New Mexico State University

C S 272/463 Introduction to Data Structures

Midterm Exam

Wednesday, Oct 6, 10:30am-11:30am

Name:	
NMSU Email Address:	

This examination is closed book and notes. The examination duration is 1 hour. All students must answer all the questions. It contains 10 pages (including this one). The total exam value is 100 points. **No collaboration allowed on any exam in this course.**

Before you start, please check your copy to make sure it is complete. Turn in all pages, together, when you are finished. **Write your initials on the top of ALL pages**.

Question	Max	Score
1	20	
2	20	
3	20	
4	20	
5	20	
Total	100	

Question 1 (20 pts) Implement a class called `Location` that contains the following instance variables and methods:

1. the coordinate `x` (data type: double)
2. the coordinate `y` (data type: double)

You are required to implement the following methods:

a) One no-argument constructor. This constructor sets zero to `x` and `y`.
b) The get and set methods of all the instance variables
c) A method that computes the distance between two locations. The distance between Location p_1 and Location p_2, $d(p_1, p_2)$, is computed as follows:

$$d(p_1, p_2) = \sqrt{(p_1.x - p_2.x)^2 + (p_1.y - p_2.y)^2}$$

In Java, you can use `java.lang.Math.sqrt(double a)` to compute the square root.

Answer:

```
public class Location {

//(1.1) (2pts) Declare instance variables
    private double x; //1 pt
    private double y; //1 pt

//(1.2) (2 pts) Implement one no-argument constructor. This constructor sets zero to x and y.
    public Location() { //0.5 pt
        x = 0; //0.5 pt
        y = 0; //0.5 pt
    } //0.5 pt

//(1.3) (8 pts) Implement get and set methods of all the instance variables
    //each correct method 2 pts
    public double getX() { //0.5 pt
        return x; // 1 pt
    } //0.5 pt
    public double getY() { //0.5 pt
        return y; // 1 pt
```

```
    }//0.5 pt
    public double setX(double value) {//0.5 pt
        x = value; //1 pt
    }//0.5 pt
    public double setY(double value) {//0.5 pt
        y = value;//1 pt
    }//0.5 pt

//(1.4) (8 pts) A method that computes the distance between two locations
    public static double distance(Location p1, Location p2) { // 2pts
        x2 = Math.pow(p1.getX()-p2.getX(), 2); //2pts
        y2 = Math.pow(p1.getY()-p2.getY(), 2); //2pts
        return Math.sqrt(x2+y2); // 2pts
    }
}//end class Location
```

Question 2 (20 pts) Please express each of the following formula in big-O notation

2.1) (4 pts) $5 + 0.001n^3 + 0.025n$

Answer:

$O(n^3)$

2.2) (4 pts) $n^2 \log n + n(\log n)^2$

Answer:

$O(n^2 \log n)$

2.3) (4 pts) $n \log_3 n + n \log_2 n$

Answer:

$O(n \log n)$

2.4) (4 pts) $2000n^2 + 2^n$

Answer:

$O(2^n)$

2.5) (4 pts)

What is the worst-case big-O complexity of the following code fragment?

```
int ans =0;
for(int i=0; i<n ; i++){
      for(int j=0 j=1; j<n; j=j*2){//typo: j=0
            ans = ans + 1;
      }
}
```

Answer:

$O(\text{nlog}\, n)$

Due to the typo, all students get full mark for Question 2.5.

Question 3 (20 pts)

Given the class IntArrayBag, please answer the following questions.

```
public class IntArrayBag {
      //The invariant for this class is:
      //1. The number of elements in the bag is stored in the instance variable manyItems,
      //which is no more than data.length.
      //2. For an empty bag, we do not care what is stored in any of data; for a non-empty
      //bag, the elements of the bag are stored in data[0] through data[manyItems-1],
      //and we don't care what is stored in the rest of data.
      private int manyItems; // How much of the array is used
      private int [] data; // An array to store elements
      public IntArrayBag () { manyItems = 0; data = new int[10];}
      public IntArrayBag (int initialCapacity) {
            if(initialCapacity > 0) {
                  manyItems = 0;
```

```
                data = new int[initialCapacity];
            }
        }
        public void add(int element)
        {
            if (manyItems == data.length)
            { // Ensure twice as much space as we need.
                ensureCapacity(manyItems*2 + 1);
            }
            data[manyItems] = element;
            manyItems++;
        }
        public void ensureCapacity(int minimumCapacity)
        {
            int[ ] biggerArray;
            if (data.length < minimumCapacity)
            {
                biggerArray = new int[minimumCapacity];
                System.arraycopy(data, 0, biggerArray, 0, manyItems);
                data = biggerArray;
            }
        }
}
```

3.1 (10 pts) The `add` method uses `ensureCapacity(manyItems*2+1)`. What would go wrong if we forgot the "+1"?

