CSCI 2120

Exam 2

Spring 2019

Part I. MINI-ESSAYS. Answer the questions as completely as you can without sacrificing conciseness. The space provided should be a good indication of the length of answer required.

1. In computer science, we need a way to compare various algorithms. One metric that we have at our disposal is "Big O". a) What does Big O measure? b) What is N? c) Make a chart with the Big O signatures in increasing complexity from constant time to quadratic time and give an example of an algorithm with this signature (if we discussed one) (7 pts total).

-1½

2. We spent an entire chapter discussing various Generic Data Structures: ArrayList, Linked Lists, Stacks, Queues, Maps, etc. Why write a Data Structures using generics, as opposed to a) writing a type-specific data structure (i.e. StudentList) or b) simply using the abstract type Object everywhere? (5 pts.)

3. a) Make an annotated (labeled) drawing of an object that represents a singly linked list with 5 elements in it. b) What instance variables does the LinkedList object ***itself*** keep track of? Indicate them on the diagram. (5 pts.)

4. a) When we apply the "remove" operation on an element in a singly linked list, how is this actually accomplished? b) Draw a diagram of the state of a linked list before and after the removal of some element. c) Discuss how the idea of "garbage collection" relates to this. (5 pts.)

5. a) What is the difference between the Comparable<E> and Comparator<E> interfaces? Why would you ever use a Comparator when Comparable classes exist? b) Write the signatures for the respective methods that these interfaces declare. (5 pts).

Part II. *Multiple Choice.* Indicate the best term from among the given options that best completes/defines/describes the item. *(2 pts. each)*

1. Big O notation allows us to compare the _____ between algorithms.
   a. possible size of the input
   b. difficulty coding the implementation
   c. runtime as a function of input size
   d. difficulty of debugging

2. Deleting the first element is faster in a Linked List than in an array-based list.
   a. true
   b. false

3. Which of the following algorithm efficiencies provides the best performance?
   a. O(n)
   b. $O(n^2)$
   c. O(lg(n)))
   d. O(n*lg(n))

4. Using an integer index to access a random element is faster in a Linked List than in an array-based list.
   a. true
   b. false

5. A Linked list where Nodes know about both their next and previous Nodes is called a:
   a. Doubly-Linked List
   b. Singly-Linked List
   c. Recursively Linked List
   d. Binary Tree

6. A singly linked list has the following property:
   a. each Node has a reference to the next Node
   b. the List knows about the first Node
   c. it can only be traversed in one direction
   d. a and c
   e. all of the above

7. Which of the following classes does NOT implement the java.util.List<E> interface?
   a. Vector<E>
   b. Map<K,V>
   c. LinkedList<E>
   d. ArrayList<E>

8. The correct syntax to use in a class definition if we want to limit a generic type to only subtypes of Number would be:
   a. <T <= Number>
   b. <T extends Number>
   c. <T instanceof Number>
   d. <T implements Number>

9. When the compiler replaces a generic with the most abstract type possible during compilation this is known as _____.
   a. revision
   b. substitution
   c. erasure
   d. boxing
   e. none of the above

10. Generally speaking, Big-O notation indicates the _____ run-time for an algorithm.
    best-case
    average-case
    worst-case

11. In Big-O notation, constant time efficiency is denoted as _____.
    a. O(n)
    b. $O(n^2)$
    c. O(1)
    d. O(n*lg(n))
    e. none of the above

12. Which of the following algorithms is the more efficient searching algorithm?
    Linear Search
    Binary Search
    They are equally as efficient

13. Binary search depends on the input data being _____.
    a. numeric
    b. sorted
    c. random
    d. none of the above

14. Which of the following is the most efficient sorting algorithm (in terms of worst-case Big-O)?
    a. Selection Sort
    b. Insertion Sort
    c. Merge Sort
    d. They are all equally as efficient

