You may take this test with you afterwards, but you must turn in your “bubble form” answer sheet. Please be sure to write down your UIN number on the bubble form, left-justified with no spaces, in the section for ID number. Please be sure to write down the letters for your name as well as filling in the corresponding bubble under each letter.

This test has the following sections:
   I. True/False .......................... 60 points; (40 questions, 1.5 points each)
   II. Multiple Choice ................ 40 points; (15 questions, variable number of points each)
------------------------
100 points total

This test is worth 20% of your final grade. You must put your answers on the bubble form. You are allowed to have resources with you printed on paper, but no computers. For the multiple choice problems, select the best answer for each one and select the appropriate letter on your answer sheet. Be careful - more than one answer may seem to be correct. Some questions are tricky.

True/False: (40 questions, 1.5 points each) On your bubble form fill out A for true and B for false.

   T F  1.  An algorithm that is O(n^3) will take longer for all values of n as compared to an algorithm that is O(2^n).
   T F  2.  Assume you are in a timed competition using a O(2^n) program that will be run 60 times for values of n>40. It is worth the investment to spend 3 weeks to come up with a O(log n) version of the program instead (assuming this is possible).
   T F  3.  There are generally tradeoffs between algorithms and data structures.
   T F  4.  Adding a dummy sentinel node to a linked list allows us to simplify the code used to insert, delete and find list elements.
   T F  5.  Adding a dummy sentinel node to a linked list makes the code more complex, but enables faster run-times when inserting, deleting and finding list elements.
   T F  6.  Consider adding a large number of random elements to an ordered linked list, and then deleting random elements one-by-one. This is generally faster when implemented using a linked list as compared to using dynamically allocated arrays.
   T F  7.  Skip Lists can give O(log n) behavior although we are using a linked list implementation.
   T F  8.  A stack can be used to manually implement recursion when using a programming language that otherwise doesn’t allow recursion.
   T F  9.  Three nodes can be combined to create at most 3 distinct binary trees.
   T F 10.  A stack is needed to do a breadth-first tree traversal on a normal (non-threaded) binary tree.
   T F 11.  A binary tree and a binary search tree are two different names for the same thing.
   T F 12.  The number of levels on the left side of a binary search tree compared to the number of levels on the right side is always either the same or differing by at most 1.
13. When creating a binary search tree, the order in which values are added changes what the tree looks like.  

14. A complete binary tree that has 4 levels (where the root is level 1) and has the bottom level completely filled out has 7 internal nodes.

15. In a complete binary tree that has 4 levels (where the root is level 1) and has the bottom level completely filled out, over half of the nodes are internal nodes.

16. Every full binary tree is also a complete binary tree.

17. Every complete binary tree is also a full binary tree.

18. On average it would take longer to insert values into an AVL tree than it does to insert the same values into a min-heap.

19. Consider accessing $n$ values that are already stored and then printing them out in ascending order. On average it would be quicker to do this using an AVL tree than it would using a min-heap. Could simply traverse the AVL tree.

20. A min-heap can be implemented using an array, but it takes the same amount of storage as compared to a linked node representation.

21. We can find and display the three smallest values in a min heap in constant time.

22. Finding the out-degree of a vertex for a sparse graph on average is faster when representing the graph with adjacency lists as compared to when using an adjacency matrix.

23. Finding the in-degree of a vertex in a sparse graph can be found in less than $O(n)$ time when using adjacency lists. True only if in and out lists are kept separately.

24. While only half of an adjacency matrix needs to be used when representing an undirected graph, the whole adjacency matrix must be used when representing a directed graph.

25. A depth-first-search can only be done on trees, but cannot be done on graphs.

26. Dijkstra's algorithm is more similar to Kruskal's algorithm than it is to Prim's algorithm.

27. Twice the length of a minimum spanning tree (MST) gives an upper bound on the traveling salesman problem (TSP).

28. Greedy algorithms cannot be used to give an optimal solution to some problems.

29. A single run of Dijkstra's algorithm provides the shortest distances between all pairs in a graph. Single-point all pairs is true, however

30. Prim's algorithm is guaranteed to give a minimum cost spanning tree solution.

31. When all edges have unique distance values, Prim's and Kruskal's algorithms will always give the same solution.

32. Both Morse code and early telephone area codes are variable-length codes.

33. Hollerith used a bin-sorting technique that started with the least-significant digits and moved one digit at a time to the left.

34. A binary trie only has keys stored on leaf nodes.
T F 35. In a compressed binary trie all internal nodes have in-degree and out-degree of 1.
T F 36. A Patricia trie combines the efficiency of storage of a compressed binary trie along with combining internal and leaf nodes.
T F 37. One advantage of hashing is that values can be put into place without needing to sort all values first.
T F 38. When hashing, generally the table size should be a prime number larger than the number $n$ of elements to be stored.
T F 39. When using hashing and handling collisions using linear probing, if an element is deleted, all subsequent elements that might have hashed to that same location must also be checked and possibly moved.
T F 40. The number of levels on the left side of an AVL tree compared to the number of levels on the right side is always either the same or differs by at most 1.

