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.......................... 20 points; (20 questions, 1 point each)
II. Multiple Choice................ 80 points; (20 questions, 4 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:** (20 questions, 1 point each) On your bubble form fill out A for true and B for false.

T F 1. We can find and display the two smallest values in a min heap in constant time.

T F 2. The quicksort routine included in stdlib.h requires using a pointer to a function for doing a comparison.

T F 3. While the stdlib.h built-in quicksort is fast, it is limited to sorting integer values.

T F 4. Mergesort with extra storage of size n has time complexity O(n).

T F 5. Mergesort in-place has time complexity O(n).

T F 6. The mergesort in-place algorithm we studied in class partitions the list elements into sqrt(n) blocks.

For the following four questions consider the following two lists of 8 elements each, stored a single array, which are to be merged in-place using the merge sort code we discussed in class:

```
ceimnpuvdfgjswyz
```

T F 7. The first step is to find the four largest elements out of both lists, putting those at the end of the first list, with the swapped-out elements placed at the end of the second list. Those two groups of 4 elements each are then respectively sorted.

T F 8. After the above step, the next step is to sort the blocks in order according to their last element.

T F 9. Just before we begin the first merge step, the list should look like:

```
vwyzdfgjceimnpsu
```

T F 10. Once merging begins, the second and third blocks are merged, with the destination starting as the first block.

T F 11. Both Morse code and ASCII are variable-length codes.
T F 12. Hollerith used a bin-sorting technique that started with the most-significant digits and moved one digit at a time to the right.

T F 13. When creating a binary search tree, the order of values added changes what the tree looks like.

T F 14. A binary trie only has keys stored on leaf nodes.

T F 15. A complete and full binary tree that has 3 levels (where the root is level 0) has 7 internal nodes.

T F 16. In a complete and full binary tree that has 3 levels (where the root is level 0) over half of the nodes are leaf nodes.

T F 17. A compressed binary trie has no internal nodes with degree of 1.

T F 18. A Patricia tree combines the efficiency of storage of a binary trie along with the elimination of internal nodes with degree of 1, which a compressed representation has.

T F 19. When using hashing and handling collisions using linear probing, if an element is deleted, all other elements that might have hashed to that same location must also be checked and possibly moved.

T F 20. The advantage of AVL trees is that they always keep the left and right side of every tree and subtree perfectly balanced.

Multiple Choice (20 questions, 4 points each)

21) Assume you have a stream of n integer values in descending order that you want to insert into a binary search tree. What is the run-time complexity (Big-Oh) of running this program on this data?

a) $O(c)$

b) $O(n^2)$

c) $O(n \log n)$

d) $O(\log n)$

e) $O(n)$
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>A</td>
<td>0</td>
</tr>
<tr>
<td>{A}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

22) 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) B  b) C  c) E  d) D  e) None of these

23) After filling out the table, what are the values, left-to-right, in the two-element circle of the table?
   a) 6 4  
   b) 5 1  
   c) 4 6  
   d) 1 2  
   e) None of the above

24) After filling out the table, what are the values in the 4-value circle of the table?
   a) 2 7  
   b) 7 5  
   c) 5 4  
   d) 7 6  
   e) None of these
Given the following directed graph, fill in the values for the packed array representation of an adjacency multi-list in the table at right below. Handle all elements in ascending vertex order.

<table>
<thead>
<tr>
<th>Index</th>
<th>From</th>
<th>To</th>
<th>Outlink</th>
<th>Inlink</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

25) What are the 6 numbers in the highlighting **rectangle** at left in the chart above?

a) 2 2  
b) 1 0  
c) 1 0  
d) 0 1  
e) None of these

0 1  
1 0  
2 1  
2 0  
1 2

26) What are the 6 numbers shown in the highlighting **circle** at right in the chart above? (Note that the slash character ‘/’ represents NULL.)

a) 4 3  
b) 3 4  
c) 5 6  
d) 6 5  
e) None of these

6 5  
5 6  
7 /  
/ 7  
/ 7  
6 5  
5 6

27) Consider the list shown below, where the merge part of the mergesort algorithm discussed in class is being used. Assume the list has already been set up properly in preparation for the first merge step:

```
v x y z   d f g k   c e h m   n p s t
```

What will the list look like after the first merge step has completed?

a) cdef xzgk vyhm npst  
b) cdef gkxz vyhm npst  
c) cdef gkxz hmvy npst  
d) cdef ghxk vyzm npst  
e) None of the above
28) Continuing with the above example, what are the pointers to the merge destination segment and the next two blocks to be merged?
   a) cdef xzgk vyhm npst
      ↑ ↑    ↑
   b) cdef gkxz vyhm npst
      ↑ ↑    ↑
   c) cdef gkxz hmvy npst
      ↑  ↑    ↑
   d) cdef ghkx vyzm npst
      ↑    ↑ ↑
   e) None of the above

29) 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.

