

1. (20 Points) The Java class below reads grades from the user, counts and sums the grades until a negative number is entered. The code then computes and prints the count and the average of the grades. The class has 10 errors. Rewrite and fix all the errors in the class.

```
import java.util.Scanner;
public class ErrorProne
    public static void main(String[] args)
        Scanner scnr = new Scanner(System.in);
        int gradeSum = 0
        int gradeCount = 0
        grade = scnr.nextInt();

        while (grade >= 0) {
            gradeSum += grade;
            gradeCount--;
            grade = scnr.nextInt()
        }

        double gradeAverage = gradeSum/gradeCount;

        system.out.println("Number of Grades = " + gradeCount);
        System.out.println("Average of Grades = " , gradeAverage);
    }
}
```

Corrections:

```
import java.util.Scanner;
public class ErrorProne {
    public static void main(String[] args) {
        Scanner scnr = new Scanner(System.in);
        int gradeSum = 0;
        int gradeCount = 0;
        int grade = scnr.nextInt();

        while (grade >= 0) {
            gradeSum += grade;
            gradeCount++;

            grade = scnr.nextInt();
        }

        double gradeAverage = (double)gradeSum/(double)gradeCount;

        System.out.println("Number of Grades = " + gradeCount);
        System.out.println("Average of Grades = " + gradeAverage);
    }
}
```

2. (20 Points) What is the output of the following program?

```
public class Switch1 {
    public static void main(String[] args) {
        int[] arr = { 1, 3, 5, 7, 9, 8, 6, 4, 2 };

        for (int i = 0; i < arr.length; i++) {
            switch (i) {
                case 0:
                    System.out.println("i = " + i + " : " + (arr[i + 1] * arr[i + 2])); break;
                case 1:
                    System.out.println("i = " + i + " : " + (arr[i + 3] + arr[i + 3]));
                case 2:
                    System.out.println("i = " + i + " : " + (arr[i + 4] / arr[i + 6])); break;
                case 3:
                    System.out.println("i = " + i + " : " + (arr[i - 1] - arr[i - 3]));
                case 4:
                    System.out.println("i = " + i + " : " + (arr[i - 2] * arr[i - 2])); break;
                case 5:
                    System.out.println("i = " + i + " : " + (arr[i + 3] + arr[i + 3]));
                case 6:
                    System.out.println("i = " + i + " : " + (arr[i - 1] / arr[i + 2])); break;
                case 7:
                    System.out.println("i = " + i + " : " + (arr[i - 5] - arr[i + 1]));
                default:
                    System.out.println("i = " + i + " : " + (arr[i - 7] * arr[i - 4]));
            }
        }
    }
}
```

Output:

```
i = 0 : 15
i = 1 : 18
i = 1 : 2
i = 2 : 3
i = 3 : 4
i = 3 : 9
i = 4 : 25
i = 5 : 4
i = 5 : 2
i = 6 : 4
i = 7 : 3
i = 7 : 7
i = 8 : 27
```

3. (20 Points) What is the output of the following program?

```
public class BreakContinue1 {
    public static void main(String[] args) {
        for ( int i = 0 ; i <= 20 ; i += 5 ) {
            for ( int j = 5; j <= 25 ; j += 5 ) {
                if ( i < j) {
                    break;
                } else if ( i == j) {
                    continue;
                }
                System.out.println("i = " + i + " : " + "j = " + j);
            }
        }
    }
}
```

Output:

```
i = 10 : j = 5
i = 15 : j = 5
i = 15 : j = 10
i = 20 : j = 5
i = 20 : j = 10
i = 20 : j = 15
```

4. (20 Points) Write a method with the following signature:

```
public static int[] reverse(int[] arr) {
```

This method accepts an array of integers `arr` as a parameter. The method copies all the elements of `arr` into another array in reverse order. The method then returns the reversed array.

