Data types

1) **Simple Java program that demonstrates different data types:**

```
public class DataTypesProgram {
  public static void main(String[] args) {
    // Integer data types
    byte byteVar = 10;
    short shortVar = 100;
    int intVar = 1000;
    long longVar = 100000L; // Note the 'L' suffix for long literals

    // Floating-point data types
    float floatVar = 10.5f; // Note the 'f' suffix for float literals
    double doubleVar = 10.1234;

    // Character data type
    char charVar = 'A';

    // Boolean data type
    boolean boolVar = true;

    // Output
    System.out.println("byteVar: " + byteVar);
    System.out.println("shortVar: " + shortVar);
    System.out.println("intVar: " + intVar);
    System.out.println("longVar: " + longVar);
    System.out.println("floatVar: " + floatVar);
    System.out.println("doubleVar: " + doubleVar);
    System.out.println("charVar: " + charVar);
    System.out.println("boolVar: " + boolVar);
  }
}
```

2) **Example program demonstrating data type conversion in Java for all primitive data types:**

a) **Implicit conversion (widening) from smaller data types to larger ones.**

b) **Explicit conversion (narrowing) from larger data types to smaller ones.**

c) **Overflow and underflow scenarios where the value exceeds the range of the target data type.**

```
public class DataTypeConversion {
  public static void main(String[] args) {
    // Implicit conversion (Widening)
    int intValue = 10;
    long longValue = intValue; // int to long
    float floatValue = longValue; // long to float
    double doubleValue = floatValue; // float to double

    System.out.println("Implicit Conversion (Widening):");
    System.out.println("int to long: " + longValue);
    System.out.println("long to float: " + floatValue);
    System.out.println("float to double: " + doubleValue);

    // Explicit conversion (Narrowing)
    double doubleNum = 123.456;
    float floatNum = (float) doubleNum; // double to float
    long longNum = (long) floatNum; // float to long
    int intNum = (int) longNum; // long to int

    System.out.println("\nExplicit Conversion (Narrowing):");
    System.out.println("double to float: " + floatNum);
    System.out.println("float to long: " + longNum);
    System.out.println("long to int: " + intNum);

    // Overflow and Underflow
    int largeInt = Integer.MAX_VALUE;
```

```
        short shortNum = (short) largeInt; // int to short (overflow)
        byte byteNum = (byte) largeInt; // int to byte (overflow)

        System.out.println("\nOverflow and Underflow:");
        System.out.println("int to short (Overflow): " + shortNum);
        System.out.println("int to byte (Overflow): " + byteNum);

        int smallInt = Integer.MIN_VALUE;
        short shortNum2 = (short) smallInt; // int to short (underflow)
        byte byteNum2 = (byte) smallInt; // int to byte (underflow)

        System.out.println("int to short (Underflow): " + shortNum2);
        System.out.println("int to byte (Underflow): " + byteNum2);
    }
}
```

3) Java program for String methods and String constructors

```
public class StringMethodsAndConstructors {
    public static void main(String[] args) {
        // String Constructors
        String str1 = new String(); // Empty String
        String str2 = new String("Hello"); // String with specified content
        char[] charArray = {'W', 'o', 'r', 'l', 'd'};
        String str3 = new String(charArray); // String from char array
        byte[] byteArray = {65, 66, 67, 68, 69};
        String str4 = new String(byteArray); // String from byte array

        System.out.println("String Constructors:");
        System.out.println("str1: " + str1);
        System.out.println("str2: " + str2);
        System.out.println("str3: " + str3);
        System.out.println("str4: " + str4);

        // String Methods
        String original = "Hello World";
        System.out.println("\nString Methods:");
        System.out.println("Original String: " + original);

        // Length
        System.out.println("Length: " + original.length());

        // Concatenation
        String concatString = original.concat("!");
        System.out.println("Concatenated String: " + concatString);

        // Substring
        String substring = original.substring(6);
        System.out.println("Substring from index 6: " + substring);

        // Character extraction
        char charAt5 = original.charAt(5);
        System.out.println("Character at index 5: " + charAt5);

        // Index of a character
        int indexOfW = original.indexOf('W');
        System.out.println("Index of 'W': " + indexOfW);

        // Index of a character starting from a specific index
        int indexOfl = original.indexOf('l', 4);
        System.out.println("Index of 'l' after index 4: " + indexOfl);

        // Replace
```

```
        String replacedString = original.replace('l', 'L');
        System.out.println("String after replacing 'l' with 'L': " + replacedString);

        // Convert to uppercase
        String upperCaseString = original.toUpperCase();
        System.out.println("Uppercase String: " + upperCaseString);

        // Convert to lowercase
        String lowerCaseString = original.toLowerCase();
        System.out.println("Lowercase String: " + lowerCaseString);

        // Trim leading and trailing spaces
        String stringWithSpaces = "   Trimmed String   ";
        String trimmedString = stringWithSpaces.trim();
        System.out.println("Trimmed String: '" + trimmedString + "'");

        // String comparison
        String str5 = "Hello";
        String str6 = "hello";
        System.out.println("Comparison of str2 and str5 (case sensitive): " + str2.equals(str5));
        System.out.println("Comparison of str2 and str6 (case insensitive): " +
str2.equalsIgnoreCase(str6));

        // Check if starts with/ends with
        System.out.println("Does str2 start with 'He'?: " + str2.startsWith("He"));
        System.out.println("Does str2 end with 'lo'?: " + str2.endsWith("lo"));
    }
}
```

