CMSC132 Summer 2018 Final Exam

Name (PRINT): ____

Instructions

- This exam is a closed-book and closed-notes exam.
- Total point value is 100 points.
- The exam is a 80 minutes exam.
- Please use a pencil to complete the exam.
- **WRITE NEATLY**. If we cannot understand your answer, we will not grade it (i.e., 0 credit).

1. T/F	/ 15
2. Multiple Choice	/ 20
3. Short Answer	/ 40
4. Programming	/ 25
Total	/

1. [15 pts] True/False and Filling the Blank Questions (1 point each)

1) T / F The merge operation required for mergesort is an O(nlog(n)) operation.

2) T / F The execution order of threads cannot always be predicted.

3) T / F A 2-3-4 tree is always balanced.

4) T / F On average, hash tables have O(1) item retrieval.

5) T / F Minimum spanning tree has no cycles

6) T / F In order to insert elements into a binary search tree, items, must be comparable, or a comparator must be provided

7) T / F Both DFS and BFS can detect if a vertex t is reachable from a given vertex s.

8) T / F A DFS cannot find unconnected Nodes in a graph.

9) T / F Finding the height of a red black tree can be done in O(log(n)) time.

10) T / F It is possible for a binary search tree with more than one node to also be a heap.

11) T / F A static variable of a given class is only accessible after the first time an object of that class has been instantiated

12) Time complexity of binary search of a sorted array O( )

13) T / F If there is a negative edge on the graph, Dijkstra's shortest path algorithm may not work correctly.

14) To provide guaranteed performance, we have to use ________________________
Red & Black Tree Hash Table

15) T / F If edge weights are distinct, the minimum spanning tree of any given graph is unique.

2.[20 pts] Multiple Choice Questions

1) [2 pts] What is the value of mystery(3)?

```
public int mystery(int n){
          if(n == 1) return 2;
          else return 2 + mystery(n-1);
}
```

  a) 4
  b) 6
  c) 8
  d) 10

2) [2 pts] What is the complexity of the following code in terms of **n**?

```
int sum = 0;
for (int i = 0; i < n; i++)
      sum += i;
for (int j = 0; j < n; j++)
      for (int k = 0; k < n; k++)
            sum += j*k;
```

  a) O(n)
  b) $O(n^2)$
  c) $O(n^2 \log n)$
  d) $O(n^3)$

3) [2 pts] ____________________ improves the performance of the Dijkstra's shortest path algorithm.

  a) ArrayList
  b) LinkedList
  c) Set
  d) Priority Queue

4) [2 pts] We can calculate the longest path if the graph has

  a) No directed cycles
  b) No negative cycles
  c) No negative edges
  d) No self-loops

5) [2 pts] The preorder traversal sequence of a binary search tree is 30, 10, 15, 25, 23, 27, 40, 39, 35, 42. Which one of the following is the postorder traversal sequence of the same tree?

  a) 23,27,25,15,10,35,42,40,39,30
  b) 23,27,25,15,10,35,39,42,40,30
  c) 23,27,15,25,10,35,42,39,40,30
  d) 23,27,25,15,10,30,35,42,39,40

6) [2 pts] Which of the following applications may use a queue?

   a) Parentheses balancing program.
   b) Keeping track of local variables at run time.
   c) Iterative preorder traversal of a binary tree
   d) Level order traversal of a binary tree

7) [2 pts] Suppose B is a binary heap with 14 nodes. What is the height (number of edges in the longest path) of B?

   a) 13
   b) 3
   c) 4
   d) 14

8) [2 pts] Which of the following binary trees has an inOrder traversal of 17, 20, 4, 22, 13, 1, 9?

```
a)      (17)            b)      (17)            c)      (9)             d)      (22)
       /     \                  /     \                 /     \                 /     \
    (20)      (4)            (20)     (13)           (4)      (1)           (20)      (1)
    /  \      /  \           /  \     /  \           /  \     /  \          /  \      /  \
(22)  (13)  (1)  (9)      (4)  (22) (1)   (9)    (17) (20) (22) (13)    (17)  (4)  (13)  (9)
```