For Example, if `arr` is:

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

The method would return the following array:

```
{9, 8, 7, 6, 5, 4, 3, 2, 1}
```

```
public static int[] reverse(int[] arr) {
    int[] reversed = new int[arr.length];

    int rIndex = arr.length - 1;

    for ( int i = 0 ; i < arr.length ; i++ ) {
        reversed[rIndex] = arr[i];
        rIndex--;
    }

    return reversed;
}
```

5. (40 Points) Write a complete Java class named `ArrayMethods` that implements the methods listed below. All the methods accept the following parameters:

Parameters:

<code>int[] arr</code>	An array of <code>int</code> values.
<code>int firstIndex</code>	The index of the first element to include in the sum.
<code>int lastIndex</code>	The index of the last element to include in the sum.

Please note that all methods **must** verify the validity of `firstIndex` and `lastIndex`.

```
public static int sum(int[] arr, int firstIndex, int lastIndex)
```

This method will compute the sum of the elements in the parameter array `arr` starting at index `firstIndex` and ending at index `lastIndex`.

Return Value:

This method returns an `int` representing the computed sum of the elements in `arr` starting at `firstIndex` and ending at `lastIndex`. If either index is invalid, the method should return the value `-666`.

```
public static double average(int[] arr, int firstIndex, int lastIndex)
```

This method will compute the average of the elements in the parameter array `arr` starting at index `firstIndex` and ending at index `lastIndex`. You should make use of the `sum(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns a `double` value representing the computed average of the elements in `arr` starting at index `firstIndex` and ending at index `lastIndex`. If either index is invalid, the method should return the value `-666.0`.

```
public static int belowAverage(int[] arr, int firstIndex, int lastIndex)
```

This method will search the elements in the parameter array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, and count the number of elements that are below average. You should make use of the `average(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns the number of elements in the array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, that are below average. If either index is invalid, the method should return the value `-666`.

```
public static int aboveAverage(int[] arr, int firstIndex, int lastIndex)
```

This method will search the elements in the parameter array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, and count the number of elements that are above average. You should make use of the `average(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns the number of elements in the array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, that are above average. If either index is invalid, the method should return the value `-666`.

```
public class ArrayMethods {

    private static boolean validIndices(int[] arr, int firstIndex, int lastIndex) {
        return ((firstIndex >= 0) && (firstIndex < arr.length)) &&
            ((lastIndex >= 0) && (lastIndex < arr.length)) &&
            (firstIndex <= lastIndex);
    }

    public static int sum(int[] arr, int firstIndex, int lastIndex) {
        int sum = 0;
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            sum += arr[i];
        }
        return sum;
    }

    public static double average(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666.0;
        }
        return (double)sum(arr, firstIndex, lastIndex)/(lastIndex - firstIndex + 1);
    }

    public static int belowAverage(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        double average = average(arr, firstIndex, lastIndex);
        int countBelowAverage = 0;
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            if (arr[i] < average) {
                countBelowAverage++;
            }
        }
        return countBelowAverage;
    }

    public static int aboveAverage(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        double average = average(arr, firstIndex, lastIndex);
        int countAboveAverage = 0;
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            if (arr[i] > average) {
                countAboveAverage++;
            }
        }
        return countAboveAverage;
    }
}
```

1. (20 Points) The Java class below reads grades from the user, counts and sums the grades until a negative number is entered. The code then computes and prints the count and the average of the grades. The class has 10 errors. Rewrite and fix all the errors in the class.

```
import java.util.Scanner;
public class ErrorProne
    public static void main(String[] args)
        Scanner scnr = new Scanner(System.in);
        int gradeSum = 0
        int gradeCount = 0
        grade = scnr.nextInt();

        while (grade >= 0) {
            gradeSum += grade;
            gradeCount--;
            grade = scnr.nextInt()
        }

        double gradeAverage = gradeSum/gradeCount;

        system.out.println("Number of Grades = " + gradeCount);
        System.out.println("Average of Grades = " , gradeAverage);
    }
}
```

Corrections:

```
import java.util.Scanner;
public class ErrorProne {
    public static void main(String[] args) {
        Scanner scnr = new Scanner(System.in);
        int gradeSum = 0;
        int gradeCount = 0;
        int grade = scnr.nextInt();

        while (grade >= 0) {
            gradeSum += grade;
            gradeCount++;

            grade = scnr.nextInt();
        }

        double gradeAverage = (double)gradeSum/(double)gradeCount;

        System.out.println("Number of Grades = " + gradeCount);
        System.out.println("Average of Grades = " + gradeAverage);
    }
}
```

