1)a)

Aim: The aim of the program is to convert temperatures from Celsius to Fahrenheit. This conversion is intended to demonstrate how modern technology, particularly software development, has simplified and streamlined tasks that were once complex and time-consuming.

Algorithm:

Start the program.

Display a prompt asking the user to enter the temperature in Celsius.

Read the user input (temperature in Celsius).

Convert the Celsius temperature to Fahrenheit using the conversion formula.

Display the result, which is the temperature in Fahrenheit, to the user.

End the program.

1)b)

Aim:The aim of the program is to create a BMI calculator for a health app. This calculator allows users to input their weight in kilograms and height in meters, and then computes and displays their Body Mass Index (BMI). BMI is a measure of body fat based on height and weight that applies to adult men and women.

Algorithm:

Start the program.

Prompt the user to enter their weight in kilograms.

Read the user input (weight).

Prompt the user to enter their height in meters.

Read the user input (height).

Calculate the BMI.

Display the calculated BMI value to the user.

End the program.

2)a)

Aim:The aim of the program is to develop a retail billing system that calculates the total cost of a purchase after applying discounts based on certain conditions. The program helps users understand the total cost they will incur after applying discounts according to the purchase amount.

Algorithm:

Start the program.

Prompt the user to enter the purchase amount.

Read the user input (purchase amount).

If the purchase amount does not meet the conditions for discounts:

Apply no discount.

Calculate the total cost after applying the discount.

Display the total cost after discount to the user.

2)b)

Aim:The aim of the program is to prompt users to input a number and then determine if it's positive, negative, or zero. Based on this analysis, the program provides immediate feedback to the user regarding the nature of the number.

Algorithm:

Start the program.

Prompt the user to enter a number.

Read the user input number.

Check if the number is greater than zero:

If true, display "The entered number is positive."

If false, proceed to the next step.

Check if the number is less than zero:

If true, display "The entered number is negative."

If false, the number must be zero, so display "The entered number is zero."

3)a)

The aim of the program is to create a fun learning tool to help users understand multiplication better by generating multiplication tables from 1 to 10 for any number entered by the user.

Algorithm:

Start the program.

Display a prompt asking the user to enter a number or to type 'exit' to quit.

Read the user input.

If the input is 'exit', exit the program.

If the input is a valid number (between 1 and 10), proceed to step 6. Otherwise, display an error message and go back to step 2.

Generate and display the multiplication table for the entered number.

Repeat steps 2-6 until the user decides to exit the program.

3)b)

Aim: The aim of the program is to assist users in finding the factors of a positive integer. It provides a tool for users to easily identify all the numbers that divide the given integer without leaving a remainder.

Algorithm:

Start the program.

Display a prompt asking the user to enter a positive integer or to type 'exit' to quit.

Read the user input.

If the input is 'exit', exit the program.

If the input is not a positive integer, display an error message and go back to step 2.

If the input is a positive integer, proceed to step 7.

Initialize an empty list to store the factors.

Iterate from 1 to the entered number (inclusive).

a. For each iteration, check if the current number divides the entered number without leaving a remainder.

b. If it does, append the current number to the list of factors.

Display the list of factors to the user.

Go back to step 2 unless the user decides to exit the program.

4)a)

Aim: The aim of this program is to prompt the user to input a positive integer and determine whether the entered number is a prime number or not. It should continuously prompt the user until they enter 0 to exit the program. The program should utilize a loop to traverse through each number's factors up to its square root and determine its primality.

Algorithm:

Step1: Enter a positive integer.

step 2: If the input number is less than or equal to 1, print an error message and exit the program.

step 3: Initialize a boolean variable isPrime to true.

step 4:Check divisibility of the number by all integers from 2 up to the square root of the number:

step 5: If the number is divisible by any integer in this range, set isPrime to false and break out of the loop.

step 6: If isPrime is true, print that the number is a prime number.

step 7: If isPrime is false, print that the number is not a prime number.

4)b)

Aim: The aim of this Java program is to prompt the user to input a positive integer and reverse its digits using a while loop. It should then display the reversed number to the user.

