CSCI 132: Basic Data Structures and Algorithms

Midterm Study Guide

Logistics

- Wednesday, March 12th @ 3:10 PM in Norm Asbjornson Hall 165
- Time length: 50 minutes. We will try to start the exam at 3:05 .This exam is designed to be completed in 40 minutes (hopefully).
- Midterm Exam will be a D2L Quiz (let me know if you don't have access to a laptop)
- Open notes. You are allowed to use your laptop, your IDE, any notes, slides, lecture examples, lecture recordings, Java documentation.
- You are NOT allowed to use the internet to access external resources (Google, Stack Overflow, W3 Schools, etc) and you are NOT allowed to use AI tools
- The midterm exam will consist of different types of questions, such as:
  - Multiple choice questions
  - True/False
  - Matching

Content

The following topics are all fair game for the midterm exam.

- Basic Java Classes, OOP
- Operations and variables
- Methods
- If statements
- Loops
- Arrays
- Inheritance
- References
- Static Methods
- Exceptions
- Abstract Classes, Interfaces
- ArrayLists
- LinkedLists (Singly, Doubly, Circular)
- Stacks
- Growth Rates
- Big-O Notation, How to determine running time of an algorithm

Sample Exam Questions

1. Consider the following Java Class

```
public class Duck {

   private String name;

   public Duck() {

       this.name = n;

   }

 }
```

What does the **private** keyword mean?

a. The user cannot create a class
b. Other classes cannot directly access the value
c. Only the parent class can directly access the value
d. name cannot be changed

Consider the following line of code in a demo class:

```
Duck don = new Duck("Donald");
```

This line of code results in an error. Why is this error occurring?

a. “Donald” is not a valid String
b. The constructor is not defined
c. The constructor is being passed too many arguments
d. The constructor is being passed too little arguments

Rewrite the constructor below so that the line of code from part B work correctly

a. public Duck(String name)
b. public Donald(String n)
c. public Duck(String don)
d. public Duck(String n)

2. What is an **interface** in Java?
    a. A class with method signatures and no bodies
    b. A class that cannot be created
    c. A class that is implemented by another class
    d. All of the above

3. What will the following code print out?

```
int myArray[] = {1, 2, 3, 4, 5};

for(int i = 0; i < myArray.length - 1; i++) {

      System.out.print(myArray[i+1] * 2 );

}
```

    a. 2 4 6 8
    b. 4 6 8 10
    c. 1 2 3 4 5
    d. 2 4 6 8 10

4. True/False. To add a node to the very end of a doubly linked list (that has a head and tail pointer), we must traverse the entire linked list first.

5. The code below prints out an N x N multiplication table.

```
public void print_table(int n) {

   for (int i =1; i <= n; i++){

        for (int b=1; b <= n; b++){

              System.out.print(i*b + " ");

           }

        System.out.println();

        }

}
```

For each instruction in the function, clearly mark/label the running time of that operation.

What is the total running time (in Big-O) of this function?

a. a. O(1)
b. O(N)
c. O(N^2)
d. O(X^N)

6. What is the running time of printing out a Circular Linked List?

a. O(1)
b. O(N)
c. O(N^2)
d. O(X^N)

7. Consider the following code that uses a Stack data structure

```
stack.push(“Red”);

stack.push(“Blue”);

stack.pop();

stack.push(“Yellow”);

stack.push(“Purple”);

stack.peek();

stack.pop();
```

When this code is complete, what value is at the top of the stack?

a. Red

b. Blue

c. Purple

d. Yellow

8. Consider this basic Node class that is used in a circular linked list.

```
public class Node {
    private String name;
    private Node next;
    private Node prev;
    public Node(String c) {
        this.name = c;
        this.next = null;
        this.prev = null;
    }
    public Node getNext() {
        return this.next;
    }
    public Node getPrev() {
        return this.prev;
    }
    public void setNext(Node newNode) {
        this.next = newNode;
    }
    public void setPrev(Node newNode) {
        this.prev = newNode;
    }
```

A new node needs to be inserted into the circular linked list before the head. What is the correct code so that a new node (`newNode`) will get added. You can assume that the circular linked list has at least 1 node already in it. The linked list class does keep track of the `head` node, and `tail` node.

a.
```
public void addBeforeHead(Node newNode){

            newNode.setNext(head);

            tail.setNext(head);

            head.setPrev(tail);

}
```

b.
```
public void addBeforeHead(Node newNode){

            tail.setNext(head);

            tail.setNext(newNode);

            head.setNext(newNode);

}
```

c.
```
public void addBeforeHead(Node newNode){

      tail.setNext(newNode);

      newNode.setNext(head);

      head.setPrev(newNode);

      newNode.setPrev(tail);

}
```

d.
```
public void addBeforeHead(Node newNode){

      head.setNext(tail);

      newNode.setNext(head);

      head.setPrev(newNode);

      newNode.setPrev(tail);

}
```

What is the running time of your algorithm from the previous question? (You must state your answer in Big-O notation)

a. O(1)
b. O(n)
c. O(n^2)
d. O(2^n)