CSCI 151
Exam 1 Solutions
There are 7 questions and 100 total points. Remember to write the Honors Pledge on the last page.

1. [15 points] Here are two methods. Give a worst-case Big-Oh analysis of their performance in terms of the argument $n$. In each case give some justification or explanation of your answer.

```java
public static int A(int n) {
    int result = 1;
    for (int i = 0; i < n; i++) {
        int s = 0;
        for (int j = 0; j < 10; j++)
            s += result;
        result = s;
    }
    return result;
}
```

**A is $O(n)$. It does 10 things $n$ times.**

```java
public static int B(int n) {
    int result = 0;
    for (int i = 2; i <= n; i++) {
        boolean ok = true;
        for (int j = 2; j < i; j++)
            if (i%j == 0)
                ok = false;
        if (ok)
            result += 1;
    }
    return result;
}
```

**B is $O(n^2)$. It does (i-2) things where i is 2, 3, .... n. This is a total of $1+2+...+(n-2)$, which is $O(n^2)$.**
2. [15 points] Here are pictures of two different linked list implementations (A) is singly-linked, (B) is doubly-linked.

![Diagram of linked lists](image)

Here are some operations. For each give a worst-case Big Oh analysis of the running time on a list of size n:

<table>
<thead>
<tr>
<th>Operation</th>
<th>ArrayList</th>
<th>Singly-linked list</th>
<th>Doubly-linked List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert at the front</td>
<td>O(n)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td>Insert at the end</td>
<td>O(n)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td>Delete the first element</td>
<td>O(n)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td>Delete the last element</td>
<td>O(1)</td>
<td>O(n)</td>
<td>O(1)</td>
</tr>
<tr>
<td>Get the middle element</td>
<td>O(1)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
</tbody>
</table>
3. [10 points] Describe (in two or three English sentences) a sorting algorithm that will work for linked lists (you can choose whether these are singly-linked or doubly-linked). “Dump it into an array, sort the array and dump it back” is not an acceptable answer. Give a Big-Oh upper bound for the running time of your algorithm on a list of size n.

I would probably choose SelectionSort: Start pointer p out at the first node. Look through the list starting at p for the smallest node, then switch its data field with p’s. Change p to p.next, and repeat this process until p gets to the end of the list. This is O(n^2).

Other options: If you have a doubly-linked list you can do InsertionSort just as it works for arrays. With a singly-linked list you can do a version of InsertionSort where you start inserting from the front. MergeSort can be done for linked lists but needs a lot of extra computation to find the middle element. QuickSort works for doubly-linked lists just as it does for arrays.

4. [10 points] What are the two cardinal rules for recursion?
   a) You must have a base case and test for it before recursing.
   b) Your recursive calls should make progress towards the base case.

5. [10 points] What is the difference between an abstract class and an interface? Give an example where an interface is useful.

   Abstract classes can have data (class variables) and implemented methods; they just have one or more abstract methods. Interfaces have only method signatures; no executable code. A class can implement many interfaces, but only extend one abstract class. We use interfaces when we want to guarantee that a certain functionality is present. For example, all of our Sort routines specify that the objects being sorted must implement the Comparable interface.
6. [20 points] Here is a recursive method

```java
public static int H(int n) {
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else if (n%2 == 1)
        return 1 + H(n+1) + H(n-1);
    else
        return 2*H(n/2);
}
```

Give a dynamic programming version of this that eliminates the inefficiencies. If you add any arguments to the function write a sentence that says how they are initialized.

```java
public static int H(int n, int[] Values) {
    if (Values[n] >= 0)
        return Values[n];
    else if (n%2 == 1) {
        int t = 1 + H(n+1, Values) + H(n-1, Values);
        Values[n] = t;
        return t;
    } else {
        int t = 2*H(n/2, Values);
        Values[n] = t;
        return t;
    }
}
```
7. [20 points] Draw a picture of a linked implementation of a Queue of base type Integer.

```
  head  2  5  last
```

a) What class variables will your Node class contain?
   - Integer data
   - Node next

b) What class variables will your Queue class contain?
   - Node head, last

c) Give code for the method `dequeue()` . This should remove and return the element currently at the head of the queue.

```java
public Integer dequeue() throws NoSuchElementException {
    if (isEmpty())
        throw new NoSuchElementException();
    else {
        Node p = head.next;
        if (p == last)
            last = head;
        head.next = p.next;
        return p.data
    }
}
```
You can use this page for extra space for any problem.

Please write and sign the Honor Pledge when you have finished the exam.