COMP 210 – Fall 2018

Midterm Sample

This is a sample of the kinds of questions you can expect for the midterm.

1. Determine the order of each of the following growth functions.
   a) $10n^5 + 100n^3 + 1000$
      Answer: $O(n^5)$
   b) $2^n + 100n^3$
      Answer: $O(2^n)$
   c) $n^2 \log_2 n$
      Answer: $O(n^2 \log_2 n)$

2. Which of the previous functions will eventually grow fastest?
Answer: b) Exponential functions grow very fast for large n

3. Determine the growth function and the order of the following code fragment:

```
int n = 10;
int sum = 0;
for (int i = 1; i < n; ++i)
{
   int j = i;

   while (j < n)
   {
      sum += j;
      j++;
   }
}
```

Answer: This is a situation with nested loops. The outer *for* loop is performed once for every item and therefore has a complexity of $O(n)$. The inner *while* loop is performed j times for every item in n (first n-1 times, then n-2, n-3, … 1) and therefore multiplies the complexity by a factor of n giving a complexity on the order of $O(n^2)$.

4. Define: primitive data type
Answer: A primitive data type is built into the language an represent the building blocks of data manipulation in Java. Such types serve only one purpose — containing pure, simple values of a kind. An example is **int**.

5. Define: abstract data type
Answer: An ADT is a concept, a mathematical definition of data organization with a set of operations that is not concerned with the implementation details.

6. Briefly (two or three sentences) describe the relationship between a class and an object.
Answer: A class defines the properties and methods for an ADT, hence worrying about implementation details (what data type to use, what's the big-oh of using one technique against another, and so on). An object is an instance of a class, much like a variable is an instance of a primitive data type.

7. We discussed three types of lists (not JAVA implementations of lists). Name them.
Answer:

a) Ordered list
b) Unordered list
c) Indexed list

8. Pick one of the three ways we learned to implement ordered lists in JAVA (Array, LinkedList, or ArrayList). Give at least one advantage and at least one disadvantage of the implementation you picked.
Answer Array: arrays are easy to define and manipulate. Arrays have properties that allow you to easily find the first, last, or any given element of the array. Unfortunately, arrays are fixed size and can neither grow or shrink as needed without having to perform extra steps.

Answer ArrayList: ArrayLists are a more flexible implementation of an array. ArrayLists have all of the advantages of arrays with the added ability to grow when needed. The main disadvantage is in the method used to grow the ArrayList. A new ArrayList is created that is twice the size and then the ArrayList is copied into the new one. This is not a quick procedure.

Answer LinkedList: A LinkedList starts at one element and grows very effectively. The LinkedList takes up only as much memory as actually required. However, it is not as easy to find the first element of the linked list (you must keep track of where the first element is in your program) and to find any element the program must start at the first element and then follow the links to desired element.

9. How is a Queue different from a Stack? What properties are similar?
Answer: a) Queues are FIFO, stacks are LIFO b) *Inserting:* A queue inserts elements at the end, a stack inserts at the top. c) *Removing:* A queue removes

elements from the front, a stack removes from the top. d) *Peeking:* A queue returns the first element (element at the front), a stack returns element at the top.
Similarities: You can't iterate over the elements of a stack and queue (it is not in its nature). Both can be implemented using Arrays or LinkedLists. Both are linear collections of elements.

10. Explain why a Binary Search can be implemented in $O(\log_2 n)$:
Answer: Because you don't need to compare all the elements with the one being searched, you just divide the list in two halves (one half containing elements that are smaller than the other half). You continue the process of halving the halves until you get an answer. With this process you end up doing at most $\log_2 n$ comparisons.