15. Selection Sort has an efficiency of _____.
    a. O(1)
    b. O(n)
    c. O(n*lg(n))
    d. $O(n^2)$

16. Merge Sort has an efficiency of _____.
    a. O(1)
    b. O(n)
    c. O(n*lg(n))
    d. $O(n^2)$

17. In a binary search tree, the left subtree of a node has only values that are _____ the value of the node.
    a. less than
    b. greater than
    c. equal to
    d. subclasses of

18. An $O(n^2)$ algorithm is said to operate in _____ time.
    Logarithmic
    Constant
    Linear
    Quadratic
    Exponential

$O(n^2)$ algorithms run more quickly than O(n*lg(n)) algorithms (for large n).
    true
    false

20. Which of the following snippets makes use of the generic wildcard feature?
    a. MyClass <T super List>
    b. Map<K,V>
    c. ArrayList<? extends Number>
    d. LinkedList<String> list = LinkedList<>()

21. Removing the last element of an array list is faster than doing so to a singly-linked linked list.
    true
    false

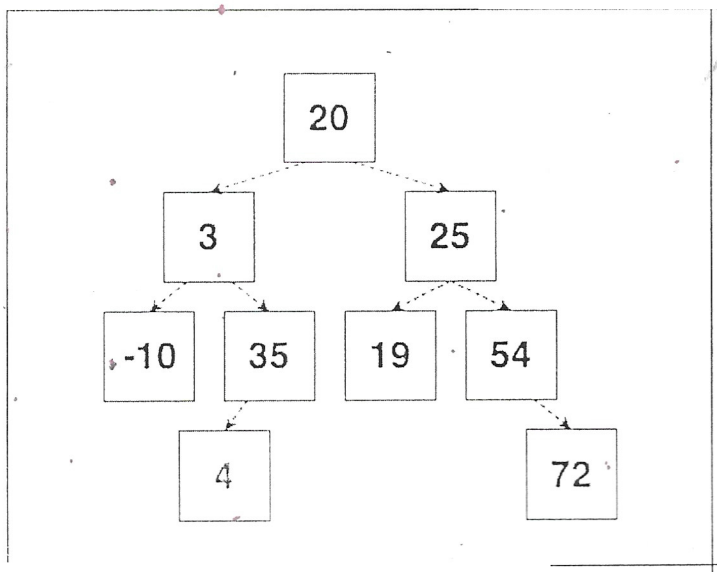
22. To extract the elements of a BST in sorted order, you might use a _____ tree traversal.
    a. Pre-Order
    b. In-Order
    c. Post-Order
    d. Level-Order

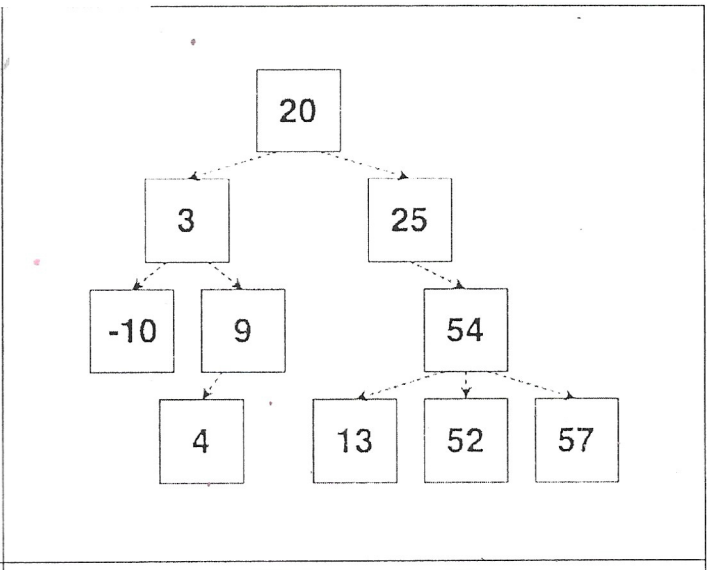
23. Classes that implement the _____ interface give a natural ordering to its instances.
    a. Comparable<T>
    b. Comparator<T>
    c. Orderable<T>
    d. AbstractComparisonFactory<T>