4) String builder and String buffer program differences with example

Demonstrate simple examples of StringBuilder and StringBuffer appending strings.

Then compare the performance of StringBuilder and StringBuffer by appending a large number of strings in a loop (iterations times) and measuring the time taken.

a) StringBuilder:

Not synchronized: StringBuilder is not synchronized, meaning it is not thread-safe. It is faster than StringBuffer but should not be used in multi-threaded environments unless explicitly synchronized externally.

Preferred when thread safety is not required or manually synchronized externally.

b) StringBuffer:

Synchronized: StringBuffer is synchronized, making it thread-safe. It is slightly slower than StringBuilder due to the overhead of synchronization.

Preferred when thread safety is required or when working in multi-threaded environments.

In general, StringBuilder is preferred for single-threaded scenarios or when synchronization is managed externally, while StringBuffer is preferred in multi-threaded environments where thread safety is crucial.

```
public class StringBuilderVsStringBuffer {
  public static void main(String[] args) {
    // StringBuilder Example
    StringBuilder stringBuilder = new StringBuilder("Hello");
    stringBuilder.append(" World");
    System.out.println("StringBuilder Example:");
    System.out.println("StringBuilder: " + stringBuilder);

    // StringBuffer Example
    StringBuffer stringBuffer = new StringBuffer("Hello");
    stringBuffer.append(" World");
    System.out.println("\nStringBuffer Example:");
    System.out.println("StringBuffer: " + stringBuffer);

    // Performance Comparison
```

```
        int iterations = 1000000;

        long startTimeStringBuilder = System.currentTimeMillis();
        StringBuilder stringBuilderConcat = new StringBuilder();
        for (int i = 0; i < iterations; i++) {
            stringBuilderConcat.append("Hello");
        }
        long endTimeStringBuilder = System.currentTimeMillis();
        long durationStringBuilder = endTimeStringBuilder - startTimeStringBuilder;
        System.out.println("\nPerformance Comparison:");
        System.out.println("StringBuilder Time: " + durationStringBuilder + " milliseconds");

        long startTimeStringBuffer = System.currentTimeMillis();
        StringBuffer stringBufferConcat = new StringBuffer();
        for (int i = 0; i < iterations; i++) {
            stringBufferConcat.append("Hello");
        }
        long endTimeStringBuffer = System.currentTimeMillis();
        long durationStringBuffer = endTimeStringBuffer - startTimeStringBuffer;
        System.out.println("StringBuffer Time: " + durationStringBuffer + " milliseconds");
    }
}
```

Arrays

1) Java Array for printing first loop from 1 to 9 and second loop from 9 to 1 and store them in matrix A and matrix B

```
public class MatrixPopulate {
    public static void main(String[] args) {
        // Define the size of the matrices
        int rows = 9;
        int cols = 9;
        // Define matrices A and B
        int[][] matrixA = new int[rows][cols];
        int[][] matrixB = new int[rows][cols];

        // Populate matrix A with the first loop (1 to 9)
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrixA[i][j] = i * cols + j + 1;
            }
        }

        // Populate matrix B with the second loop (9 to 1)
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrixB[i][j] = (rows - i) * cols - j;
            }
        }

        // Print matrix A
        System.out.println("Matrix A:");
        printMatrix(matrixA);

        // Print matrix B
        System.out.println("\nMatrix B:");
        printMatrix(matrixB);
    }

    // Method to print a matrix
    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int num : row) {
                System.out.print(num + "\t");
            }
```

```
   System.out.println();
  }
 }
}
```

