You may take this test with you afterwards, but you must turn in your “bubble form” answer sheet.

This test has the following sections:

I. True/False ......................... 50 points; (25 questions, 2 points each)
II. Multiple Choice ................ 50 points; (12 questions, various points each)

100 points total

This test is worth 15% 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: (2 points each)** On your bubble form fill out A for true and B for false.

T  F  1. The order of an algorithm can be determined by repeatedly running a program.

T  F  2. When it comes to determining which Big-Oh dominance class a program is in, constants can be ignored.

T  F  3. $n$ unique consecutive values could be sorted in $O(n)$ time by using an additional array of size $n$

T  F  4. Assume a binary search tree has been built using $n$ integer values. Regardless of the distribution of the input values used to build the tree, any value can be found in $O(n)$ time or better.

T  F  5. An algorithm that runs in $O(2n)$ runs faster than another version that runs in $O(4n)$.

T  F  6. Every full binary tree is also a complete binary tree.

T  F  7. Twice the length of a minimum spanning tree (MST) is an upper bound on the Traveling Salesman Problem (TSP).

T  F  8. Prim’s algorithm guarantees a minimum length solution, but Kruskal’s algorithm does not.

T  F  9. For any graph, a Minimum Spanning Tree (MST) can always be found using a greedy algorithm.

T  F  10. To calculate the shortest path from one node to all others in an unweighted graph we can use a breadth-first traversal.

T  F  11. We can always find the smallest value in a min heap in constant time.

T  F  12. Given a min heap with more than two values, we can always find the two smallest values in a min heap in constant time.
T F 13. The following code using an explicit stack will display theNumber equivalent in binary, with the digits in the correct order.

```c
void decimalToBinaryStack( int theNumber) {
  Node *pHead = NULL;
  
  while( theNumber > 0) {
    push( pHead, theNumber%2);
    theNumber = theNumber / 2;
  }

  while( pHead!=NULL) {
    printf("%d", pop(pHead));
  }
}
```

An Eulerian walk is a trail in a graph which visits each edge exactly once.

T F 14. There is a Eulerian walk for the following graph:

![Graph](image)

T F 15. There is a Eulerian walk for the following graph:

![Graph](image)

T F 16. A heap of \( n \) elements stored using an array representation in an array of size \( n \) takes less storage than an equivalently sized linked representation.

T F 17. A graph of interstate highways would be better represented by an undirected graph, as compared to a directed graph.

T F 18. The social network of UIC CS students would tend to be a sparse graph.

T F 19. In a weighted graph a path that is twice as long could be half the weight.

T F 20. Compared to an adjacency matrix, an adjacency list for a sparse graph gives better performance at finding the predecessors to a vertex.

T F 21. A complete graph with 4 vertices has 4 edges.

T F 22. A complete directed graph with 4 vertices has 8 edges.

T F 23. The sum of ones in a column for an adjacency matrix is the in-degree for that column’s vertex.

T F 24. When using an adjacency matrix to represent a graph, an algorithm to find the number of edges in the graph can be done in \( O(n^2) \) time.
Finding the \textit{in} values for a node in a graph represented using regular adjacency lists requires scanning all the values on all the lists.

\textbf{Multiple Choice (50 points total, different numbers of points for different problems)}

\textbf{26) (4 points)} The expression \( A \times (B + D \times E) / F + (G + H) \times I \) converted to postfix is:

\begin{itemize}
  \item [a)] \(ABDE**F/GH*I++\)
  \item [b)] \(ABDE**F/GHHI*+\)
  \item [c)] \(AB+DE**F/GHI++\)
  \item [d)] \(AB+DE**F/GH+I**\)
  \item [e)] None of the above
\end{itemize}

\textbf{27) (2 points)} 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?

\begin{itemize}
  \item [a)] \(O(c)\)
  \item [b)] \(O(n^2)\)
  \item [c)] \(O(n \log n)\)
  \item [d)] \(O(\log n)\)
  \item [e)] \(O(n)\)
\end{itemize}

For the following two problems consider the graph shown at right, implemented using adjacency lists, where each list contains nodes in alphabetical order.

