Practice Exam

Part I

1) What is the best case time complexity of insertion sort? What property must the input have for the best case?

2) What is the worst case time complexity of quick sort? What is the property of the input that will result in the worst case?

3) What is the time complexity of the “removal” operation for a priority queue(where the underlying data structure is a heap)?

4) Given a sorted array of ints, what is the worst-case time complexity of searching for the number 5?

5) Assume we are searching through an unsorted array of ints. What is the worst case time complexity to find the number 18?

6) What is the worst case time complexity of the delete operation in a balanced BST?

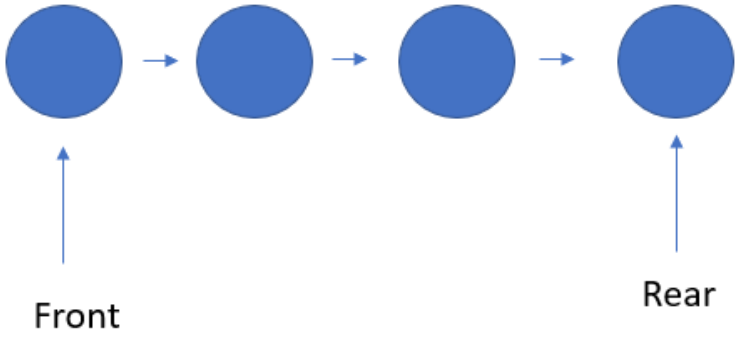
Part II

1) What is the runtime complexity of the operation push and operation pop for a stack(where the underlying data structure is an array)?

2) What is the performance of dequeue operation on a queue(where the underlying data structure is a singly linked list)

3) What is meant by LIFO?

4) John is implementing a queue with the underlying datastructure being a singly linked linkedlist. At which end of the linkedlist should the enqueue operation add a node, and at which end of the linkedlist should the dequeue operation remove a node, and why?



A) Enqueue at front and dequeue at rear
B) Enqueue at rear and dequeue at front
C) It does not matter

Part III

1) Is it necessary for an abstract class to have an abstract method?

2) Is this a valid example of method overloading?

```
class A{

        public void doProcedure(int x, String y){
                System.out.println(y.length() > x);
        }

        public boolean doProcedure(int x, String y){
                return y.length() > x;
        }
```

}

3) What is an interface? Can an interface be instantiated?

4) Explain what it means for a sorting algorithm to be “stable”.

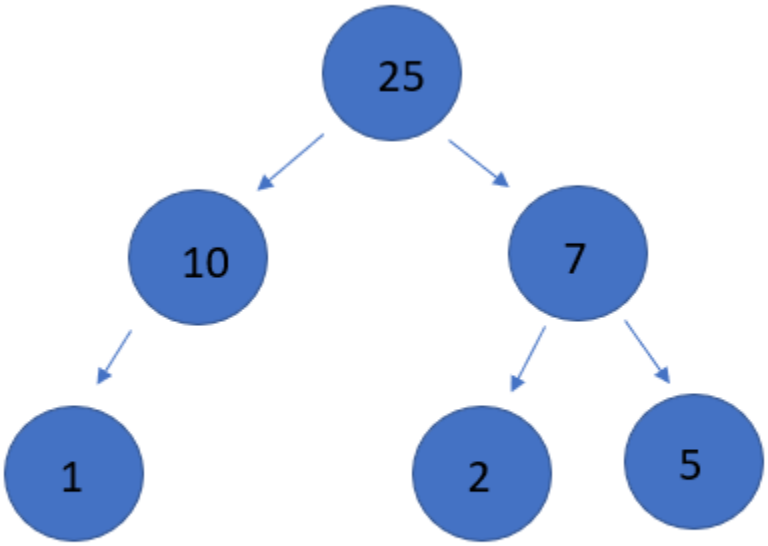
Part IIII

1) Add the following to a BST: 5, 7, 2, 6, 10 , 25, 1, 4, 9

2) Is the resulting binary search tree balanced? First define balanced then explain why or why not.

3) Delete 7 from the BST. Draw the resulting BST.

4) Is this a max heap? List the properties of a max heap, and use these properties to justify your answer.



5) The underlying data structure of a heap is an array. We are looking for the parent of the element at index 13. At what index will I find the parent? **For this question, assume that the first index of the array is left empty. Meaning, assume the root will be at index 1.**

6) How many nodes does a perfect tree, with a height of 12 have? You may express your answer in the form of a exponential equation.

7) Define what a binary tree is.

Coding-By-Hand Questions

1) Implement a function called “find evens”. Given an input array, the function should copy all the evens into a new array and return this new array.

```
public int[] findEvens(int[] a){




}
```

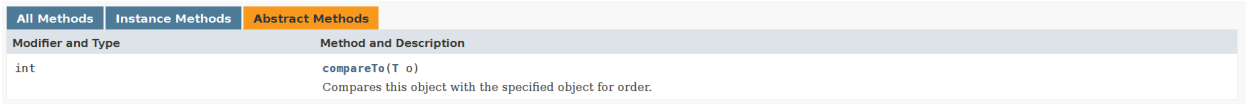
2) Implement a function called “reverseString”. Given an input string, the function should return the string reversed.

```
public String reverseString(String s){




}
```

3) Make a class the follows the following requirements:

   a. Create an Employee class such that it implements the comparable interface.

   b. For your reference:



All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
int	compareTo(T o) Compares this object with the specified object for order.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

The compareTo method should be implemented as follows: this Employee is greater if its tenure is greater, this Employee is equal if its tenure is equal, and this Employee is lesser if its tenure is lesser.

c. The Employee class should have instance variables name(a String), tenure(an int), and salary(an int)

d. Create a constructor for the Employee class which takes 3 arguments: String name, int tenure, int salary and initializes the instance variables appropriately