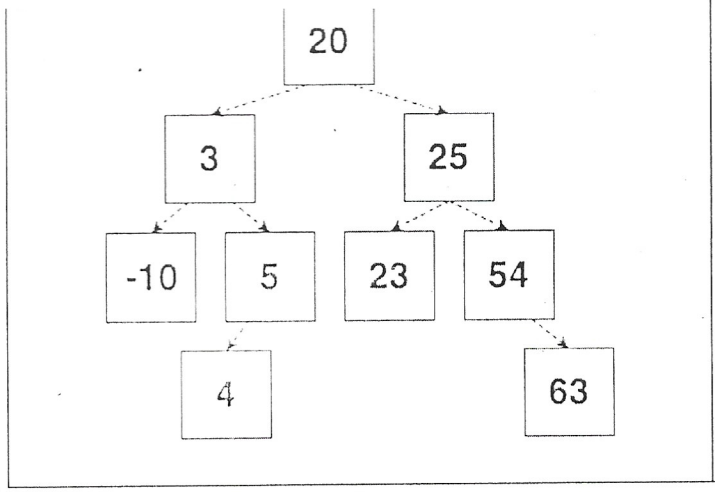
-2

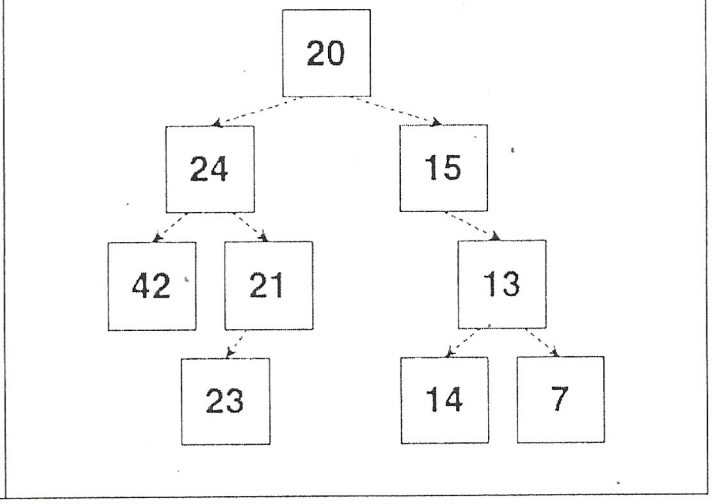
Part III. BINARY SEARCH TREES. Based on your knowledge of Binary Search Trees, answer the following questions below (**3 pts**):

1) Which of the following trees demonstrates the Binary Search Tree property (e.g. which ONE is a valid BST)? Assume the elements are ordered according to their natural ascending integer order. Choosing more than one will result in zero credit.









Part IV. BIG O ANALYSIS. Fill in the following chart with the worst-case Big-O run time of the following algorithms for both ArrayList and LinkedList. (1/2 **pt. each. Total 8 pts for Part IV)**

Remember that in the worst case, ArrayList will need to resize itself to accommodate new elements. Consider this to be part of the worst case. Assume the LinkedList implementation is singly-linked with references to both the first and last elements only.