2. (20 Points) What is the output of the following program?

```
public class Switch2 {  
    public static void main(String[] args) {  
        int[] arr = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };  
  
        for (int i = 0; i < arr.length; i++) {  
            switch (i) {  
                case 0:  
                    System.out.println("i = " + i + " : " + (arr[i + 1] * arr[i + 2])); break;  
                case 1:  
                    System.out.println("i = " + i + " : " + (arr[i + 3] + arr[i + 3]));  
                case 2:  
                    System.out.println("i = " + i + " : " + (arr[i + 3] / arr[i - 1])); break;  
                case 3:  
                    System.out.println("i = " + i + " : " + (arr[i - 1] - arr[i - 3]));  
                case 4:  
                    System.out.println("i = " + i + " : " + (arr[i - 2] * arr[i - 2])); break;  
                case 5:  
                    System.out.println("i = " + i + " : " + (arr[i + 3] + arr[i + 3]));  
                case 6:  
                    System.out.println("i = " + i + " : " + (arr[i - 1] / arr[i - 4])); break;  
                case 7:  
                    System.out.println("i = " + i + " : " + (arr[i + 1] - arr[i - 4]));  
                default:  
                    System.out.println("i = " + i + " : " + (arr[i - 7] * arr[i - 4]));  
            }  
        }  
    }  
}
```

Output:

```
i = 0 : 6  
i = 1 : 10  
i = 1 : 5  
i = 2 : 3  
i = 3 : 2  
i = 3 : 4  
i = 4 : 9  
i = 5 : 18  
i = 5 : 2  
i = 6 : 2  
i = 7 : 5  
i = 7 : 4  
i = 8 : 10
```

3. (20 Points) What is the output of the following program?

```
public class BreakContinue2 {
    public static void main(String[] args) {
        for ( int i = 0 ; i <= 8 ; i += 2 ) {
            for ( int j = 2; j <= 10 ; j += 2 ) {
                if ( i < j) {
                    break;
                } else if ( i == j) {
                    continue;
                }
                System.out.println("i = " + i + " : " + "j = " + j);
            }
        }
    }
}
```

Output:

```
i = 4 : j = 2
i = 6 : j = 2
i = 6 : j = 4
i = 8 : j = 2
i = 8 : j = 4
i = 8 : j = 6
```

4. (20 Points) Write a method with the following signature:

```
public static int[] reverse(int[] arr) {
```

This method accepts an array of integers `arr` as a parameter. The method copies all the elements of `arr` into another array in reverse order. The method then returns the reversed array.

For Example, if `arr` is:

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

The method would return the following array:

```
{9, 8, 7, 6, 5, 4, 3, 2, 1}
```

```
public static int[] reverse(int[] arr) {
    int[] reversed = new int[arr.length];

    int rIndex = arr.length - 1;

    for ( int i = 0 ; i < arr.length ; i++ ) {
        reversed[rIndex] = arr[i];
        rIndex--;
    }

    return reversed;
}
```

5. (40 Points) Write a complete Java class named `ArrayMethods` that implements the methods listed below. All the methods accept the following parameters:

Parameters:

<code>int[] arr</code>	An array of <code>int</code> values.
<code>int firstIndex</code>	The index of the first element to include in the sum.
<code>int lastIndex</code>	The index of the last element to include in the sum.

Please note that all methods **must** verify the validity of `firstIndex` and `lastIndex`.

```
public static int sum(int[] arr, int firstIndex, int lastIndex)
```

This method will compute the sum of the elements in the parameter array `arr` starting at index `firstIndex` and ending at index `lastIndex`.

