Csci 102: Sample Exam

Duration: 65 minutes

Name:____

NetID:____

Student to your left:	**Student to your right:**
____________________________________	____________________________________

DO NOT OPEN THIS EXAM UNTIL INSTRUCTED

Instructions:

- **Write your full name and your NetID on the front of this exam.**
- Make sure that your exam is not missing any sheets. There should be four (4) double sided pages in the exam.
- Write your answers in the space provided below each problem. If you make a mess, clearly indicate your final answer.
- If you have any questions during the exam, raise your hand and we will try to get to you.
- At the end of the exam, there is one blank page. Use this as your scrap paper. If you need additional scrap paper, raise your hand and we will get it for you.
- This exam is closed books, closed notes, closed computers.
- **You need to stay in your seat until the exam is finished.** You should not leave the room even if you finish the exam. This distracts other students who are still working.

Good luck!

Problem 1 (9 points) .

Consider the following recursive method

```
public int recMethod ( int number ) {
  if ( number <= 0 )
    return 0;
  if ( number % 2 == 0 )
    return recMethod ( number - 1 );
  else
    return number + recMethod ( number - 1);
}

```

1. How many times is this method called (including the initial call) when we run `recMethod(10)` ?

2. How many times is this method called (including the initial call) when we run `recMethod(-10)` ?

3. What does `recMethod` do (i.e. what does it compute)?

Problem 2 (11 points) .

Write a recursive function that given a non-negative integer `n`, returns the count of the occurrences of 7 as a digit, so for example 717 yields 2.

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 3 (10 points) .

Consider the class definition on the right.

Write a method that creates an `ArrayList` object and fills it with `n` objects of type `Foo` initialized with random values of `x` and `y` (you can pick the range of values).

The value of `n` is specified as a parameter.

```
public class Foo implements Comparable<Foo>{

  double x;
  double y;

  public Foo ( double x, double y ) {
    this.x = x;
    this.y = y;
  }

  public int compareTo ( Foo other ) {
    double d1 = x*x + y*y;
    double d2 = other.x * other.x + other.y * other.y;

    return (d1 - d2);
  }

  public String toString ( ) {
    return "( " + x + ", " + y + " )"; //returns ( x, y )
  }
}
```

Problem 4 (10 points) .

If you have the following array of `Foo` objects

0	1	2	3	4	5	6	7	8	9
$x = 1.0$ $y = 1.0$	$x = -2.0$ $y = 2.0$	$x = 1.0$ $y = 2.0$	$x = 1.0$ $y = -1.0$	$x = 2.5$ $y = 0.0$	$x = -1.0$ $y = 0.0$	$x = 0.0$ $y = 3.0$	$x = -1.0$ $y = -4.0$	$x = 0.0$ $y = 0.0$	$x = 0.0$ $y = 1.5$

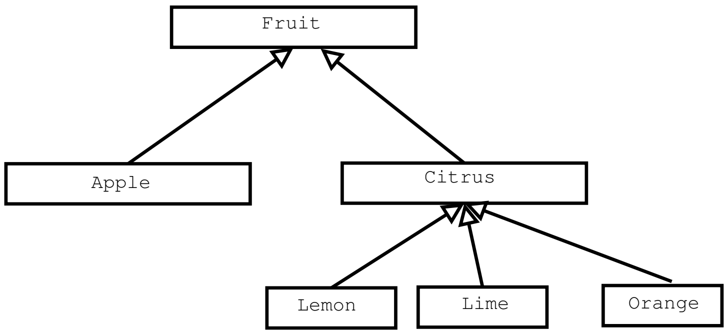
show what the array will look like after it is passed as an argument to `Arrays.sort(... )` method

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 5 (12 points) .

Suppose that `Fruit},` Apple, Citrus, `Lemon`, `Lime`, `Orange` are classes defined in the following inheritance hierarchy.
Circle all statements below that are valid.

1. `Fruit f = new Citrus();`
2. `Fruit f = new Lime();`
3. `Citrus c = new Fruit();`
4. `Citrus c = new Orange();`
5. `Apple a = new Citrus();`
6. `Citrus c = new Citrus();`



Problem 6 (8 points) .

Consider following code of four classes. What is the output from calling the `main()` method of the `Meryland` class?

```java

public class Maryland extends State {

  public Maryland() {  }

  public void printMe( ) {
    System.out.println("Read it.");
  }
  public static void main(String[ ] args) {
    Region east = new State();
    State md = new Maryland();
    Object obj = new Place();
    Place usa = new Region();
    md.printMe( );
    east.printMe( );
    ((Place) obj).printMe();
    obj = md;
    ((Maryland) obj).printMe();
    obj = usa;
    ((Place) obj).printMe();
    usa = md;
    ((Place) usa).printMe();
  }
}
```

```java
public public class State extends Region {
  public State() {  }

  public void printMe( ) {
    System.out.println("Ship it.");
  }
}
```

```java
public class Region extends Place {
  public Region() {  }

  public void printMe( ) {
    System.out.println("Box it.");
  }
}
```

```java
public class Place extends Object {
  public Place() {  }

  public void printMe( ) {
    System.out.println("Buy it.");
  }
}
```

Problem 7 (10 points) .

Write a method of a **`LinkedList`** class that computes and returns the sum of the values stored in the nodes. Assume that the node definition is as on the right. Assume there are two data fields: **`head`** and **`size`** in the **`LinkedList`** class.

```
class Node {
  int data;
  Node next;
}
```

Problem 8 (10 points) .

For each of the following methods, determine its big-O characterization in terms of `n`.

A.
```
public static int exampleA(int[ ] arr) {
  int n = arr.length, total = 0;
  for (int j=0; j < n; j += 2)
    total += arr[j];
  return total;
}
```

B.
```
public static int exampleB(int[ ] arr) {
  int n = arr.length, total = 0;
  for (int j=0; j < n; j++)
    for (int k=0; k <= j; k++)
      total += arr[k];
  return total;
}
```

C.
```
public static int exampleC(int[ ] arr) {
  int n = arr.length, total = 0;
  for (int j=0; j < n; j++)
    for (int k=0; k <= 5; k++)
      total += arr[j];
  return total;
}
```

D. a function searching for an item in a sorted array

E. a function returning an element stored at index `i` in a circular singly linked list

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 9 (10 points) .

Consider the **MyArrayList** class that extends the **ArrayList** class povided by Java API. Implement the missing **find** method. This method should return an index of the first occurance of the string **key** in the list, or **-1** if the **key** does not apprear in the list.

You cannot use the **contains** or **indexOf** methods of the **ArrayList** class in your solution.

```java
class MyArrayList extends ArrayList<String> {

 public MyArrayList () {
   super();
 }

 public int find (String key ) {
   //TO DO: implement this method
 }
}
```

Problem 10 (10 points) .

Draw the recursion trace for the execution of **reverseArray(data, 0, 4)** on array **data=[4, 3, 6, 2, 6]**.

```java
void reverseArray(int[ ] data, int low, int high) {
  if (low < high) {
    int temp = data[low];
    data[low] = data[high];
    data[high] = temp;
    reverseArray(data, low + 1, high     1);
  }
}
```

SCRAP PAPER NAME ____________________

SCRAP PAPER NAME ____________________