

CS 112 – Data Structures

Summer, 2016

Midterm Practice Exam I – **Answers**

NOTE: See the topic list on Sakai > Resources > Exam Info > Exam 1 for a complete list of topics that may be on the actual exam.

- **Do not start** until everyone has an exam and the instructor tells you to begin.
- There are 6 pages in this exam, including this one. Make sure you have them all.
- This exam is closed book – closed notes.
- You must put your cellphone, iPod, or other electronic devices in a backpack, etc., and leave it out of your reach. The only exception is that you can use a “dumb” watch (e.g. not a iWatch or other smart watch, calculator watch, etc).
- Write clearly – if we can’t read or can’t find your answer your, answer is wrong.
- Make clear what is your answer versus intermediate work.
- When you turn in your exam, you must show your Rutgers ID card with a legible picture, or else your driver's license
- If this were a real exam you would have 100 minutes to do it. The time for the actual exam may be different.

1.

a. For each of the following functions give its simplest big-O equivalent.

E.g. for $F(n) = 5$ the answer would be $O(1)$

i. $F(n) = n^2 + 100$ is $O(n^2)$

ii. $F(n) = ((n+1)*(n-1))^{0.5}$ is $O(n)$

iii. $F(n) = 2^{n*n}$ is $O(2^{n*n})$

iv. $F(n) = n + (n * \log_2(n))$ is $O(n * \log(n))$

2. Suppose you have a sorted linked list of n positive integers and another integer (the target) and you want to find the smallest integer in the list at least as large as the target. For instance, if the list is $2 \rightarrow 33 \rightarrow 42 \rightarrow \text{null}$ (n is 3) and the target is 20, since the list is sorted, as soon as you see the 33 you would stop and return the 33, since $33 \geq 20$. How many elements of the list would you have to compare with the target? Give a specific answer in terms of n , not in terms of big-O. Also explain how you get this answer. There will be no credit unless the explanation is correct.

a. worst case? n Explain your answer. because in the worst case you have to compare the target with every number in the linked list

b. best case? 1 Explain your answer. because in the best case the first element of the list is \geq target

3. Consider the following problem: Given a sorted array S holding s numbers and an unsorted linked list U holding u numbers, find all the numbers in U that are not in S . For instance if S holds 3 6 44 45 and U holds 2 6 3 1 the answer is 2 1. Assume that there are no duplicates within U or within S .

- a. Suppose Algorithm A is to go through U element by element and check each one (with the fastest search that we have seen that can work on S) to see if it is in S. What is the big-O worst case complexity of Algorithm A?

$u \log s$

- b. Suppose Algorithm B is to go through S element by element and check each one (with the fastest search that we have seen that can work on U) to see if it is in U. What is the big-O worst case complexity of Algorithm B?

$u * s$

4.

- a. Suppose you start with an empty stack and do the following operations. For each pop, show the data that would be returned by the pop. If any Push 7, push 5 push 4, pop: result 4, pop:result 5, pop: result 7, pop: result error

- b. Do the same as a above but for a queue.

Enqueue 4, enqueue 5, dequeue: result: _4_, enqueue 3, dequeue: result _5_

5. Suppose you implemented a queue as a Circular Linked List. Complete the enqueue method in the following code. Assume the CLLNode class defined below.

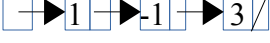
```
public class Queue{
    CLLNode tail;
    public enqueue(int data){
        if(tail == null){
            tail = new CLLNode(data, null);
            tail.next = tail;
        } else {
            tail.next = new CLLNode(data, tail.next);
            tail = tail.next;
        }
    }
}

class CLLNode{
    int data;
    CLLNode next;

    public CLLNode( int num, CLLNode nxt){
        data = num; next = nxt;    }}
```

6. Suppose you had a generic doubly-linked list as below. Complete the `DLLNode<T>` definition (by filling in the blanks) and the `insertAfter(here, data)` method which inserts a node with the given data after the node here.

```
class DLLNode<T>{  
  
    DLLNode<T> previous;  
    DLLNode<T> next;  
    T data;  
    DLLNode<T>(DLLNode<T> prev, DLLNode<T> nxt, T dat){  
        previous = prev;  
        next = nxt;  
        data = dat  }}  
  
public class DLL<T> {  
    DLLNode<T> first;  
    public void insertAfter(DLLNode<T> here, T newData{// fill in below  
        DLLNode<T> newNode = new DLLNode<T>(here, here.next, newData);  
        if (here.next != null){  
            here.next.previous = newNode; }  
        here.next = newNode;  
    }}  
}
```

7. The method `countPos` below must be a **recursive** method that takes a `Node head` as its argument, goes down the list headed by `head`, and counts the number of nodes which have a positive data field. If the input is: `head`  then the result of `countPos(head)` should return 2. Finish `countPos`.

```
public class Node{
    int data;
    Node next;
    public int countPos(Node head){

        if (head == null){
            return 0;
        } else {
            int recResult = countPos(head.next);
            if (head.data > 0){
                return recResult+1;
            } else {
                return recResult;
            }
        }
    }
}
```

8.

Assignment 1 involved classes Person and Friend, with instance variables and constructors as below. Finish the method hasMoreFriends below. If p1 and p2 are person objects, p1.hasMoreFriends(p2) returns true if p1's list of friends is longer than p2's list of friends. If p2's list is longer or if they are the same size it should return false. For full credit, your code should run in $O(\min(\text{number of P1's friends}, \text{number of p2's friends}))$.

```
public class Person {
    public String name;           // the person's name
    public Friend firstFriend;    // the first friend in the list of this
                                // person's friends
    public Person nextPerson;     // the next person in the list of people

    public Person(String name, Person nextPerson) {
        this.name = name;
        this.nextPerson = nextPerson;
    }
}

public class Friend {
    public Friend nextFriend;     // next Friend object
    public Person who;           // Person object who is the friend

    public Friend(Person who, Friend nextFriend) {
        this.who = who;
        this.nextFriend = nextFriend;
    }
}

public boolean hasMoreFriends(Person otherPerson){
    Friend friend1 = this.firstFriend;
    Friend friend2 = otherPerson.firstFriend;
    while (friend1 != null && friend2 != null){
        friend1 = friend1.nextFriend;
        friend2 = friend2.nextFriend;
    }
    return friend1 != null;
}
```