Answer:

If `manyItems` happens to be zero, then `manyItems*2` would also be zero, and the capacity would not increase.

3.2 (10 pts) Write a new bag method that removes all copies of a specified target from a bag (rather than removing just one copy of the target). The return value should be the number of copies of the target that were removed from the bag.

Answer:

```
//using remove function
public int removeAll(int target) {
        int num = 0; //2pts
        while(remove(target)) {// 4pts
                num++; // 2pts
        }
        return num; // 2pts
}

//not using remove function
public int removeAll(int target){
        int num = 0; //2 pts
        for (int i = 0; i < manyItems; i++) { // 2pts
                if(data[i]==target) { // 1 pt
                        num++; // 1 pt
                        manyItems--; // 1 pt
                        data[i]=data[manyItems]; // 1 pt
                        i--; // 1 pt
                }
        }
        return num; // 1 pt
}//end removeAll
```

Question 4 (20 pts)

For the `IntNode` class, which is used to define a singly linked list, please answer the following questions.

public class IntNode{

```
private int data; // contain the real content
private IntNode link; // point to the current node's next node
public IntNode() { data = 10; link = null;}
public IntNode(int _data, IntNode _link) {data = _data; link = _link; }
public void addNodeAfter(int item) {link = new IntNode(item, link);}

//(4.1) (10 pts) Implement a static method with one parameter that is a head reference
for a linked list. The return value of the method is the sum of all the numbers on the list.
public static int sum(IntNode head)
{
        int sum = 0; // 2pts
        for(IntNode cursor = head; cursor!=null; cursor=cursor.link) // 2pts
        {
                sum += cursor.data; //2 pts
        }
        return sum; // 2pts
} What's the time complexity of your algorithm? _____O(n)  //2 pts_____

//(4.2) (10 pts)  Implement a method to add a new item to the end of the linked list.
public void addLast(int item) {
        IntNode pre = null; // 2.5 pts
        for(IntNode cursor = this; cursor!=null; cursor=cursor.link) // 2.5 pts
        {
                pre = cursor; // 2.5 pts
        }
        pre.addNodeAfter(item); // 2.5 pts
}
}
```

Question 5 (20 pts)

Given the following DNode class, which is used to define a doubly linked list, please answer the following questions.

```
public class DNode {
        public int data; //The value for the node
        public DNode next = null; //The next node of the current one
        public DNode prev = null; //The previous node of the current one
        public DNode(){;} //Constructor
}
public class DoublyLinkedListDummy {
        //The head and the tail of a doubly linked List, which are dummy nodes, NOT null
        public DNode head;
        public DNode tail;
        //constructor
        public DoublyLinkedListDummy ( ){
                head = new DNode ();
                tail = new DNode ();
                head.next = tail;
                tail.prev=head;
        }
```

//(5.1) (10 pts) Compare doubly linked lists with singly linked lists. When would you choose to use one rather than the other?

Answer:

Singly linked list: use less memory, the complexity of insertion and deletion at a known position is O(n).

Doubly linked list: can iterate in both directions, the complexity of insertion and deletion at a known position is O(1).

```
//(5.2) (10 pts) A method to find the mth-to-last element of the doubly linked List. Define
mth to last such that m=0, the last element of the list is returned.
public DNode findMthToLast(int m){
        int index = -1; // 2 pts
        for(DNode cursor = tail.prev; cursor!=head; cursor=cursor.prev) { // 2 pts
                index++; // 2 pts
                if(index == m){ // 2 pts
                        return cursor; // 1 pt
                }
        }
        return null; // 1 pt
} //end findMthToLast
}//end class DoublyLinkedListDummy
```

---END---