\textbf{28) (4 points)} What is the order in which vertices in the above graph are visited when doing a \textit{depth-first} traversal of the graph, starting at vertex A?

\begin{itemize}
  \item [a)] A B C D E F G
  \item [b)] A B E C F D G
  \item [c)] A B C D F E G
  \item [d)] G F E D C B A
  \item [e)] None of the above
\end{itemize}

\textbf{29) (4 points)} What is the order in which vertices in the above graph are visited when doing a \textit{breadth-first} traversal of the graph, starting at vertex A?

\begin{itemize}
  \item [a)] A B C D E F G
  \item [b)] A B E C F D G
  \item [c)] A B C D F E G
  \item [d)] G F E D C B A
  \item [e)] None of the above
30) (5 points) How many different ways can the set of input values \{8, 11, 13, 22, 26, 29, 30\} be presented and still result in the binary search tree shown below?

![Binary Search Tree](image)

a) 1  
b) 2  
c) 4  
d) 8  
e) More than 8

31) (4 points) Consider a weighted graph represented using adjacency lists, with the same weights on several different edges. Two different programs give minimum spanning trees with different sets of edges in the solution. What is the most plausible reason for this?

a) The two programs use different starting points  
b) One uses adjacency matrices, while the other uses an adjacency list  
c) One of the programs is using Kruskal’s algorithm, while the other is using Prim’s algorithm  
d) The adjacency lists in the two programs are in different orders  
e) One is using local weighting for a greedy algorithm, while the other is not using a greedy algorithm.

32) (4 points) Consider a binary tree T in which all the levels are completely full and the nodes are numbered in a breadth-first fashion, where the root is 1, its immediate left and right children are 2 and 3, and their children respectively are 4, 5 and 6, 7, and so on. Which of the following are true about T, where it is implemented as an array?

A. To maintain the efficiency of representation Nodes can only be deleted from the end of the array.  
B. It would require less storage than using a tree of Node structures  
C. The index of a node can be used to determine the tree level of the node  
D. Accessing any node can be done in constant time

a) A, B  
b) A, C, D  
c) A, B, D  
d) B, C, D  
e) A, B, C, D
33) (5 points) Consider the max Heap shown at right:

Which of the heap values could be the first one added to the heap and still end up with the tree shown at right?

a) Only 17
b) Only 13, 15 or 17
c) Only 2, 11, 15, or 17
d) Only 8, 9, 13 or 17
e) Any of the values could be the first one

For the following 3 problems if it makes a difference assume the graph is stored using adjacency lists, where vertices are stored in ascending distance order. If necessary the alphabetic order of the nodes is used as a secondary sorting criterion.

34) (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 C D H G E F B
b) A C D H G F E B
c) A C D H G E G B
d) A C D H G F E B
e) None of the above

35) (4 points) Select the edges in the order in which they would be added for Kruskal’s algorithm to find a Minimum Spanning Tree, just as we did in class. (The above graph is given again here for your convenience so you can draw on it.)

What is the order of edges added?

e) None of the above
Consider the graph shown at right. 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 Selected</th>
<th>Vertex</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>0 5 4 6 12 maxint maxint maxint</td>
</tr>
</tbody>
</table>

36) (5 points) After filling out the table, what are the 2 values (top down) you wrote down in the “Vertex Selected” column that are highlighted with a rectangle?
   a) C B
   b) B C
   c) B D
   d) D B
   e) None of the above

37) (5 points) After filling out the table, what are the values, left-to-right, in the circled cells of the table?
   a) 12 6 8 6
   b) 10 6 7 6
   c) 9 6 8 6
   d) 9 6 7 6
   e) None of the above