Return Value:

This method returns an `int` representing the computed sum of the elements in `arr` starting at `firstIndex` and ending at `lastIndex`. If either index is invalid, the method should return the value `-666`.

```
public static double average(int[] arr, int firstIndex, int lastIndex)
```

This method will compute the average of the elements in the parameter array `arr` starting at index `firstIndex` and ending at index `lastIndex`. You should make use of the `sum(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns a `double` value representing the computed average of the elements in `arr` starting at index `firstIndex` and ending at index `lastIndex`. If either index is invalid, the method should return the value `-666.0`.

```
public static int belowAverage(int[] arr, int firstIndex, int lastIndex)
```

This method will search the elements in the parameter array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, and count the number of elements that are below average. You should make use of the `average(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns the number of elements in the array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, that are below average. If either index is invalid, the method should return the value `-666`.

```
public static int aboveAverage(int[] arr, int firstIndex, int lastIndex)
```

This method will search the elements in the parameter array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, and count the number of elements that are above average. You should make use of the `average(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns the number of elements in the array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, that are above average. If either index is invalid, the method should return the value `-666`.

```
public class ArrayMethods {

    private static boolean validIndices(int[] arr, int firstIndex, int lastIndex) {
        return ((firstIndex >= 0) && (firstIndex < arr.length)) &&
            ((lastIndex >= 0) && (lastIndex < arr.length)) &&
            (firstIndex <= lastIndex);
    }

    public static int sum(int[] arr, int firstIndex, int lastIndex) {
        int sum = 0;
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            sum += arr[i];
        }
        return sum;
    }

    public static double average(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666.0;
        }
        return (double)sum(arr, firstIndex, lastIndex)/(lastIndex - firstIndex + 1);
    }

    public static int belowAverage(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        double average = average(arr, firstIndex, lastIndex);
        int countBelowAverage = 0;
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            if (arr[i] < average) {
                countBelowAverage++;
            }
        }
        return countBelowAverage;
    }

    public static int aboveAverage(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        double average = average(arr, firstIndex, lastIndex);
        int countAboveAverage = 0;
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            if (arr[i] > average) {
                countAboveAverage++;
            }
        }
        return countAboveAverage;
    }
}
```

1. (20 Points) The Java class below reads grades from the user, counts and sums the grades until a negative number is entered. The code then computes and prints the count and the average of the grades. The class has 10 errors. Rewrite and fix all the errors in the class.

```
import java.util.Scanner;
public class ErrorProne
    public static void main(String[] args)
        Scanner scnr = new Scanner(System.in);
        int gradeSum = 0
        int gradeCount = 0
        grade = scnr.nextInt();

        while (grade >= 0) {
            gradeSum += grade;
            gradeCount--;
            grade = scnr.nextInt()
        }

        double gradeAverage = gradeSum/gradeCount;

        system.out.println("Number of Grades = " + gradeCount);
        System.out.println("Average of Grades = " , gradeAverage);
    }
}
```

Corrections:

```
import java.util.Scanner;
public class ErrorProne {
    public static void main(String[] args) {
        Scanner scnr = new Scanner(System.in);
        int gradeSum = 0;
        int gradeCount = 0;
        int grade = scnr.nextInt();

        while (grade >= 0) {
            gradeSum += grade;
            gradeCount++;

            grade = scnr.nextInt();
        }

        double gradeAverage = (double)gradeSum/(double)gradeCount;

        System.out.println("Number of Grades = " + gradeCount);
        System.out.println("Average of Grades = " + gradeAverage);
    }
}
```

2. (20 Points) What is the output of the following program?

```
public class Switch3 {
    public static void main(String[] args) {
        int[] arr = { 2, 4, 6, 8, 9, 7, 5, 3, 1 };

        for (int i = 0; i < arr.length; i++) {
            switch (i) {
                case 0:
                    System.out.println("i = " + i + " : " + (arr[i + 1] * arr[i + 2])); break;
                case 1:
                    System.out.println("i = " + i + " : " + (arr[i + 3] + arr[i + 3]));
                case 2:
                    System.out.println("i = " + i + " : " + (arr[i + 2] / arr[i + 5])); break;
                case 3:
                    System.out.println("i = " + i + " : " + (arr[i - 1] - arr[i - 3]));
                case 4:
                    System.out.println("i = " + i + " : " + (arr[i - 2] * arr[i - 2])); break;
                case 5:
                    System.out.println("i = " + i + " : " + (arr[i + 3] + arr[i + 3]));
                case 6:
                    System.out.println("i = " + i + " : " + (arr[i - 1] / arr[i - 4])); break;
                case 7:
                    System.out.println("i = " + i + " : " + (arr[i - 2] - arr[i - 6]));
                default:
                    System.out.println("i = " + i + " : " + (arr[i - 7] * arr[i - 4]));
            }
        }
    }
}
```