Algorithm:

step 1: Enter a positive integer.

step 2: If the input number is less than or equal to 0, print an error message and exit the program.

step 3: Initialize a variable reversedNumber to 0 to store the reversed number.

step 4: Initialize a variable remainder to store the remainder of the division.

step 5: While the input number is greater than 0:

step 6: Calculate the remainder of the input number when divided by 10 and store it in the remainder variable.

step 7: Multiply the reversedNumber variable by 10 and add the remainder to it.

step 8: Divide the input number by 10.

step 9: Print the reversed number.

5)a)aim:create a java program to develop software for a weather monitoring station. You're tasked with creating a program to analyze temperature data stored in an array of doubles. Your goal is to determine the maximum temperature recorded and its corresponding time index.

Algorithms:

1. The user is prompted to enter the size of the array.
2. The program then creates an array of doubles with that size.
3. The user is prompted to enter temperature readings, which are stored in the array.
4. The program finds the maximum temperature and its index in the array.
5. Finally, it prints out the maximum temperature and its corresponding index.

5)b) aim: create a java program to develop software for a temperature monitoring system. Your task is to create a program that analyzes temperature data stored in an array of integers. The program finds the highest recorded temperature, along with its index.

Algotithms:

1. The user is prompted to enter the size of the array.
2. The program then creates an array of integers with that size.
3. The user is prompted to enter elements of the array, which are stored in the array.
4. The program finds the maximum element and its index in the array.
5. Finally, it prints out the maximum element and its corresponding index.

6) a) Implementation of 2D Array:

Aim:

Design a Java program for a financial analytics tool to process a 2D array of integers, aiming to calculate the total value of all entries in the dataset and print the result.

Algorithm:

Step 1: Import the Scanner class to enable user input.

Step 2: Define a class named TwoDArraySum to encapsulate the program logic.

Step 3: Initialize variables sum and rowCount to store the cumulative sum and count of rows, respectively.

Step 4: Continuously prompt the user for input until an empty line is entered.

Step 5: Split each non-empty line into individual elements.

Step 6: Calculate the sum of elements in each row and update the rowCount accordingly.

Step 7: Once input is complete, print the total sum of all elements in the 2D array.

6) b)

Aim:

Develop a Java program to analyze financial data stored in a 2D array of doubles, with the objective of identifying the highest value in the dataset along with its respective row and column indices.

Algorithm:

Step 1: Declare variables for row count, column count, maximum element, and their indices.

Step 2: Input array dimensions from the user.

Step 3: Fill the array with user-provided values.

Step 4: Set initial maximum element and indices to the first element.

Step 5: Iterate through the array to find the maximum element and its indices.

Step 6: Print the maximum element and its indices.

Step 7: Close the scanner.

Step 8: End the program.

7) a) Implementation of string functions

Aim

To implement a java program that develops a Text Refinement Tool to process text inputs. Users expect the tool to remove duplicate characters and count occurrences of vowels for linguistic analysis. The tool accepts a string as input and redefines it, presenting the refined text without duplicate characters and providing counts of vowels. Design a program that meets these requirements efficiently.

Algorithm:

Step 1: Create a StringBuilder refinedText to store the refined text and an array vowelCount of size 26 to store counts of each vowel (a to z).

Step 2:Iterate over each character c in the input string.

Step 3: Check if c is not a whitespace character.

Convert c to lowercase.

Step 4: If c is a lowercase letter (a to z), increment the corresponding vowel count in vowelCount.

Step 5: If c is not already present in refinedText, append it to refinedText.

Step 6: Create a new StringBuilder output and append refinedText to it.

Step 7: Iterate over each vowel count in vowelCount. If a vowel count is greater than 0, append the vowel and its count to output. If no vowels are found, append "No vowels found." to output. Finally, return output as a string

7) b)

Aim:

To implement a java program that enhance a popular text-editing app with a new feature to stylize text for a more captivating presentation. With the app's extensive user base, efficiency and performance are critical considerations. Users desire a feature that reverses the word order of their input sentence while alternating the case of each letter, starting with uppercase for a visually engaging effect.

Algorithm:

Step 1: Split the input string into an array of words using split(" ").

Step 2: Create a StringBuilder transformed to store the transformed text.

Step 3: Iterate over the words array in reverse order.

