COSC 121: Computer Programming II
Practice Final Examination

*This is a practice exam. While the structure is somewhat similar to the actual exam, the length of each section **will be** different. The questions themselves will also be different (obviously) and will focus on any of the topics we learned in the course*

In order to prepare for this exam, you should be able to solve all questions from the lectures and labs on your own. No questions are more important than others. You also need to practice writing code on paper.

/ 10

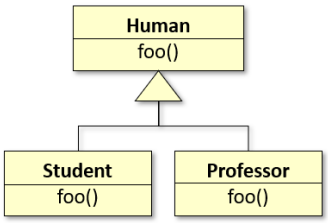
PART 1. Multiple Choice Questions (Circle correct answer) (20 marks)

1. The worst-time complexity for merge sort is?

A. O(1)
B. O(log n)
C. O(n)
D. O(n log n)
E. O(n*n)

2. In order for the code below to be valid, what must the type of the reference variable **x** be?

```
????? x;
x = new Student();
x.foo();
x = new Professor();
x.foo();
```



A. Human
B. Student
C. Professor
D. Something else

3. Suppose a list is {2, 9, 5, 4, 8, 1}. After the first pass of bubble sort, what does the list become? becomes?

A. 2, 9, 5, 4, 8, 1
B. 2, 9, 5, 4, 1, 8
C. 2, 5, 9, 4, 8, 1
D. 2, 5, 4, 8, 1, 9
E. 2, 1, 5, 4, 8, 9

4. Which of these is true about Java Interfaces?

A. All interfaces share a single root, the Object interface
B. Interfaces can have constructors, abstract/concrete methods and instance variables
C. An interface can extend any number of interfaces
D. An interface is a class

5. Which of these declarations is valid?

A. `class WorkingStudent extends Employee, Student implements Comparable`
B. `class WorkingStudent extends Comparable`
C. `class WorkingStudent extends Employee implements Student`
D. `class WorkingStudent implements Comparable, Serializable`

6. Suppose list1 is an MyArrayList and list2 is a MyLinkedList. Both contains 1 million double values. Analyze the following code:

```
A:
for (int i = 0; i < 100000; i++)
  list1.add(0, i);

B:
for (int i = 0; i < 100000; i++)
  list2.add(0, i);
```

A. Code fragment A runs faster than code fragment B.
B. Code fragment B runs faster than code fragment A.
C. Code fragment A runs as fast as code fragment B.

7. Which data structure is most appropriate to store patients in an emergency room?

A. Stack
B. Queue
C. Priority Queue
D. Linked List

8. After the following program is finished, how many bytes are written to the file t.dat?

```
import java.io.*;


public class Test {
 public static void main(String[] args) throws IOException {
  DataOutputStream output = new DataOutputStream(
   new FileOutputStream("t.dat"));
  output.writeInt(1234);
  output.writeShort(5678);
  output.close();
 }
}
```

A. 2 bytes.
B. 4 bytes.
C. 6 bytes.

D. 8 bytes.
E. 12 bytes
F. ClassCastException

9. To create a list to store integers, use
A. ArrayList<Integer> list = new ArrayList<>();
B. ArrayList<Number> list = new ArrayList<Integer>();
C. ArrayList<Object> list = new ArrayList<Integer>();
D. ArrayList<int> list = new ArrayList<int>();

10. Which of the following algorithms does not compare keys?
A. Quick sort
B. Merge sort
C. Heap sort
D. Radix sort

/ 10

Part2. Code Analysis **(10 marks)**

11. What is the output of the following code? Explain how you reached your answer.

```
public class A {
  boolean x;

  public static void main(String[] args) {
    A a = new A();
    System.out.println(a.x);
  }
}
```

12. What is the output of the following code? Explain how you reached your answer.

```
public class Revision {
    public static void main(String[] args) {
           try {
            int value = 30;
            if (value < 40) {
                 System.out.println("value is too small");
              }
            }
            catch (Exception ex) {
              System.out.println("Exception is caught");
            }
            System.out.println("Continue after the catch block");
          }
    }
```

/ 10

Part3: Programming **(10 marks)**

Note: the questions below are taken from exercises E07 and E08. The aim is to help you practice for the exam.

12. Write a program that creates an `ArrayList` and adds any three different objects to the list. Those objects could be of the type of any classes you previously created in this course, e.g., `Robot`, `Circle`, and `Apple`. Then, use a loop to display all the elements in the list by invoking their `toString()` method. Rewrite your loop using an iterator.

13. In the lecture, we discussed how to implement `MyStack` class using composition (i.e. a stack class with an attribute of the type `ArrayList`). Create another stack class `MyStack2` that extends `ArrayList`. Write a test program that reads a number of strings from the user then displays them in reverse order.

14. Write Java code to do the following:

    a) Create a `Robot` class with four attributes: `x`, `y`, `weight`, and `year` (i.e., manufacturing). Include a 4-arg constructor, all setter and getter methods, and the `toString` method.

    b) Implement the `Comparable` interface for comparing two robots based on their weights. If their weights are the same, compare based on their x-location. If their x's are the same, compare based on their y's.

    c) Test your code with a ***priority queue*** that includes five robots.

    d) Create another class `RobotComparator` that implements the `Comparator` interface. In this class, use the robot's manufacturing year as the base for comparison. Test your code again with a priority queue with five robots.

15. Write a recursive method that returns the number of the uppercase letters in a string using the following method header:

```
public static int countUppercaseLetter(String s)
```

    For example, `countUppercaseLetter("ABc")` returns 2.

Solution:

Part1

Question #	1	2	3	4	5	6	7	8	9	10
Your Choice	D	A	D	C	D	B	C	C	A	D

Part2

Q11- False

Q12 - value is too small

Continue after the catch block

** You also need to justify how you reached the above answers.*

Part3:

Q13-14 - see solutions to exercise E07 and E08

Q15

```
public class Test {
  public static void main(String args[]) {
    System.out.println(countUppercaseLetter("Welome"));
  }

  public static int countUppercaseLetter(String s) {

    return countUppercaseLetter(s, 0, s.length() - 1);
  }

  public static int countUppercaseLetter(String s, int low, int high) {
    if (low <= high)
      if (Character.isUpperCase(s.charAt(low)))
         return 1 + countUppercaseLetter(s, low + 1, high);

    return 0;
  }
}
```