Output:

```
i = 0 : 24
i = 1 : 18
i = 1 : 1
i = 2 : 3
i = 3 : 4
i = 3 : 16
i = 4 : 36
i = 5 : 2
i = 5 : 2
i = 6 : 1
i = 7 : 3
i = 7 : 16
i = 8 : 36
```

3. (20 Points) What is the output of the following program?

```
public class BreakContinue3 {
    public static void main(String[] args) {
        for ( int i = 0 ; i <= 12 ; i += 3 ) {
            for ( int j = 3; j <= 15 ; j += 3 ) {
                if ( i < j) {
                    break;
                } else if ( i == j) {
                    continue;
                }
                System.out.println("i = " + i + " : " + "j = " + j);
            }
        }
    }
}
```

Output:

```
i = 6 : j = 3
i = 9 : j = 3
i = 9 : j = 6
i = 12 : j = 3
i = 12 : j = 6
i = 12 : j = 9
```

4. (20 Points) Write a method with the following signature:

```
public static int[] reverse(int[] arr) {
```

This method accepts an array of integers `arr` as a parameter. The method copies all the elements of `arr` into another array in reverse order. The method then returns the reversed array.

For Example, if `arr` is:

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

The method would return the following array:

```
{9, 8, 7, 6, 5, 4, 3, 2, 1}
```

```
public static int[] reverse(int[] arr) {
    int[] reversed = new int[arr.length];

    int rIndex = arr.length - 1;

    for ( int i = 0 ; i < arr.length ; i++ ) {
        reversed[rIndex] = arr[i];
        rIndex--;
    }

    return reversed;
}
```

5. (40 Points) Write a complete Java class named `ArrayMethods` that implements the methods listed below. All the methods accept the following parameters:

Parameters:

<code>int[] arr</code>	An array of <code>int</code> values.
<code>int firstIndex</code>	The index of the first element to include in the sum.
<code>int lastIndex</code>	The index of the last element to include in the sum.

Please note that all methods **must** verify the validity of `firstIndex` and `lastIndex`.

```
public static int sum(int[] arr, int firstIndex, int lastIndex)
```

This method will compute the sum of the elements in the parameter array `arr` starting at index `firstIndex` and ending at index `lastIndex`.

Return Value:

This method returns an `int` representing the computed sum of the elements in `arr` starting at `firstIndex` and ending at `lastIndex`. If either index is invalid, the method should return the value `-666`.

```
public static double average(int[] arr, int firstIndex, int lastIndex)
```

This method will compute the average of the elements in the parameter array `arr` starting at index `firstIndex` and ending at index `lastIndex`. You should make use of the `sum(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns a `double` value representing the computed average of the elements in `arr` starting at index `firstIndex` and ending at index `lastIndex`. If either index is invalid, the method should return the value `-666.0`.

```
public static int belowAverage(int[] arr, int firstIndex, int lastIndex)
```

This method will search the elements in the parameter array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, and count the number of elements that are below average. You should make use of the `average(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns the number of elements in the array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, that are below average. If either index is invalid, the method should return the value `-666`.

```
public static int aboveAverage(int[] arr, int firstIndex, int lastIndex)
```

