CS171 Midterm Exam

October 29, 2012

Name:_________________

You are to honor the Emory Honor Code. This is a closed-book and closed-notes exam. You have 50 minutes to complete this exam. Read each problem carefully, and review your answers. Good luck!

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Earned</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Bonus</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>102</td>
<td></td>
</tr>
</tbody>
</table>

Below are some potentially helpful formulas.

\[
1 + 2 + 3 + \ldots + (n-1) + n = \frac{n \times (n+1)}{2}
\]

\[
1 + x + x^2 + \ldots + x^{n-1} + x^n = \frac{x^{n+1} - 1}{x - 1}
\]
Problem 1. (30 points) Give brief answers to the following questions.

1. (5 points) What are the main operations for stacks and queues?

   Push(), Pop() for stack, Enqueue(), Dequeue() for queue

2. (5 points) What are the advantage and disadvantages of arrays and linked list?

   Arrays:
   Disadvantages: Fixed size
   Advantages: stores elements continuously in memory, support indexes

   Linked list:
   Disadvantages: memory overhead
   Advantages: supports dynamic size

3. (10 points) Suppose we implement a stack using an array with initial capacity of 1 and double the size of the array to increase the capacity as necessary. What is the runtime cost in Big-O notation for pushing $N$ items into the stack? Show your reasoning. Assume that inserting an item into an array and copying an item from one array to another array each takes 1 unit of time.

   Every push takes O(1), increase takes O(N), so Big-O is O(N)

4. (10 points) What is the runtime cost of the following loop in Big-O notation given an input size $N$. Show your reasoning. Assume that each loop statement takes 1 unit of time, and the update of loop variable can be ignored.

   ```java
   for (int i = 0; i < N; i++)
       for (int j = i; j < 3*N; j++)
           System.out.println("1 iteration executed!");
   ```

   First for is O(N), second for is 3*N for every i, so:

   $$ N \cdot 3 \cdot N = O(N^2) $$
Problem 2. (20 points)

1. (10 points) Given an array of integers: 6, 4, 2, 7, 1, 5, 8, 3, show the key intermediate steps of how the array gets sorted.

   Solution: Just draw division to subsets and show merge them.

2. (5 points) What’s the cost function in recurrence relation for mergesort and what’s the base case? (You do not have to solve it.)

   Solution:
   \[ T(N) = 2 \cdot T(N/2) + N \]
   \[ T(1) = 0 \text{ or } T(1) = O(1) \]

3. (5 points) Briefly explain why improved mergesort uses insertion sort when the number of elements is small.

   Solution: The constant factors in insertion sort make it faster for small number of elements.
Problem 3. (25 points) Given a search key, a sorted array, and a search range specified by a lower bound and an upper bound index, a binary search proceeds by checking the value of the midpoint of the search range, repeating the search on the left half or the right half of the previous search range until the value is found or the search range is of size 1.

1. (15 points) Implement a method for the above binary search. Your method can be either recursive or non-recursive. The method should return true if the key is in the array, false otherwise.

Solution: version non-recursive:

```java
public boolean binsearch(int key, int[] a, int lower, int upper)
{
    int lo = lower;
    int up = upper
    while (lo <= hi) {
        int mid = lower + (upper - lower) / 2;
        if (key < a[mid]) lo = mid - 1;
        else if (key > a[mid]) lo = mid + 1;
        else return true;
    }
    return false;
}
```

2. (10 points) The runtime cost for the binary search is given as the following recurrence relation.

\[ T(N) = T(N/2) + 1 \]
\[ T(1) = 1 \]

Solve the recurrence relation, and derive the Big-O notation.

\[ T(N) = T(N/2) + 1 \]
\[ T(N) = T(N/4) + 1 + 1 \]
\[ T(N) = T(N/8) + 1 + 1 + 1 \]
\[ \ldots \]
\[ \ldots \]
\[ T(N) = T(N/(2^k)) + k \]
\[ T(1) = 1 \]
\[ N/(2^k) = 1 \]
\[ N = 2^k \]
\[ k = \log N \]

O(logN)
Problem 4. (25 pts) A double-ended linked list is a list that keeps a reference to both the head and tail element in the list. The structure of a list element and an (incomplete) list class are given by the following class definitions.

```java
public class ListElem {
    public int value;
    ListElem next; // "next" points to the successor
}

public class List {
    public ListElem head; // "head" points to the first element in list
    public ListElem tail; // "tail" points to the last element in list

    public List() // Constructor
    {
        head = null; // Empty list
        tail = null;
    }

    //... other methods ...
}
```
Hint: be careful and take care of the special *empty list* case for all the following questions.

1. (10 points) Complete the method for the *List* class that *inserts* an *ListElem* object `x` before the head of the list.

```java
public void insertAtHead(ListElem x){
    if (isEmpty()) tail = x;
    x.next = head;
    head = x;
}
```

2. (15 points) Complete the method for the *List* class that *deletes* a *ListElem* object with a given key. If the key does not exist in the list, the method should not change the list.

```java
public void delete(int key){
    ListElem current = head;
    ListElem previous = head;
    while (current != null && !current.item.equals(key)){
        previous = current;
        current = current.next;
    }
    // If current == head, set head as next
    if (current == head) head = head.next;
    // If current == tail, set tail as previous and tail.next as null
    if (current == tail){
        tail = previous;
        tail.next = null;
    }
    else if (current != null) previous.next = current.next;
}
```
Survey Question (2 bonus points) What’s your most and least favorite topic and assignment so far? Any feedback or suggestions about the class will be appreciated.