9) [2 pts] What is the complexity of the following code in terms of **n**?

```
for(int i = 0; i < 100; i ++)
      for(int j = N; j >= 0; j --)
            print(“bark”)
```

   a) O(NlogN)
   b) O(1)
   c) O(N)
   d) O(logN)

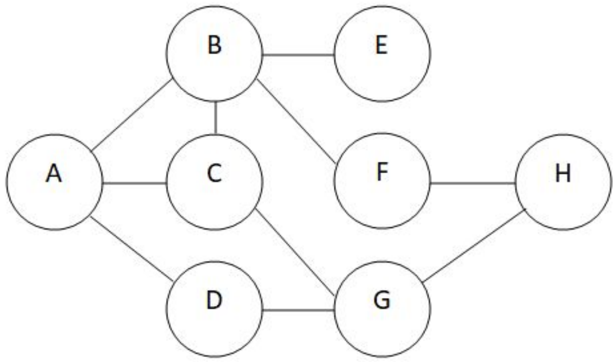
10) [2 pts] What is the complexity of the following code in terms of **n**?

```
      for(int i = 0; i < N; i += 2)
            for(int j = 1; j < N; j *= 2)
                  print(“go”);
```

   a) O($\sqrt{N}$)
   b) O(N^2)
   c) O(NlogN)
   d) O(logN)

3.[40 pts] Short Answer

1)[4 pts] Starting at vertex A and print the vertices in the order they are processed by **DFS**. As usual, assume the adjacency lists are in lexicographic order, e.g., when exploring vertex F, the algorithm considers the edge F→A before F→G.



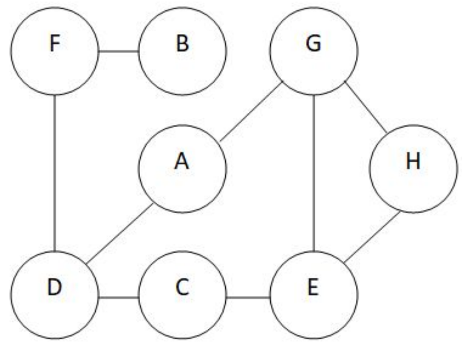
A							

2) [3 pts] Fill in the array after the FIRST iteration of the quick sort (sorting from smallest to largest, pivot is 5).

Original	5	3	8	9	1	7	0	2	6	4
After 1 iteration										

3) [4 pts] Build a max-heap by inserting 91, 77, 46, 68, 81, 11

4) [4 pts] Run a **breadth first search** through the following graph starting with node A, and record the order that each node is visited. If a node has more than one neighbor, the next node to visit will be the one that comes first lexicographically.



A							

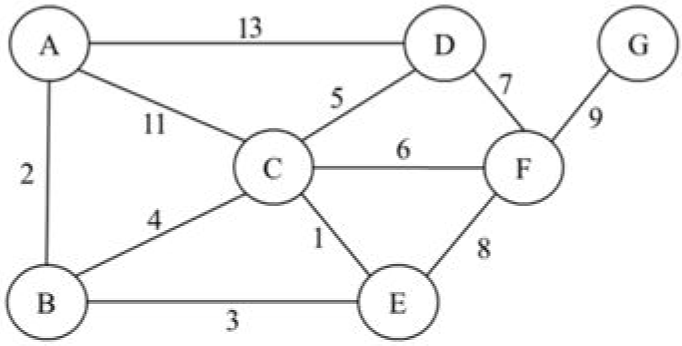
5) [3 pts] Use the hash function H(K)=K mod 7, insert these integer keys into the hash table: 23,16,38,49,41,32

0	1	2	3	4	5	6

6) [6 pts] Provide an example of an operation that has a growth rate corresponding to the given Big-O. Assume worst-case performance and don't provide duplicate answers.

Big-O Characterization	**Example**
O($logn$)	
O(1)	
O($nlogn$)	
O(2^n)	
O(n)	
O(n^2)	

7) Consider the following weighted graph.



a)[3 pts] Complete the sequence of edges in the MST in the order that Kruskal's algorithm includes them.