2) **Java program to create confusion matrix and calculate TP (True Positive), TN (True Negative), FP (False Positive), FN (False Negative), and F1-score:**

```
import java.util.Arrays;

public class ConfusionMatrix {
 public static void main(String[] args) {
  // Actual values and Predicted values arrays
  int[] actualValues = {1, 0, 1, 1, 0, 1, 0, 0, 1, 1};
  int[] predictedValues = {1, 1, 1, 1, 0, 0, 0, 1, 0, 1};

  // Calculate confusion matrix
  int TP = 0, TN = 0, FP = 0, FN = 0;
  for (int i = 0; i < actualValues.length; i++) {
   if (actualValues[i] == 1 && predictedValues[i] == 1) {
    TP++;
   } else if (actualValues[i] == 0 && predictedValues[i] == 0) {
    TN++;
   } else if (actualValues[i] == 0 && predictedValues[i] == 1) {
    FP++;
   } else if (actualValues[i] == 1 && predictedValues[i] == 0) {
    FN++;
   }
  }

  // Calculate precision, recall, and F1-score
  double precision = (double) TP / (TP + FP);
  double recall = (double) TP / (TP + FN);
  double f1Score = 2 * (precision * recall) / (precision + recall);

  // Print confusion matrix and F1-score
  System.out.println("Confusion Matrix:");
  System.out.println("TP: " + TP);
  System.out.println("TN: " + TN);
  System.out.println("FP: " + FP);
  System.out.println("FN: " + FN);
  System.out.println("F1-Score: " + f1Score);
 }
}
```

3) **Write a program using Arrays class in java for creating 2D matrix**

```
import java.util.Arrays;

public class MatrixCreation {
 public static void main(String[] args) {
    // Define the dimensions of the matrix
    int rows = 3;
    int cols = 3;

    // Create a 2D matrix using the Array class
    int[][] matrix = (int[][]) Array.newInstance(int.class, rows, cols);

    // Populate the matrix with values
    for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
       matrix[i][j] = i * cols + j + 1; // Example: For a 3x3 matrix, values will start from 1 and go
up to 9
      }
    }

    // Print the matrix
    for (int[] row : matrix) {
      System.out.println(Arrays.toString(row));
    }
  }
}
```

4) Java program to find if 2 arrays have equal length, if not make it equal according to the smallest array among the both, use Arrays class and copyOf() function.

```
import java.util.Arrays;

public class EqualizeArrays {
  public static void main(String[] args) {
    int[] array1 = {1, 2, 3, 4, 5};
    int[] array2 = {1, 2, 3};

    System.out.println("Before equalization:");
    System.out.println("Array1: " + Arrays.toString(array1));
    System.out.println("Array2: " + Arrays.toString(array2));

    // Check if arrays have equal length
    if (array1.length != array2.length) {
      // Make them equal according to the smallest array length
      int minSize = Math.min(array1.length, array2.length);
      if (array1.length > minSize) {
        array1 = Arrays.copyOf(array1, minSize);
      } else {
        array2 = Arrays.copyOf(array2, minSize);
      }
    }

    System.out.println("\nAfter equalization:");
    System.out.println("Array1: " + Arrays.toString(array1));
    System.out.println("Array2: " + Arrays.toString(array2));
  }
}
```

Conditional Statements

1) **Find greatest of 3 numbers in java, without using If statement**
   Hint: Ternary operators

```
public class GreatestOfThree {
  public static void main(String[] args) {
    int num1 = 10;
    int num2 = 20;
    int num3 = 15;

    int greatest = num1 > num2 ? (num1 > num3 ? num1 : num3) : (num2 > num3 ? num2 : num3);

    System.out.println("The greatest number is: " + greatest);
  }
}
```

2) **Use nested if to find which if statement contains the greatest value**

```
public class GreatestInNestedIf {
  public static void main(String[] args) {
    int num1 = 10;
    int num2 = 20;
    int num3 = 15;

    if (num1 > num2) {
      if (num1 > num3) {
        System.out.println("The greatest value is in the first if statement: " + num1);
      } else {
        System.out.println("The greatest value is in the third if statement: " + num3);
      }
    } else {
      if (num2 > num3) {
        System.out.println("The greatest value is in the second if statement: " + num2);
      } else {
        System.out.println("The greatest value is in the third if statement: " + num3);
      }
    }
  }
}
```