Step 4: For each word, convert it to a char array.

Step 5: For each character in the char array:

- If it is uppercase, append its lowercase version to transformed.
- If it is lowercase, append its uppercase version to transformed.
- If it is not an alphabetic character, append it as is.

Step 6: If the current word is not the first word (i.e., i > 0), append a space to transformed.

Step 7: Return the transformed text as a string.

8) a) Implementation of java program using streams

Aim:

The aim of the provided Java program is to read the content of a text file named input.txt, calculate the number of characters, words, and lines in the file, and then display these counts as output.

Algorithm:

Step 1: Begin the program execution.

Step 2: Initialize variables to store the filename, character count, word count, and line count.

Step 3: Attempt to open the input file for reading.

- If successful, read each line of the file.
- Increment the line count for each line read.
- Split the line into words using spaces as separators.
- Increment the word count by the number of words in the line.
- For each word in the line, increment the character count by the length of the word.

Step 4: If there's an IOException, handle it by printing an error message and exit the program.

Step 5: After processing all lines, print the total character count, word count, and line count.

Step 6: End the program execution.

8) b)

Aim:

The aim of this program is to prompt the user to enter a file name, check if the specified file exists in the current directory, and then display a message accordingly.

Algorithm:

Step 1: Start the program execution.

Step 2: Prompt the user to enter the name of the file.

Step 3: Read the file name provided by the user.

Step 4: Create a File object using the provided file name.

Step 5: Check if the file exists in the current directory.

- If the file exists:
  - Print "File found!".
  - Print the absolute path of the file.
- If the file does not exist:
  - Print "File not found!".

Step 6: End the program execution.

9) a)

Aim:

Imagine you're developing a utility to determine users' current ages for a registration platform. Develop a program that prompts users to input their birth year.

Algorithm:

Step 1: Initialize a variable named birthYear.

Step 2: Create a Scanner object to read user input.

Step 3: Begin a loop to prompt the user to enter their birth year until valid input is provided.

Step 4: Within the loop, check if the input is an integer and ensure it's not negative.

Step 5: If the input is valid, retrieve the current year.

Step 6: Calculate the user's age by subtracting their birth year from the current year.

Step 7: Print the calculated age.

Step 8: Close the Scanner object to release system resources.

9) b)

Aim:

To create a Java program that rounds a given floating-point number to the nearest integer for accurate representation of monetary values in financial applications.

Algorithm:

1. Define the main class NearestIntegerRounder.

2. Implement a static method roundToNearestInteger(double number) to round the input number.

3. Inside roundToNearestInteger, return the result of Math.round(number).

4. In the main method:

   a. Create a Scanner object scanner to read user input.

   b. Prompt the user to input a floating-point number.

   c. Read the input using scanner.nextDouble().

   d. Call roundToNearestInteger with the input number and store the result in a variable.

   e. Print the rounded number.

5. Close the Scanner object to release resources.

10) a)

Aim:

Design a Java text analysis tool to detect the prevalent usage of short words beginning with 'a' or 'A' in students' essays.

Algorithm:

1. Initialize a Scanner object to read user input.

2. Prompt the user to input a sentence.

3. Tokenize the sentence into individual words.

4. Define a regular expression pattern to match words starting with 'a' or 'A' and having exactly three characters.

5. Iterate through each word in the tokenized sentence.

6. Apply pattern matching to check if the word matches the defined pattern.

7. If a match is found, print the word.

8. Close the Scanner object to release system resources.

10) b)

Aim:

To create a Java application, "DataFinder," that enables users to locate specific data patterns within a text document by providing a file path and a data pattern to search for

Algorithm:

1. Define the main class DataFinder.

2. Implement the main method:

   a. Prompt the user to enter the file path of the text document.

   b. Read the entered file path and store it in a variable filePath.

   c. Prompt the user to enter the data pattern to search for.

   d. Read the entered data pattern and store it in a variable dataPattern.

   e. Use try-with-resources to open a BufferedReader for the specified filePath.

   f. Iterate through each line of the document:

      i. Apply pattern matching to locate instances of dataPattern.

      ii. Print each located data pattern.

   g. Handle any IOException that may occur during file reading.

3. The application terminates after executing the main method