For the next three problems find the Huffman code for the following sample text string:
   madam im adam

Assume the the first 8 bits in the Huffman encoded file is a number representing the size of the
dictionary, and assume each Huffman dictionary entry is 12 bits.

30) How many bits are required for a Huffman-encoded file?
   a) 83
   b) 85
   c) 91
   d) 97
   e) None of the above

31) How many bits are required for an ASCII-encoded file, assuming 7 bits per character?
   a) 83
   b) 85
   c) 91
   d) 97
   e) None of the above

32) Using the dictionary created in the above example, what is the minimum message length where
    Huffman encoding takes fewer bits than ASCII?
   a) 11
   b) 13
   c) 14
   d) 17
Consider adding letters one at a time to create a binary trie using the following representation for letters of the alphabet:

<table>
<thead>
<tr>
<th>Letter</th>
<th>Binary</th>
<th>Letter</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 00001</td>
<td>N</td>
<td>13 01110</td>
</tr>
<tr>
<td>B</td>
<td>1 00010</td>
<td>O</td>
<td>14 01111</td>
</tr>
<tr>
<td>C</td>
<td>2 00011</td>
<td>P</td>
<td>15 10000</td>
</tr>
<tr>
<td>D</td>
<td>3 00100</td>
<td>Q</td>
<td>16 10001</td>
</tr>
<tr>
<td>E</td>
<td>4 00101</td>
<td>R</td>
<td>17 10010</td>
</tr>
<tr>
<td>F</td>
<td>5 00110</td>
<td>S</td>
<td>18 10011</td>
</tr>
<tr>
<td>G</td>
<td>6 00111</td>
<td>T</td>
<td>19 10100</td>
</tr>
<tr>
<td>H</td>
<td>7 01000</td>
<td>U</td>
<td>20 10101</td>
</tr>
<tr>
<td>I</td>
<td>8 01001</td>
<td>V</td>
<td>21 10110</td>
</tr>
<tr>
<td>J</td>
<td>9 01010</td>
<td>W</td>
<td>22 10111</td>
</tr>
<tr>
<td>K</td>
<td>10 01011</td>
<td>X</td>
<td>23 11000</td>
</tr>
<tr>
<td>L</td>
<td>11 01100</td>
<td>Y</td>
<td>24 11001</td>
</tr>
<tr>
<td>M</td>
<td>12 01101</td>
<td>Z</td>
<td>25 11010</td>
</tr>
</tbody>
</table>

33) If a binary trie is created using the word: CENTER then how many levels are in the trie, assuming the root is level 0?
   a) 1
   b) 2
   c) 3
   d) 4
   e) 5

34) If a binary trie is created using the word: CENTERED then how many levels are in the trie, again assuming the root is level 0?
   a) 1
   b) 2
   c) 3
   d) 4
   e) 5

35) If a binary trie is created using the word: CENTERED then what are the letters in the trie leaf nodes, reading left-to-right?
   a) TRND
   b) TREND
   c) CDENR
   d) DENRT
   e) None of the above
Given the following Patricia tree, what does the resulting tree look like after inserting 0110?

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.

Items to insert: 40 19 58 18 79 38 21

<table>
<thead>
<tr>
<th>Index</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Chaining</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

37) 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
38) 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

39) How many collisions are there when chaining is used? Assume we keep end-of-list pointers so that adding to a list is done in constant time, without it counting as a collision.

a) 0
b) 1 to 5
c) 6 to 10
d) 11 to 15
e) more than 15

40) Assume the following values are inserted in the following order into an AVL tree:

```
10  5  7  2  3  6
```

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

a) 2 3 5 6 7 10
b) 7 3 10 2 5 6
c) 7 5 10 3 6 2
d) 5 3 7 2 6 10
e) None of the above