3) **If a person is age 28 when his younger brother is 24, what will be his age when the older brother is 56. If age difference is more than 3 then find the age of younger brother when older brother was 5.**

```
public class AgeCalculation {

  public static void main(String[] args) {
        int personAgeAt28 = 28;
        int youngerBrotherAgeAt28 = 24;

        // Find the age difference
        int ageDifference = personAgeAt28 - youngerBrotherAgeAt28;

        // Calculate the age of the person when the older brother is 56
        int personAgeAt56 = 56 + ageDifference;

        // Check if the age difference is more than 3
        if (ageDifference > 3) {
            // Calculate the age of the younger brother when the older brother was 5
            int youngerBrotherAgeAt5 = youngerBrotherAgeAt28 - ageDifference + 5;
            System.out.println("The age of the younger brother when the older brother was 5: " + youngerBrotherAgeAt5);
        } else {
            // Print the age of the person when the older brother is 56
            System.out.println("The age of the person when the older brother is 56: " + personAgeAt56);
        }
    }
}
```

4) **Find a person's birth year based on their eligibility to vote, given that the current year is 2073 and the eligibility age is 21 years, we first need to calculate the birth year. If the birth year is more than 2060, we then find the years between the person's vote-eligible year and 2023 else find median of birth year and 2023**

```
public class BirthYear {
    public static void main(String[] args) {
        int currentYear = 2073;
        int eligibilityAge = 21;
        int voteEligibleYear = currentYear - eligibilityAge;

        if (voteEligibleYear > 2060) {
            System.out.println("Birth year is more than 2060.");
        System.out.println("Years between vote-eligible year and 2023:");

        for (int year = voteEligibleYear; year >= 2023; year--) {

          System.out.println(year);

        }

      } else {

        int birthYear = currentYear - eligibilityAge;

        System.out.println("Birth year: " + birthYear);

      }

    }

  }
```

Looping statements

1) **Use nested if to find which loop contains the greatest value**

```
public class GreatestInLoop {
  public static void main(String[] args) {
    int[] loop1 = {10, 20, 30};
    int[] loop2 = {15, 25, 35};

    int maxLoop1 = Integer.MIN_VALUE;
    int maxLoop2 = Integer.MIN_VALUE;

    // Find the maximum value in loop1
    for (int num : loop1) {
      if (num > maxLoop1) {
        maxLoop1 = num;
      }
    }

    // Find the maximum value in loop2
    for (int num : loop2) {
      if (num > maxLoop2) {
        maxLoop2 = num;
      }
    }

    // Compare the maximum values
    if (maxLoop1 > maxLoop2) {
      System.out.println("Loop 1 contains the greatest value: " + maxLoop1);
    } else if (maxLoop1 < maxLoop2) {
      System.out.println("Loop 2 contains the greatest value: " + maxLoop2);
    } else {
      System.out.println("Both loops contain the same greatest value: " + maxLoop1);
    }
    }
  }
```

2) **Find missing numbers in the series 1,5,11,19 using java**

```
public class MissingNumbersSeries {
  public static void main(String[] args) {
    int[] series = {1, 5, 11, 19};

    System.out.println("Missing numbers in the series:");
    for (int i = 0; i < series.length - 1; i++) {
      int diff = series[i + 1] - series[i]; // Calculate the difference between consecutive
numbers
      if (diff > 1) {
        for (int j = 1; j < diff; j++) {
          System.out.println(series[i] + j); // Print the missing numbers
        }
      }
    }
  }
}
```

3) **write java program to iterate through array elements using enhanced for loop. And find no. of prime numbers**

```
public class PrimeNumbersInArray {


  // Function to check if a number is prime

  public static boolean isPrime(int n) {

    if (n <= 1) {

      return false;

    }

    for (int i = 2; i <= Math.sqrt(n); i++) {

      if (n % i == 0) {

        return false;

      }

    }

    return true;

  }


  public static void main(String[] args) {
        int[] array = {7, 8, 11, 15, 17, 20, 23};

        int primeCount = 0;

        System.out.println("Array elements:");
        for (int num : array) {
            System.out.print(num + " ");
            if (isPrime(num)) {
                primeCount++;
            }
        }

        System.out.println("\nNumber of prime numbers in the array: " + primeCount);
    }
}
```

