ECON381 Fall 2024
Midterm Exam 06.12.2024

Student Number		**Grading**	
Name and Surname		**Questions 1-4**	…... out of 20
Student Signature		**Question 5**	…... out of 20
		Question 6	…... out of 20
Notes: 1. Part 1 is composed of Questions 1-4 (Multiple choice, simple programming review) 2. Part 2 is composed of Questions 5-8 (Applied, more about algorithms and data structures) 3. Please write your name on the corner of each page. Also if you need more space, feel free to use the back of pages.		**Question 7**	…... out of 20
		Question 8	…… out of 20.
		Total	…… out of 100.

PART 1
Questions 1-4 are multiple choice (Each 5 points)

Question 1
Consider the following Python code:

```
x = 10
```

Which of the following is the equivalent Java code?

(a) int x = 10;
(b) var x = 10;
(c) x = 10;
(d) x: int = 10;

Question 2
Analyze the following Java code (It contains an error):

```
public class Main {
    public static int square(int n) {
        int result = n * n;
    }
    public static void main(String[] args) {
        System.out.println(square(4));
    }     }
```

What error will occur when this code is compiled and run?

(a) The function is missing a return statement.
(b) The function cannot calculate n * n.

(c) The main method is incorrectly defined.
(d) There is no error; the code runs successfully.

Question 3

Consider the following Java code:

```
public class Main {
   public static void main(String[] args) {
      int[] arr = new int[5];
      for (int i = 0; i < arr.length; i++) {
         arr[i] = 9 - (i * 2);  }
      for (int i = 0; i < arr.length; i++) {
         System.out.print(arr[i] + " ");
      }     }     }
```

What will be the output of this code?
(a) 9 7 5 3 1
(b) 1 3 5 7 9
(c) 8 6 4 2 0
(d) 10 8 6 4 2

Question 4

Consider the following Java code:

```
public class Main {
   public static void main(String[] args) {
      int[] arr1 = {1, 2, 3};
      int[] arr2 = {10, 20};
      int[] result = new int[arr1.length + arr2.length];
      for (int i = 0; i < arr1.length; i++) {
         result[i * 2] = arr1[i];
         result[i * 2 + 1] = arr2[i];  }
      for (int i = 0; i < result.length; i++) {
         System.out.print(result[i] + " ");
      }       }     }
```

What will happen when this code is executed?
(a) Program outputs: 1 10 2 20 3 0
(b) Progr am outputs: 1 10 2 20
(c) Program outputs: 10 1 20 2 3
(d) There is an error in the code.

For questions 5 to 8 refer to the following Java code.
It is an implementation of Merge Sort. It does not contain any errors.

```
public class NonRecursiveMergeSort {

    // Main method to sort an array using Non-Recursive Merge Sort
    public static void mergeSort(int[] array) {
        if (array == null || array.length < 2) {
            return; // Array is already sorted if it has less than 2 elements
        }

        int n = array.length;
        int[] tempArray = new int[n];

        for (int size = 1; size < n; size *= 2) {
            for (int leftStart = 0; leftStart < n - size; leftStart += 2 * size) {
                int mid = leftStart + size - 1;
                int rightEnd = Math.min(leftStart + 2 * size - 1, n - 1);
                merge(array, tempArray, leftStart, mid, rightEnd);
            }
        }
    }

    // Merge two sorted halves into a single sorted section
    private static void merge(int[] array, int[] tempArray, int left, int mid, int right) {
        System.arraycopy(array, left, tempArray, left, right - left + 1);

        int i = left;      // Pointer for the left half
        int j = mid + 1;   // Pointer for the right half
        int k = left;      // Pointer for the merged array

        while (i <= mid && j <= right) {
            if (tempArray[i] <= tempArray[j]) {
                array[k++] = tempArray[i++];
            } else {
                array[k++] = tempArray[j++];
            }
        }

        // Copy any remaining elements from the left half
        while (i <= mid) {
            array[k++] = tempArray[i++];
        }

        // Right half remaining elements are already in place
    }
    // Main method to test the algorithm
    public static void main(String[] args) {
        int[] array = {38, 27, 43, 3, 9, 82, 10};

        System.out.println("Original Array:");
        for (int num : array) {
            System.out.print(num + " ");
        }

        mergeSort(array);

        System.out.println("\n\nSorted Array:");
        for (int num : array) {
            System.out.print(num + " ");
        }
    }
}
```

Question 5
The program sorts an array initialied as {38, 27, 43, 3, 9, 82, 10}. Merge sort tries to merge segments so it should start with Size 1 merges, then move on to Size 2 merges, etc. Please show the **<u>Size 1 merges</u>** for the initial pass by filling out the table below. (15 points)

	Array 1 (Size 1)	Array 2 (Size 1)	Result
Pair 1			
Pair 2			
Pair 3			
Left Un-Paired			

Then show the array after Size 1 merges (5 points)

Question 6 Please show the **<u>Size 2 merges</u>** for the initial pass by filling out the table below. (15 points)

	Array 1 (Size 2)	Array 2 (Size 2)	Result
Pair 1			
Pair 2			

Then show the array after Size 2 merges (5 points)

Question 7

The code presented sorts ascending. If this code was provided to you, and asked to develop the option for descending. You solution is to first sort ascending, then reverse the array for descending sort which looks like the following.

```
public static void mergeSort(int[] array, boolean ascending) {
 // ... do exactly same things as before
if (!ascending) {
        reverseArray(array);
    }
}
```

Please provide a simple implementation of the reverseArray method here

Question 8

What would be the added complexity of the reverseArray method in big-O?