This method will search the elements in the parameter array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, and count the number of elements that are above average. You should make use of the `average(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns the number of elements in the array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, that are above average. If either index is invalid, the method should return the value `-666`.

```
public class ArrayMethods {

    private static boolean validIndices(int[] arr, int firstIndex, int lastIndex) {
        return ((firstIndex >= 0) && (firstIndex < arr.length)) &&
            ((lastIndex >= 0) && (lastIndex < arr.length)) &&
            (firstIndex <= lastIndex);
    }

    public static int sum(int[] arr, int firstIndex, int lastIndex) {
        int sum = 0;
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            sum += arr[i];
        }
        return sum;
    }

    public static double average(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666.0;
        }
        return (double)sum(arr, firstIndex, lastIndex)/(lastIndex - firstIndex + 1);
    }

    public static int belowAverage(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        double average = average(arr, firstIndex, lastIndex);
        int countBelowAverage = 0;
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            if (arr[i] < average) {
                countBelowAverage++;
            }
        }
        return countBelowAverage;
    }

    public static int aboveAverage(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        double average = average(arr, firstIndex, lastIndex);
        int countAboveAverage = 0;
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            if (arr[i] > average) {
                countAboveAverage++;
            }
        }
        return countAboveAverage;
    }
}
```

1. (20 Points) The Java class below reads grades from the user, counts and sums the grades until a negative number is entered. The code then computes and prints the count and the average of the grades. The class has 10 errors. Rewrite and fix all the errors in the class.

```
import java.util.Scanner;
public class ErrorProne
    public static void main(String[] args)
        Scanner scnr = new Scanner(System.in);
        int gradeSum = 0
        int gradeCount = 0
        grade = scnr.nextInt();

        while (grade >= 0) {
            gradeSum += grade;
            gradeCount--;
            grade = scnr.nextInt()
        }

        double gradeAverage = gradeSum/gradeCount;

        system.out.println("Number of Grades = " + gradeCount);
        System.out.println("Average of Grades = " , gradeAverage);
    }
}
```

Corrections:

```
import java.util.Scanner;
public class ErrorProne {
    public static void main(String[] args) {
        Scanner scnr = new Scanner(System.in);
        int gradeSum = 0;
        int gradeCount = 0;
        int grade = scnr.nextInt();

        while (grade >= 0) {
            gradeSum += grade;
            gradeCount++;

            grade = scnr.nextInt();
        }

        double gradeAverage = (double)gradeSum/(double)gradeCount;

        System.out.println("Number of Grades = " + gradeCount);
        System.out.println("Average of Grades = " + gradeAverage);
    }
}
```

2. (20 Points) What is the output of the following program?

```
public class Switch4 {
    public static void main(String[] args) {
        int[] arr = { 8, 6, 4, 2, 1, 3, 5, 7, 9 };

        for (int i = 0; i < arr.length; i++) {
            switch (i) {
                case 0:
                    System.out.println("i = " + i + " : " + (arr[i + 1] * arr[i + 2])); break;
                case 1:
                    System.out.println("i = " + i + " : " + (arr[i + 3] + arr[i + 2]));
                case 2:
                    System.out.println("i = " + i + " : " + (arr[i - 1] / arr[i + 2])); break;
                case 3:
                    System.out.println("i = " + i + " : " + (arr[i - 1] - arr[i + 2]));
                case 4:
                    System.out.println("i = " + i + " : " + (arr[i - 2] * arr[i - 2])); break;
                case 5:
                    System.out.println("i = " + i + " : " + (arr[i + 3] + arr[i + 3]));
                case 6:
                    System.out.println("i = " + i + " : " + (arr[i - 1] / arr[i - 4])); break;
                case 7:
                    System.out.println("i = " + i + " : " + (arr[i + 1] - arr[i - 6]));
                default:
                    System.out.println("i = " + i + " : " + (arr[i - 7] * arr[i - 4]));
            }
        }
    }
}
```

Output:

```
i = 0 : 24
i = 1 : 3
i = 1 : 4
i = 2 : 6
i = 3 : 1
i = 3 : 36
i = 4 : 16
i = 5 : 18
i = 5 : 0
i = 6 : 0
i = 7 : 3
i = 7 : 16
i = 8 : 6
```

3. (20 Points) What is the output of the following program?

```
public class BreakContinue4 {
    public static void main(String[] args) {
        for ( int i = 0 ; i <= 16 ; i += 4 ) {
            for ( int j = 4; j <= 20 ; j += 4 ) {
                if ( i < j) {
                    break;
                } else if ( i == j) {
                    continue;
                }
                System.out.println("i = " + i + " : " + "j = " + j);
            }
        }
    }
}
```

Output:

```
i = 8 : j = 4
i = 12 : j = 4
i = 12 : j = 8
i = 16 : j = 4
i = 16 : j = 8
i = 16 : j = 12
```

4. (20 Points) Write a method with the following signature:

```
public static int[] reverse(int[] arr) {
```

This method accepts an array of integers `arr` as a parameter. The method copies all the elements of `arr` into another array in reverse order. The method then returns the reversed array.

