COMP 241Review

Topics:

* Java Review
  + Understanding and writing Java code, understanding references and memory, class design and scope.
* Big O notation
  + Algorithms: What is the big o notation?
  + Looking at a piece of code-what is the big o running time?
  + Compare two pieces of code or two big O running times-which is slower?
* Abstract data types
  + Using them, designing for them, and programming them.
  + Big O analysis for operations with different ADTs.
  + *Implementation* vs *interface*
* Linked Lists
  + Designing linked lists, using linked lists, variables/implementation (node, head, tail, etc.)
  + Singly-linked lists
  + Doubly-linked lists
  + Circular linked lists
* Stacks
  + Using stacks, designing stacks
* Queues
  + Using queues, designing queues

1. Using a Stack data type, write a Java function to reverse a String.
2. Given an int queue (named queueOfInts) with the elements {5, 12, 2, 44, 51, 1}, what would the following code print out? (Here, the 5 element is the front of the queue.)

public static void main(String[] args)  
{

queueOfInts.dequeue();  
queueOfInts.dequeue();  
queueOfInts.enqueue(22);  
System.out.println(queueOfInts.dequeue());  
System.out.println(queueOfInts.dequeue());  
System.out.println(queueOfInts.dequeue());

}

1. What is the big O running time for the following Java code snippets? (Make sure to put this in terms of n.)
2. public static void main(String[] args)

{

recursiveMultiply(90, 80);

}

public int recursiveMultiply(int a, int b)

{

If(b <= 0)

{

return 1;

}

return a \* recursiveMultiply(a, b-1);

}

1. for(int i = 0; I < n^2; i++)

{

System.out.println(“Looping!”);

}

1. public static void main(String[] args)

{

Weird(0);

}

public String weird(int n)

{

If(n > 10)

{

return “t”;

}

return weird(n+1) + n + weird(n+2);

}

1. Suppose there exists a RLinkedList and Node class for a singly-linked list:

public class Node<E>  
{

public E data;

public Node next = null;  
}

public class RLinkedList<E>  
{  
 public Node<E> head;  
}

Using this RLinkedList class, create a circular linked list by filling in the following methods: add(E data, int position), delete(int position), and getSize(). You may add additional variables or functions to this class as needed.

public class RCircularList<E>  
{

RLinkedList<E> circleData;

public void add(E data, int position)  
 {//Your code here

}

public void add()  
 {//Your code here

}

public void delete(int position)  
 {//Your code here

}

public void getSize()  
 {//Your code here

}

}

What is the big O running time of each of the methods you’ve created?

1. There exists a data structure named RDataBank that stores the data a user enters in a String array named DataBankArray. This array initially starts with 13 elements, but once a user enters more than 13 values into the array it has to be resized. Write a function to resize the DataBankArray to twice its current size once the capacity has been exceeded.

public class RDataBank

{

String[] DataBankArray = new String[13];

//Your function here.

}

What is the big O running time of this function?

1. The company you work for needs an implementation of a Stack that uses a LinkedList to store data. Using the RLinkedList class from Problem 4, design a Stack class with a push, pop, and peek function. This class must work for multiple data types and so should use the *generic* functionality (implemented with <E> in Java). What is the big O running time of each of these functions?