1	2							

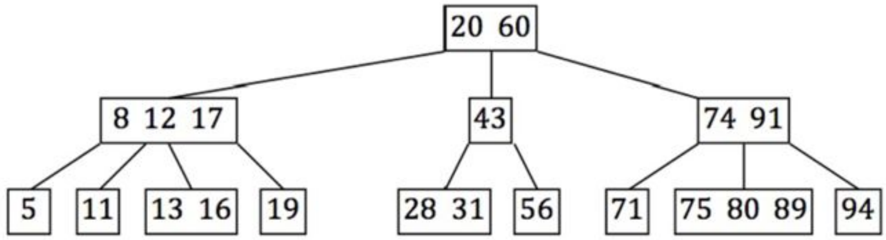
b) [3 pts] Complete the sequence of edges in the MST in the order that Prim's algorithm includes them. Start Prim's algorithm from vertex A.

2	3							

c) [4 pts] Run Dijkstra's algorithm on the weighted digraph above, starting at vertex A. List the vertices in the order in which the vertices are deleted. (for the first time) from the priority queue and give the length of the shortest path from A.

Vertex	A	B							
Distance	0	2							

8) [3 pts] Given a 2-3-4 tree as follows



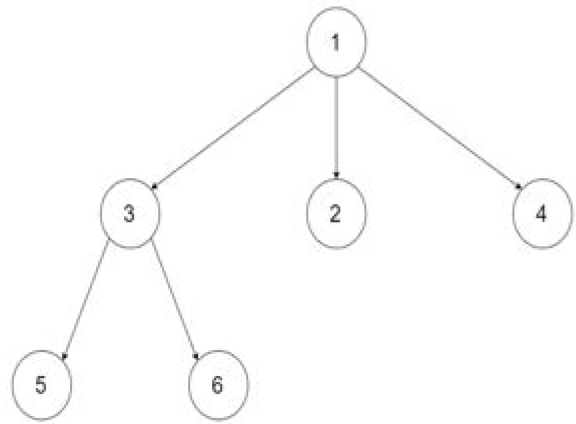
Draw the tree after deleting 56 from the original 2-3-4 tree

9)[3 pts] Construct a left-leaning red-black tree when the following elements are inserted in this order: 10, 20,30, 40, 50. Show your steps for each number. Use dashed line for red edge.

4.[25 pts] Programming

1) [8 pts] Given an n-ary tree, print the *preorder* traversal of its nodes' values.

```
// Definition for a Node.
class Node {
      public int val;
      public List<Node> children;
      public Node(int v,List<Node> c) {
            val = v;
            children = c;
      }
};
```



For example, given a 3-ary tree shown here, the preorder traversal is [1,3,5,6,2,4].

2) [8 pts] Given a sorted linked list, delete all duplicates such that each element appears only once. The Linked list node is defined as:

```
class ListNode{
int data;
ListNode next;
}
```

Example 1:	Input: 1->1->2	Output: 1->2
Example 2:	Input: 1->1->2->3->3	Output: 1->2->3

```
public ListNode removeDuplicates(ListNode head) {

}
```

3) [9 pts] Given two non-empty binary trees **s** and **t**, check whether tree **t** has exactly the same structure and node values with a subtree of **s**. A subtree of **s** is a tree consists of a node in **s** and all of this node's descendants. The tree **s** could also be considered as a subtree of itself. Node values are not unique.

```
class Node {
      int val;
      Node left,right
};
```

Example 1:

<table>
<tr>
<td>Given tree s:<pre>
       3
      / \
   4   5
  / \
 1   2
</pre></td>
<td>Given tree t:<pre>
   4
  / \
 1   2
</pre></td>
</tr>
</table>

Return **true**, because t has the same structure and node values with a subtree of s.

Example 2:

<table>
<tr>
<td>Given tree s:<pre>
       3
      / \
   4   5
  / \
 1   2
      /
   0
</pre></td>
<td>Given tree t:<pre>
   4
  / \
 1   2
</pre></td>
</tr>
</table>

Return **false**.

(Write your answer on the next page)

```
public boolean isSubTree(Node s, Node t){

}
```

END OF EXAM