## CS200 Midterm 1
### Fall 2007

Name ________________________________

<table>
<thead>
<tr>
<th>Topic</th>
<th>Possible</th>
<th>Received</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE/ADTS/OO</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Grammars</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Stacks, Queues and Coding</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Complexity</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>Recurrence</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>Divide &amp; Conquer</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>100</strong></td>
<td></td>
</tr>
</tbody>
</table>
1. **Software Engineering/ADTS/ OO [15 points]**
   a. [10 points] Match the terms (draw a line between them) in the left column to the definitions on the right:

<table>
<thead>
<tr>
<th>i. verification</th>
<th>a. condition that is always true at a certain point in the program</th>
</tr>
</thead>
<tbody>
<tr>
<td>ii. invariant</td>
<td>b. what can be accessed from outside a class</td>
</tr>
<tr>
<td>iii. abstraction</td>
<td>c. formally confirming some functionality</td>
</tr>
<tr>
<td>iv. interface</td>
<td>d. UML link to indicate inheritance</td>
</tr>
<tr>
<td>v. generalization</td>
<td>e. separate implementation from use</td>
</tr>
</tbody>
</table>

   b. [5 points] Why are generics useful?

2. **Grammars [16 points]**
   a. [8 points] Given the alphabet \{A, B, C, D, E, F, G, H, I, O\} and the following grammar:

   \[
   \begin{align*}
   &\text{<S> = <S> <S> | <X> <Y> <X>} \\
   &\text{<X> = B | C | D | F | G | H} \\
   &\text{<Y> = A | E | I | O}
   \end{align*}
   \]

   i. Show the top down derivation for BADDOG
ii. Give one legal five character string over the alphabet.

b. [8 points] Write a grammar for postfix expressions (with operators \{+,-,\times,\div\} and variables \{x, y, z\}) that include balanced parentheses.

3. **Stacks, Queues and Coding [25 points]**
   a. [6 points] For a queue, pick one of these implementations from the notes/text and CIRCLE IT: singly linked list, doubly linked list, array, ArrayList. For the most efficient implementation using it, explain:
      i. To where are new elements added?

      ii. Compared to the other possible implementations, what is a problem with your choice?
b. [5 points] As the last question indicates, specific data structures such as ArrayLists can be used to implement Queues. Explain briefly how this can be done in such that the invariants of the Stack or Queue are not violated.

c. [14 points] For this Stack implementation which is taken from the text, fill in the code for the methods push and pop. (Some code has been omitted for space considerations; this is indicated with …)

```java
public class StackArrayBased implements StackInterface {
    final int MAX_STACK = 50;
    private Object items[];
    private int top;

    public StackArrayBased() { ... }
    public boolean isEmpty() { ... }
    public Boolean isFull() { ... }
    public void push(Object newItem) throws StackException {

    }
    public Object pop() throws StackException {

    }

    ... }
```
4. Complexity [13 points]
   a. [4 points] For the following functions, determine which big-O best characterizes it. Choose from this set:
      \( n^2, \log n, 1, n!, n^3, n\log_2 n, n^4, n, 2^n \)
      i. \( f(n) = 2 + 5 + \ldots + (3n - 1) \)
      
      ii. \( f(n) = 12n + 6\log_2 n \)
   
   b. [9 points] For the following algorithms, determine which big-O best characterizes it for the worst case. Choose from this set:
      \( n^2, \log n, 1, n!, n^3, n\log_2 n, n^4, n, 2^n \)
      i. An algorithm that prints all binary numbers with two zeros in them that are of length n.
      
      ii. An algorithm that calculates the average grade on a midterm with n students.
      
      iii. This algorithm counting print statements:
          
          ```java
          for i = 1 to n/2 {
              for j = 1 to 25 {
                  for k = 1 to i {
                      print i*j*k }
              }
          }
          while (n > 0) {
              print n*n;
              n--;
          }
          ```
5. **Recurrence and Divide-and-Conquer [17 points]**
   a. [5 points] Find a recurrence relation for the number of bit strings of length $n$ that do not have two consecutive 0s. Also list its initial conditions.

   b. [12 points] For MergeSort:
      i. What are the values of $a$, $b$ and $d$ from the Master Theorem counting compares as the key operation?
         
         $a =$ 
         $b =$ 
         $d =$ 
      
      ii. If MergeSort divided the array into *three* parts, what would be the values of $a$, $b$ and $d$?
         
         $a =$ 
         $b =$ 
         $d =$ 

6. **Advanced Sorting Algorithms [14 points]**
   a. [10 points] Using the QuickSort algorithm from the Walls text:
      i. Show how the following array looks after the first invocation of partition.

      | 15 | 5 | 12 | 32 | 1 | 3 | 42 | 12 |
      |----|---|----|----|---|---|----|----|

      ii. What value does `partition` return?

   b. [4 points] Radix Sort is $O(n)$. Why isn’t it the only sorting algorithm ever used?