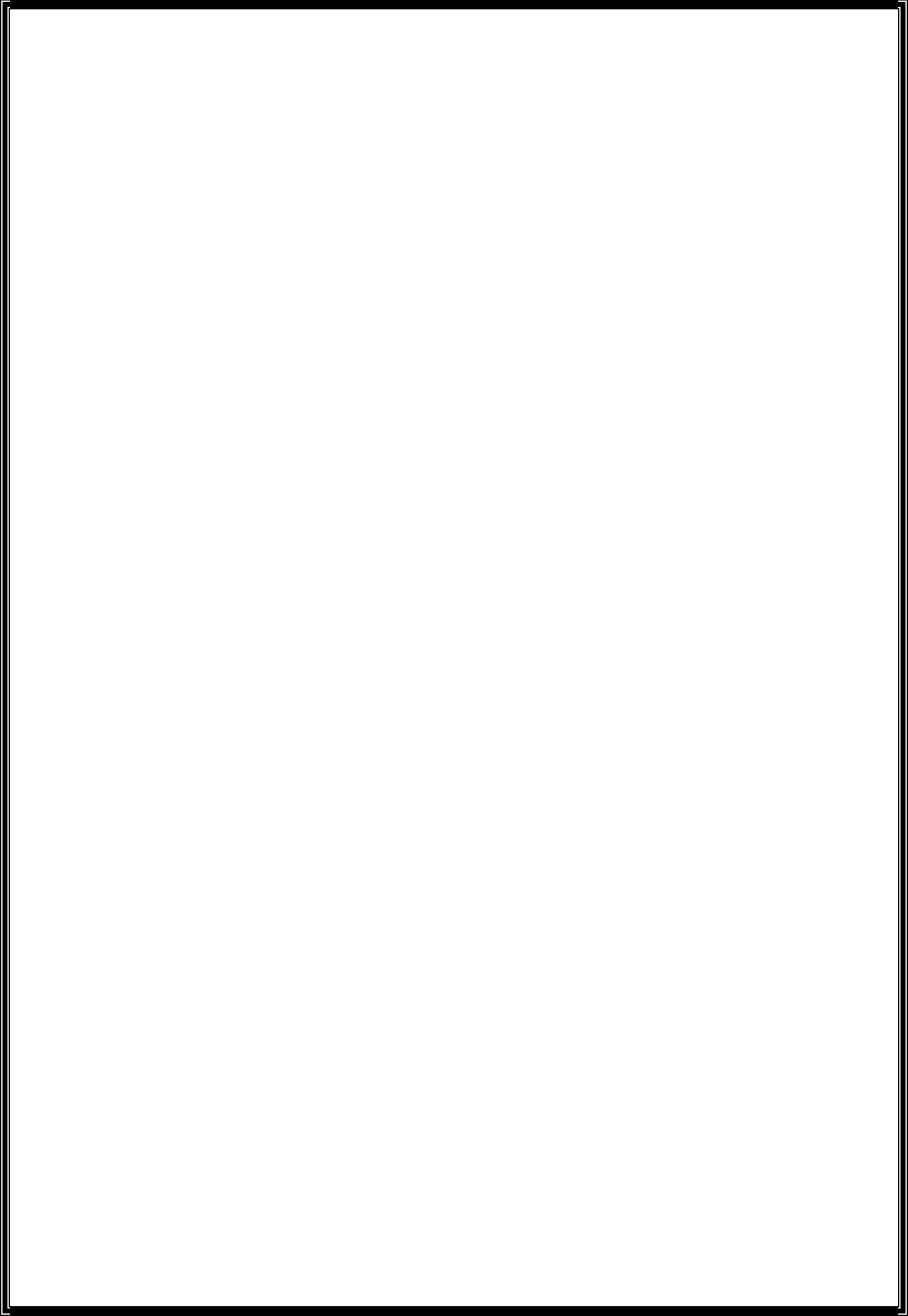
**CHAPTER 5:**

**RESULTS**

**OUTPUTS:**

****

**FIGURE : 01**

**CHAPTER 6:**

**CONCLUSION**

The provided Java program effectively demonstrates a linear search algorithm, allowing user interaction to input an array and specify a search item. Using a Scanner object, the program reads user input to create and populate an array dynamically. The linear search algorithm then checks each element sequentially until it finds the specified item or reaches the end of the array.