Operation	List Type	
	ArrayList	LinkedList
Get Element (using an integer index)		
Insert At Beginning		
Insert At End		
Insert (using integer Index)		
Insert (using iterator)		
Remove from Beginning		
Remove from End		
Get Element (using an iterator)		

-1½

Part V. CODING SECTION. Based on your knowledge of Java Programming, supply the required answers for the exercises below:

(8 pts). Given the following basic implementation:

```
public class CSCIStudent {

      private int    studentID;
      private String name;
      private double hoursPerWeekProgramming;

      public CSCIStudent(String name, int id) {…}

      public int    getID()   { return this.studentID; }
      public String getName() { return this.name; }
      public double getHPWP() { return this.hoursPerWeekProgramming; }
}
```

Write a method called sortStudents that takes as its argument an array of CSCIStudents and uses a simple sort mechanism (you may implement any sorting algorithm we've studied) to sort the items in place. The order we want to sort on is ***the number of hours the CSCIStudent spends programming, in ascending order. YOU MUST STATE WHAT ALGORITHM YOU ARE IMPLEMENTING IN THE COMMENT!!!***

Provide your implementation on the next page. Implementing MergeSort will yield **+10** bonus points (if done correctly).

```
public void sortStudents (CSCIStudent[] students, int numStudents) {

      // Name of sorting algorithm implemented:
```

(8 Points) Generics: Using an ArrayList<T> as an internal data structure, write the class Stack<T> such that the methods

```
public void push(T element) {}
```

and

```
public T pop() {}
```

are properly implemented. On the next page, are some of the instance methods of ArrayList that you might use..

```
/**
 * Implementation of a generic stack data structure using
 * an ArrayList as its underlying data structure.
 *
 */
public class Stack<T> {
```

```
} // end class Stack<T>
```

A subset of public instance methods in ArrayList:

```
public void  add(T element)
       /* Appends the specified element to the end of this list.*/

void   add(int index, T element)
       /* Inserts the specified element at the specified position in this list.*/

public T get(int index)
       /* Returns the element at the specified position in this list.*/

public int  indexOf(Object o)
       /* Returns the index of the first occurrence of the specified
       element in this list, or -1 if this list does not contain the element.*/

public boolean    isEmpty()
       /*Returns true if this list contains no elements.*/

public T remove(int index)
       /*Removes the element at the specified position in this list.*/

public int size()
       /* Returns the number of elements in this list. */
```

BONUS 1:

(10 pts.) Recall the searching algorithm Binary Search on an array or list. Binary Search operates on sorted input. It keeps two indices to the list, a beginning index and an ending index. Given a value to search for, it calculates the index of the middle element of the list. It then compares the value with that element.

Provide an implementation of **Binary Search.** Write a method that accepts as parameters an array and an integer to search for, and returns the index where that element is located. If the element is not in the array, it returns -1. **(You can assume that the input array is sorted.)** Use the natural ascending int order.

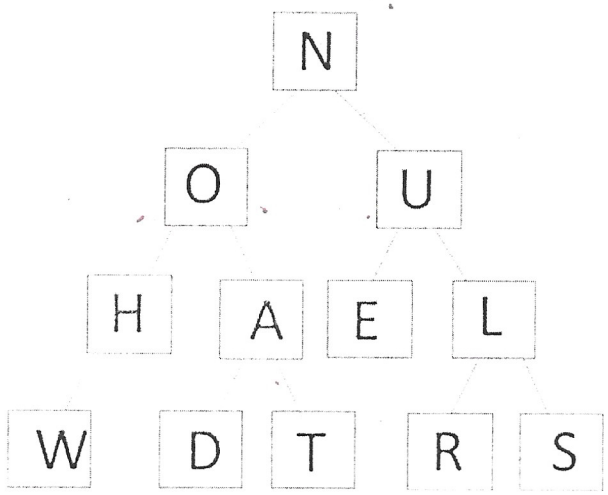
Part of the implementation has been provided for you:

```
public int binarySearch (int[] list, int key){

    int low = 0;                    //Smallest possible index.
    int high = list.length - 1;     //Largest possible index.
    while (low <= high){
        int mid = (low + high + 1) / 2;   //Current search index.
```

BONUS 2:

(5 pts. each) Given the following binary tree (over some *unknown, arbitrary data type indicated by the letters*), write a) An InOrder traversal, b) a PreOrder traversal, and c) a PostOrder traversal.



InOrder:

PreOrder:

PostOrder:

+15