4) Iterate to two for loops, both contains 10 array elements, one is normal order, another in reverse order, find the median where they will meet

```
public class MedianAtMeetingPoint {
    public static void main(String[] args) {
        int[] normalOrder = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
        int[] reverseOrder = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

        int median = -1; // Initialize median to a default value

        // Iterate through both arrays simultaneously
        for (int i = 0, j = reverseOrder.length - 1; i < normalOrder.length && j >= 0; i++, j--) {
            // If the current element from normalOrder array is greater than or equal to the
current element
            // from reverseOrder array, we have reached or passed the meeting point of the
median
            if (normalOrder[i] >= reverseOrder[j]) {
                median = normalOrder[i]; // Set the median value
                break; // Exit the loop
            }
        }

        System.out.println("Median at meeting point: " + median);
    }
}
```

Switch Case and operators

1) **Write java program using switch case to find luck guess**

```
import java.util.Scanner;

public class LuckyGuess {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Welcome to the Lucky Guess game!");
        System.out.println("Guess a number between 1 and 5:");

        int luckyNumber = (int) (Math.random() * 5) + 1; // Generate a random number
between 1 and 5
        int guess = scanner.nextInt(); // Get user's guess

        switch (guess) {
            case 1:
            case 2:
            case 3:
            case 4:
            case 5:
                if (guess == luckyNumber) {
                    System.out.println("Congratulations! You guessed the lucky number: " +
luckyNumber);
                } else {
                    System.out.println("Sorry! Better luck next time. The lucky number was: " +
luckyNumber);
                }
                break;
            default:
                System.out.println("Invalid guess! Please guess a number between 1 and 5.");
        }

        scanner.close();
    }
}
```

2) **Java switch case to find which for loop is shortest path first**

```
public class ShortestPath {
  public static void main(String[] args) {
    int[] loop1 = {1, 2, 3, 4, 5};
    int[] loop2 = {10, 20, 30, 40, 50, 60};

    int loop1Iterations = loop1.length;
    int loop2Iterations = loop2.length;

    switch (Integer.compare(loop1Iterations, loop2Iterations)) {
      case -1:
        System.out.println("Loop 1 is the shortest path.");
        break;
      case 0:
        System.out.println("Both loops have the same number of iterations.");
        break;
      case 1:
        System.out.println("Loop 2 is the shortest path.");
        break;
    }
  }
}
```

3) **Write a java program to implement OR gate and AND gate**

```
public class LogicalGates {


  // OR gate implementation

  public static boolean OR(boolean input1, boolean input2) {

    return input1 || input2;

  }


  // AND gate implementation

  public static boolean AND(boolean input1, boolean input2) {

    return input1 && input2;

  }


  public static void main(String[] args) {

    // Testing OR gate
```

```
        boolean input1 = true;
        boolean input2 = false;
        System.out.println("OR gate:");
        System.out.println(input1 + " OR " + input2 + " = " + OR(input1, input2));

        // Testing AND gate
        input1 = true;
        input2 = false;
        System.out.println("\nAND gate:");
        System.out.println(input1 + " AND " + input2 + " = " + AND(input1, input2));
      }
    }
```

4) **Write a program to shift values from left to right if A>B else shift right to left using logical shift operators. Where A and B are arrays**

```
public class ArrayShift {
  public static void main(String[] args) {
    int[] A = {5, 8, 3, 9, 2};
    int[] B = {3, 6, 2, 7, 4};

    if (A.length != B.length) {
      System.out.println("Arrays A and B must have the same length.");
      return;
    }

    // Check each pair of elements in arrays A and B
    for (int i = 0; i < A.length; i++) {
      if (A[i] > B[i]) {
        // Shift elements from left to right
        for (int j = 0; j < A.length - 1; j++) {
          A[j] = A[j + 1];
          B[j] = B[j + 1];
```

```
                }
                A[A.length - 1] = 0;
                B[B.length - 1] = 0;
            } else {
                // Shift elements from right to left
                for (int j = A.length - 1; j > 0; j--) {
                    A[j] = A[j - 1];
                    B[j] = B[j - 1];
                }
                A[0] = 0;
                B[0] = 0;
            }
        }

        // Print the shifted arrays
        System.out.println("Shifted arrays:");
        System.out.println("A: " + java.util.Arrays.toString(A));
        System.out.println("B: " + java.util.Arrays.toString(B));
    }
}
```