For Example, if `arr` is:

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

The method would return the following array:

```
{9, 8, 7, 6, 5, 4, 3, 2, 1}
```

```
public static int[] reverse(int[] arr) {
    int[] reversed = new int[arr.length];

    int rIndex = arr.length - 1;

    for ( int i = 0 ; i < arr.length ; i++ ) {
        reversed[rIndex] = arr[i];
        rIndex--;
    }

    return reversed;
}
```

5. (40 Points) Write a complete Java class named `ArrayMethods` that implements the methods listed below. All the methods accept the following parameters:

Parameters:

<code>int[] arr</code>	An array of <code>int</code> values.
<code>int firstIndex</code>	The index of the first element to include in the sum.
<code>int lastIndex</code>	The index of the last element to include in the sum.

Please note that all methods **must** verify the validity of `firstIndex` and `lastIndex`.

```
public static int sum(int[] arr, int firstIndex, int lastIndex)
```

This method will compute the sum of the elements in the parameter array `arr` starting at index `firstIndex` and ending at index `lastIndex`.

Return Value:

This method returns an `int` representing the computed sum of the elements in `arr` starting at `firstIndex` and ending at `lastIndex`. If either index is invalid, the method should return the value `-666`.

```
public static double average(int[] arr, int firstIndex, int lastIndex)
```

This method will compute the average of the elements in the parameter array `arr` starting at index `firstIndex` and ending at index `lastIndex`. You should make use of the `sum(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns a `double` value representing the computed average of the elements in `arr` starting at index `firstIndex` and ending at index `lastIndex`. If either index is invalid, the method should return the value `-666.0`.

```
public static int belowAverage(int[] arr, int firstIndex, int lastIndex)
```

This method will search the elements in the parameter array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, and count the number of elements that are below average. You should make use of the `average(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns the number of elements in the array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, that are below average. If either index is invalid, the method should return the value `-666`.

```
public static int aboveAverage(int[] arr, int firstIndex, int lastIndex)
```

This method will search the elements in the parameter array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, and count the number of elements that are above average. You should make use of the `average(int[] arr, int firstIndex, int lastIndex)` method in your computation.

Return Value:

This method returns the number of elements in the array `arr`, starting at index `firstIndex` and ending at index `lastIndex`, that are above average. If either index is invalid, the method should return the value `-666`.

```
public class ArrayMethods {

    private static boolean validIndices(int[] arr, int firstIndex, int lastIndex) {
        return ((firstIndex >= 0) && (firstIndex < arr.length)) &&
            ((lastIndex >= 0) && (lastIndex < arr.length)) &&
            (firstIndex <= lastIndex);
    }

    public static int sum(int[] arr, int firstIndex, int lastIndex) {
        int sum = 0;
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            sum += arr[i];
        }
        return sum;
    }

    public static double average(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666.0;
        }
        return (double)sum(arr, firstIndex, lastIndex)/(lastIndex - firstIndex + 1);
    }

    public static int belowAverage(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        double average = average(arr, firstIndex, lastIndex);
        int countBelowAverage = 0;
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            if (arr[i] < average) {
                countBelowAverage++;
            }
        }
        return countBelowAverage;
    }

    public static int aboveAverage(int[] arr, int firstIndex, int lastIndex) {
        if (!validIndices(arr, firstIndex, lastIndex)) {
            return -666;
        }
        double average = average(arr, firstIndex, lastIndex);
        int countAboveAverage = 0;
        for (int i = firstIndex ; i <= lastIndex ; i++ ) {
            if (arr[i] > average) {
                countAboveAverage++;
            }
        }
        return countAboveAverage;
    }
}
```