Multiple Choice (15 questions, variable number of points each)

41) (2 points) Given the seven integer values 1..7 inserted in ascending order into a binary search tree, which type of traversal will not visit the nodes in the order: 1 2 3 4 5 6 7 ?

a) In-order
b) Pre-order
\[\overline{c)} Post-order\]

42) (2 points) Consider the differences between a priority queue and a heap. Which of the following statements are true?

\[\checkmark 1. Both can be implemented using a dynamically sized array\]
\[\checkmark 2. Both can provide the smallest element in constant time\]
\[\longleftarrow 3. Both can be used to print out the stored elements in ascending order in O(n log n)\]

a) 1 only
b) 2 only
\[\overline{c)} 1 and 2\]
d) 1, 2 and 3

43) (2 points) Assume you are using Dijkstra’s algorithm. What would be the consequence of having a negative-weight path somewhere in the tree?

a) It would always still terminate and always provide a solution
b) It would always still terminate, though the solution would not be correct
\[\overline{c)} It would work only when the negative weight path is small enough of a magnitude to not change the weight on any other path besides the one currently being considered\]
d) Dijkstra’s algorithm can never be used when there is a negative weight path
44) (3 points) Consider using a binary trie to store two \( n \) bit keys. In the worst case, after storing the two keys, how many levels would there be in the trie, where the root is counted as level 1?

- a) \( n-1 \)
- b) \( n \)
- c) \( n+1 \)
- d) \( \log n \)
- e) \( 2^n \)

45) (4 points) Assume the following values are inserted in the following order into an AVL tree:

\[
6 \quad 9 \quad 8 \quad 3 \quad 7 \quad 4
\]

If you did a breadth-first traversal of the resulting tree, what would the output be?

- a) 8 3 9 6 7 4
- b) 6 3 9 4 8 7
- c) 8 6 9 3 7 4
- d) 6 3 8 4 7 9
- e) None of the above

46) (2 points) What is a primary similarity between 1) The choice of digits used in area codes for analog phones and 2) The pattern of dots and dashes used for Morse code?

- a) The most frequently used area code phone digits and Morse code letters are represented using the fewest symbols (clicks and Morse code dots and dashes)
- b) Smaller digits and letters earlier in the alphabet are represented using fewer symbols
- c) Larger digits and letters later in the alphabet are represented using fewer symbols
- d) The number of symbols in both area codes and Morse code alternates between least frequent and most frequent, to give better average performance.

47) (4 points) Starting at vertex A in the graph at right, what is the list of vertices in the order in which they would be visited using Prim’s algorithm to find a Minimum Spanning Tree?

- a) A B C D E F G
- b) A B C F D G E
- c) A D C E B G F
- d) A D C E F G B
- e) None of the above
Consider the graph shown below. Starting from vertex A fill in the table below using Dijkstra’s algorithm to show each step from vertex A to all other points, similar to what was done in class and in lab.

The first row has been done for you.

<table>
<thead>
<tr>
<th>S</th>
<th>Vertex Selected</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A  B  C  D  E  F  G</td>
</tr>
<tr>
<td>{}</td>
<td>A</td>
<td>0  11  8  3  maxint maxint maxint</td>
</tr>
<tr>
<td>{A}</td>
<td>D</td>
<td>0  11  8  3  maxint 13 maxint</td>
</tr>
<tr>
<td>A,D</td>
<td>C</td>
<td>0  11  5  3  11 13 maxint</td>
</tr>
<tr>
<td>A,B,C</td>
<td>B</td>
<td>0  11  5  3  12 13 18</td>
</tr>
<tr>
<td>A,B,C,E</td>
<td>E</td>
<td>0  11  5  3  12 13 18</td>
</tr>
</tbody>
</table>

48) (1 points) After filling out the table, what are the 3 values, top-down, you wrote down in the “Vertex Selected” column (highlighted with a rectangle)?
   a) D  b) C  c) C  d) D  e) None of these

49) (2 points) After filling out the table, what are the values, left-to-right, in the two-element circle of the table?
   a) 3 17  b) 3 14  c) 3 13  d) 3 12  e) None of the above

50) (2 points) After filling out the table, what are the values in the 4-value circle of the table?
   a) maxint maxint  b) 13 maxint  c) 13 20  d) 13 18  e) None of these
For the next three problems find the Huffman code for the following sample text string: 
abracadabra

51. (3 points) How many bits are required for a Huffman-encoding of this string?
   a) 22
   b) 23
   c) 25
   d) 27
   e) None of the above

52. (2 points) How many bits are required for an ASCII-encoded file, assuming 7 bits per character?
   a) 64
   b) 77
   c) 88
   d) 128
   e) none of the above

53. (3 points) Assume the dictionary takes 20 bits, and assume any additional message characters would all be the letter ‘a’. What is the minimum message length using only the letters seen above (and possibly some additional ‘a’ characters) where Huffman encoding would take fewer bits than ASCII?
   a) 11
   b) 12
   c) 13
   d) 17
   [I made a mistake on this and intended for the dictionary to be 55 bits, which would change the answer]

54. (2 points) Consider creating an alphabetic trie for the following words:
   about, abort, abound, able, abe
   How many nodes would there be in this alphabetic trie, including the root node?
   a) 10
   b) 11
   c) 12
   d) 13
   e) 14

The whole root node thing threw off a lot of people, so I also accepted c) 12
Consider the table given below to help you do a trace of different methods of collision handling for hashing, and the values shown to be inserted. Assume the hash function is: value % 10.

Values to insert: 9 20 30 29 21 19 22

<table>
<thead>
<tr>
<th>Index</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Chaining</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>30</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>29</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>

55) (2 points) How many collisions are there when linear probing is used?
   a) 0
   b) 1 to 5
   c) 6 to 10
   d) 11 to 15
   e) more than 15

56) (2 points) How many collisions are there when quadratic probing is used? After the first collision look ahead $1^2=1$ space, and if that is full look ahead an additional $2^2=4$ spaces, and so on.
   a) 0
   b) 1 to 5
   c) 6 to 10
   d) 11 to 15
   e) more than 15

57) (2 points) How many collisions are there when chaining is used? Assume that each list element comparison that does not match is counting as a collision.
   a) 0
   b) 1 to 5
   c) 6 to 10
   d) 11 to 15
   e) more than 15