1 Big-O

1. Consider an algorithm which sorts an array of numbers. Fill in the Big-O running time of each algorithm in the following scenarios:

	Insertion Sort	**Selection Sort**	**MergeSort**
Array sorted in ascending order			
Array sorted in descending order (backwards)			

2. Based on the previous question, consider which sorting algorithm choose if you knew your data was mostly sorted already? Which algorithm would you choose if you knew your data was mostly unsorted? Why?

3. If you knew nothing about the dataset (ie: you don't know if the array is mostly sorted or if it's totally unsorted), would you choose to write a Selection Sort or an Insertion Sort? Why?

4. In each situation, when would you choose to use a Binary Search instead of Sequential Search?
   (a) If you only need to do one search and your data is unsorted, is it better to use Binary or Sequential Search?
   (b) If you only need to do one search and your data is sorted, is it better to use Binary or Sequential Search?
   (c) (Bonus) If you need to do a billion searches of your very large unsorted dataset, would it be better to do a billion Sequential Searches, or sort via MergeSort and then perform a billion Binary Searches?

2 Misc. Programming Questions

5. Write a program that, given an array of Strings, does two passes of printing all of the strings (IE: if there are six strings, it will print all six strings once, and then print them all again). What is the Big-O running time of your algorithm?

6. Here is an implementation of fibonacci:

```
public int fibonacci(int n){
   if(n == 0) return 1;
   if(n == 1) return 1;
   return fibonacci(n - 1) + fibonacci(n - 2);
}
```

   (a) The running time of this algorithm is $O(2^n)$. Is that *better* or *worse* than an algorithm that runs in $O(n^2)$? Is $O(2^n)$ *better* or *worse* than an algorithm that runs in $O(n)$? How about one that runs in $O(log_2(n))$ time? Rank them, from *fastest* to *slowest*:

   (b) Write a program that computes fibonacci of a given n, but without recursion:

```
public int fibonacci(int n){








}
```

3 Object Oriented Programming

7. In this question, you will develop a class called *Racecar*. A Racecar has a fuel tank, which holds up to *capacity* gallons of gas. Racecars can accelerate if there is at least one gallon of gas in the tank, but this will consume a full gallon of gas, speeding the vehicle up by 10MPH. Your class should track the current speed, number of gallons of gas remaining in the tank, and the full capacity of the tank. With this information in mind, fill in the following class:

```java
class Racecar{

   // In the beginning, the car has a full tank of gas and is going 0 MPH.
   public Racecar(int capacity){}

   // Gets the current speed of the car
   public int getCurrentSpeed(){}

   // Attempts to accelerate the car.
   // If there isn't enough fuel, this function should return false
   // If there is enough fuel, you should decrease the amount of fuel by 1 and increase the
       speed by 10.
   public boolean accelerate(){}

   // This function should return true if the car is out of fuel, and false otherwise.
   public boolean isOutOfFuel(){}

   // This function is meant to refuel the car.
   // Given the costPerGallon, compute and return how much the total bill to refuel the tank
       is, and then refill the tank to capacity.
   public double refuel(double costPerGallon){}

}
```

Consider the following example. Assume each line happens sequentially (ie: the state of the Racecar isn't reset between lines). Write out what will happen with each method call.

(a) Racecar r = new Racecar(2);

(b) r.isOutOfFuel();

(c) r.accelerate();

(d) r.refuel(3.10);

(e) r.getCurrentSpeed();

(f) r.accelerate();

(g) r.getCurrentSpeed();

(h) r.accelerate();

(i) r.getCurrentSpeed();

(j) r.accelerate();

(k) r.getCurrentSpeed();

(l) r.isOutOfFuel();

(m) r.refuel(3.20);

4 String Manipulation

8. Write a program that determines if a String is a palindrome (a palindrome is a word that is the same forwards and backwards, e.g: racecar).
   (a) Using loops.
   (b) Using recursion.

9. Write a method that, given, a String *needle* and a String *haystack*, determines whether or not *needle* occurs in *haystack*. As an example, if *haystack* = *"Hello World"* and *needle* = *"lo W"*, it should return true. If *haystack* = *"Hello World!"* and *needle* = *"World?"*, it should return false.

```
public static boolean substringExists(String needle, String haystack){




}
```

10. Write a method, *dollarsToWords*, which accepts an integer argument *dollars* (such that $0 \leq dollars \leq 999$, and prints the words that represent this number of dollars. Here are some examples:

dollars	**Output**
999	Nine hundred ninety nine dollars
84	Eighty four dollars
115	One hundred fifteen dollars
1	One dollar
100	One hundred dollars

Note: This question is more difficult than it may appear on the surface. You may benefit from helper methods. No solution is provided for this problem, as it is quite lengthy.

