Name: __________________________ 100 Points, mostly 5 each. Closed book and notes.

Logs and Exponents:
1. What is (approximately) the Log base 2 of 510,000,000?

2. What is (approximately) the Log base 10 of 10,000,000?

3. What is (approximately) 4 to the power of 15?

Big O:
4. What is Big O analysis?

5. When is Big O analysis most important?

6. Give the Big-O for these algorithms.
   - Binary Search on sorted data:
   - Merge Sort of mostly sorted data:
   - SAT solver by trying all combinations:
   - Multiplying two integers of N digits each:
   - Bubble Sort of random data:

7. If you are trying to speed up code that is running slowly, where is the bottle neck code likely to be?

8. Is it good practice to always optimize all your code? Explain why or why not?

9. Which Big O category does this code use?
   - For loop N times
     - Do $N^3$ work here.
   - For loop N times
     - For loop N times
       - Do $N$ work here.
   A. $N^3$
   B. $N^4$
   C. $N^5$
   D. $N^6$
Linked Lists:

10. The picture below represents a queue implemented using a circular linked list with a back (tail) pointer. Draw arrows where the new pointers should go after enqueueing the node that n points to. (Put it on the back, or right, of the queue.) Cross off old arrows. Also, number the changes, according to the order they must be implemented.

11. Write lines of Python code (in the correct order) which handle the enqueue operation steps of the previous question. Don't worry about the empty list special case. Assume node n is already stuffed with an object.

12. Give at least one advantage that a Linked List has over a Python List.
Sorting:
13. Show what happens to this array after one pass of the basic quick sort. Use the ‘N’ on the far left as the pivot, and use the same algorithm we studied in class.
   Original Array: N T A W F V C G

14. Fill out this table with Correct Big-O categories

<table>
<thead>
<tr>
<th></th>
<th>Expected Case</th>
<th>Worst Case</th>
<th>Best Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shaker Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quick Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Use this table for problems 15 and 16 below.

<table>
<thead>
<tr>
<th></th>
<th>BubbleSort</th>
<th>ShakerSort</th>
<th>MergeSort</th>
<th>QuickSort</th>
<th>MQuickSort</th>
<th>HashSort</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5.39</td>
<td>4.16</td>
<td>4.58</td>
<td>4.45</td>
<td>3.8</td>
<td>3.0</td>
</tr>
<tr>
<td>4</td>
<td>7.9</td>
<td>5.39</td>
<td>6.0</td>
<td>6.72</td>
<td>5.24</td>
<td>4.0</td>
</tr>
<tr>
<td>5</td>
<td>9.95</td>
<td>6.49</td>
<td>7.32</td>
<td>8.77</td>
<td>6.75</td>
<td>5.0</td>
</tr>
<tr>
<td>6</td>
<td>11.97</td>
<td>7.53</td>
<td>8.58</td>
<td>10.97</td>
<td>8.11</td>
<td>6.0</td>
</tr>
<tr>
<td>7</td>
<td>13.97</td>
<td>8.56</td>
<td>9.8</td>
<td>12.96</td>
<td>9.38</td>
<td>7.0</td>
</tr>
<tr>
<td>8</td>
<td>15.99</td>
<td>9.57</td>
<td>11.0</td>
<td>14.99</td>
<td>10.64</td>
<td>8.0</td>
</tr>
<tr>
<td>9</td>
<td>17.99</td>
<td>10.57</td>
<td>12.16</td>
<td>16.99</td>
<td>11.85</td>
<td>9.0</td>
</tr>
<tr>
<td>10</td>
<td>19.99</td>
<td>11.58</td>
<td>13.32</td>
<td>18.98</td>
<td>13.03</td>
<td>10.0</td>
</tr>
<tr>
<td>12</td>
<td>23.99</td>
<td>13.58</td>
<td>15.58</td>
<td>22.99</td>
<td>15.35</td>
<td>12.0</td>
</tr>
</tbody>
</table>

15. Given the chart above which measures compares on mostly sorted data, approximately how many compares does the MQuickSort require when the data is 1024 big? Give the answer as a base 10 number, not as a power of 2.

16. If you want to keep the number of compares at or below 256K on the BubbleSort, how big can your data set be? Give the answer as a base 10 number, not as a power of 2.

ADT’s:
17. Give real life examples of where a stack and a queue might be used.

18. Why is it good for a Computer Scientist to separate the ADT from the implementation?
Homework Review:

19. (10 points) Write Python code for CreateMostlySortedList, as we used in our homework assignment. As part of your solution, you may assume there is a working function named CreateRandomList, and you may use the built in Python sort function as well.

```python
def CreateMostlySortedList(size):
```