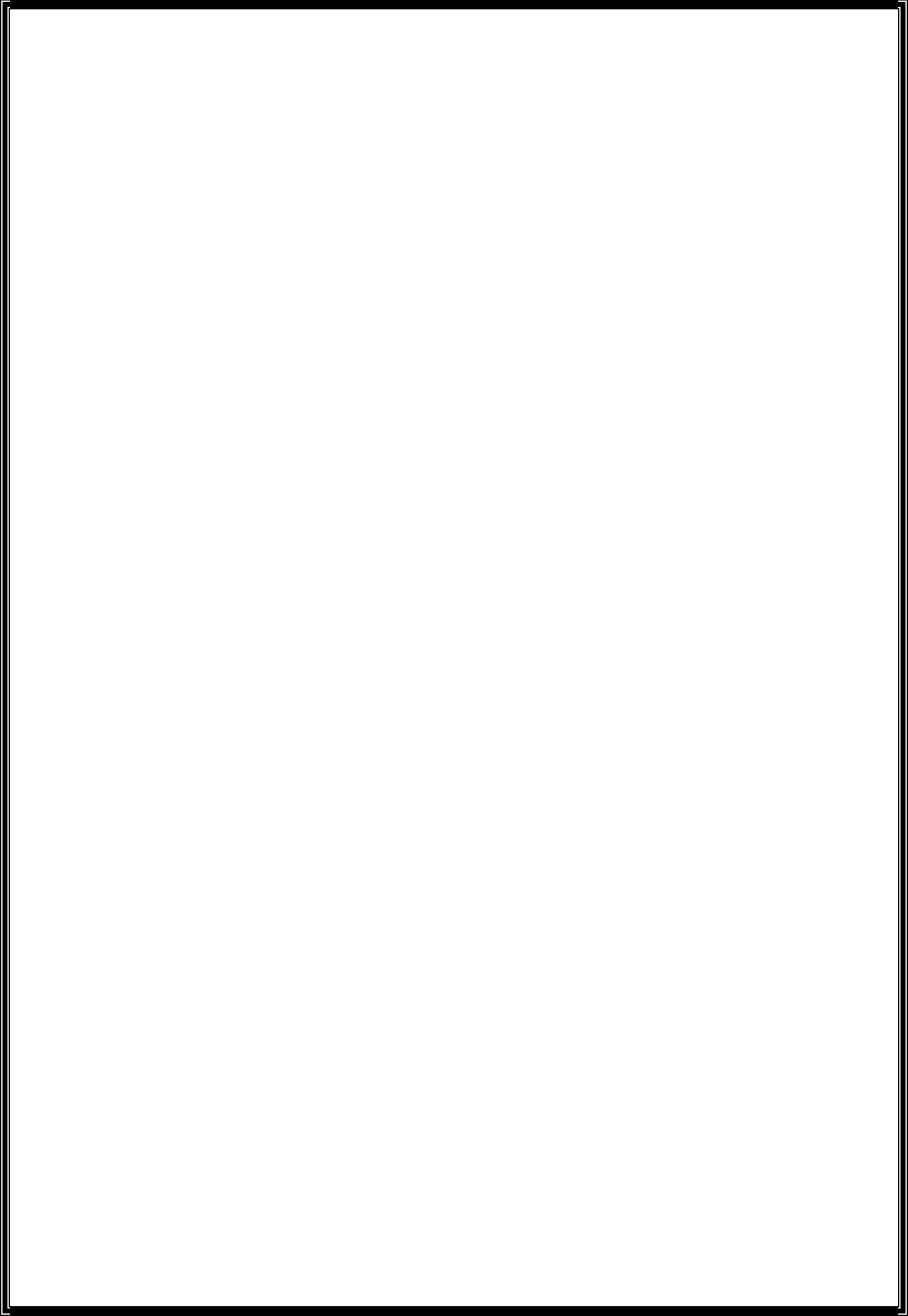
The program's structure is straightforward:

- The main method manages user input, array initialization, and calling the search function.

- The linearSearchVersion2 method performs the search, providing helpful messages during its process.

This program illustrates the basic principles of linear search and emphasizes user input handling and array manipulation in Java, serving as a foundational example for beginners to understand search algorithms in a practical context.

**.**

**REFERENCES:**

**GitHub : SmAbhi**

**Link**