11. Write a program that reverses a String.
    (a) Using loops.
    (b) Using recursion.

12. Write a program that determines if a String contains only digits.
    (a) Using loops.
    (b) Using recursion.

13. Write a program that calculates how many vowels are in a String.
    (a) Using loops.
    (b) Using recursion.

14. Write a method, *bizarre*, which takes a String *str* as a parameter, and modifies the input string and returns it. The input string will contain words separated by spaces. If the word is of an even length, your resulting string should contain the word, but backwards. If the length is odd, you should add the word to your result as it appears. Here are some examples:

str	**result**
"the old home"	"the old emoh"
"pig fish dog kitten"	"pig hsif dog nettik"
"hello world"	"hello world"
""	""

15. Without using Integer.parseInt(String), write a program that converts a String into an integer. You may assume that the number in the string will actually fit inside of an integer.
Hint: You should use loops and start at the most significant digit in the String.

5 Arrays and Matrices

16. Consider a 2D array of strings:

```
String[][] strMatrix = new String[8][8];
```

You are to write a program that fills in each cell in the matrix with a String in the format "*row*, *column*". That is, it should look roughly like:

"0, 0"	"0, 1"	"0, 2"	...
"1, 0"	"1, 1"	"1, 2"	...
"2, 0"	"2, 1"	"2, 2"	...
...	...	...	...

17. For each of the following, assume we have declared the following:

```
int[] arr = {1, 2, 3, 18};
int[] arr2 = new int[8];
int[][] arr3 = { {1, 2}, {3, 4}, {5, 6}, {7, 8}, {9, 10} };
String s = "Hello World";
```

Identify what the result will be, or write *Compile-time* or *Run-time* if there is a Compile-time or Run-time error:

(a) arr[0];
(b) arr[arr2.length - 5];
(c) s.charAt(arr.length);
(d) s.charAt(arr[3]);
(e) s.charAt(arr[5]);
(f) s.charAt(arr[1]);
(g) arr3[arr.length];
(h) arr3[arr.length][1];
(i) arr3.length;
(j) arr3[0].length;
(k) arr3[1].length();
(l) s.length;
(m) s.charAt(s.indexOf('H'));
(n) s.charAt(s.indexOf('Q'));
(o) s.indexOf('Q');
(p) arr2[0];
(q) arr2 = arr;
(r) arr[0] = arr3[2][0] + arr3[3][0];
(s) arr[0] = (int) s.charAt(6);
(t) arr[0] = s.charAt(6);
(u) int j = 6.0;
(v) double d = 7;
(w) s += 'q';
(x) (s + "Goodbye!").length();
(y) s == ("Hello " + "World");
(z) s.equals("Hello " + "World");

6 Searching and Sorting

18. List the resulting array after each call of the merge method. Indicate the number of comparisons made for each call to merge (line 20 of the merge method at the end of the assignment). Sort the following array of characters (sort into alphabetical order):

$$\{7, 12, 100, 18, 23, 8, 1, 3, 4, 0, 19, 6, 4, 1, 20, 12\}$$

19. List the resulting array after each iteration of the outer loop of the insertion sort algorithm. Indicate the number of comparisons made for each iteration (line 06 of the Insertion Sort algorithm at the end of the assignment). Sort the following array of characters (sort into alphabetical order):

$$\{8, 1, 3, 4, 0, 19\}$$

20. Trace a Binary Search of the following dataset, where we are attempting to find the number 8. Include (a) the left index and (b) the right index of the array that denote the region of the array that is still being searched, (c) the middle point of the array, and (d) the number of comparisons made during the search (line 09 and line 11 of the Binary Search algorithm at the end of the assignment).

$$\{1, 3, 5, 6, 12, 19, 20, 25, 28, 33\}$$

21. Here is an implementation of binary search. It has a number of bugs. Try to fix them to make the code work correctly.

```java
public static int binarySearch(int[] arr, int target){
   double left, right, middle;
   left = 1
   right = arr.length - 1;
   while(left > right){
      middle = (left + right) / 3; // Find the middle index.
      int middleElement = arr[middle];
      if(middleElement == target){
         return middle;
      }else if(middleElement > target){
         right = middle - 1;
      }else if(middleElement < target){
         left = middle + 1;
      }
   }

   return -1;
}
```

To help test your solution, ensure it works on the array: $\{1, 2, 4, 12, 18, 23, 54, 56, 60, 89, 91, 102, 148, 176, 192\}$ for the targets:

Target	**Expected Result**
1	0
192	14
56	7
176	13
200	-1
0	-1
24	-1