Q1.

Data Structures Final Exam (30 points)

Q2. Student Name:

test

Q3. Student Net ID:

test

Q4. 1) Multiple Choice Questions (10 points):

Q5. 1.1) Assume that you implement a dictionary with a sorted array, what is the running-time in terms of big-O for the most efficient search algorithm possible to be used here to find an entry, in the worst case scenario? [1 point]

○ O(1)
○ O(log n)
○ O(n log n)
○ O(n)
○ O(n^2)

Q6. 1.2) Assume that you implement a dictionary with a sorted linked list, what is the running-time in terms of big-O for the most efficient search algorithm possible to be used here to find an entry, in the worst case scenario? [1 point]

○ O(1)
○ O(log n)
○ O(n log n)
○ O(n)
○ O(n^2)

Q7. 1.3) A root node in a multi-way search tree can have: (select all that apply) [1 point]

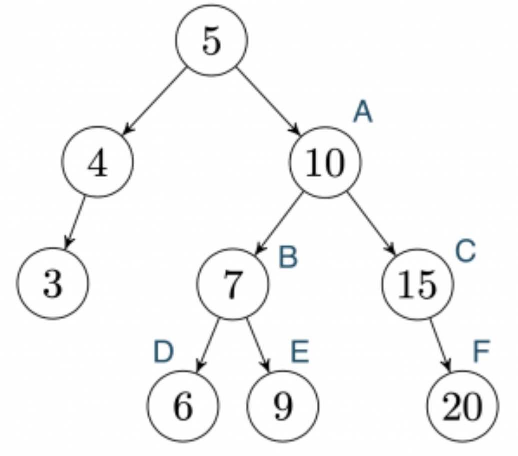
□ No entries

□ One entry
□ Two entries
□ Three entries
□ Any positive number of entries

Q8.
1.4) Which sorting algorithm is most efficient to use when the underlying system takes minimal time to copy values, but more to compare among them? [1 point]

○ Merge Sort
○ Quick Sort
○ Heap Sort
○ Insertion Sort

Q9. 1.5) Consider the binary search tree below, if we remove 'B', what descendant(s) may be used to replace it? (select all that apply) [1 point]



□ D
□ E
□ None of the above

Q10. 1.6) A complete binary tree fulfills the height-balance property at every node. True or False? [1 point]

TRUE FALSE
○ ○

Q11.
1.7) The performance of merge-sort algorithm may suffer with an input that is a sorted sequence of elements. True or False? [1 point]

TRUE FALSE

○ ○

Q12. 1.8) The order of which you insert entries into a heap does not impact the final resulted tree. True or False? [1 point]

TRUE FALSE
○ ○

Q13. 1.9) Bottom-up heap construction can be used to insert elements in O(n) time instead of O(n log n) time from a dynamic dataset. True or False? [1 point]

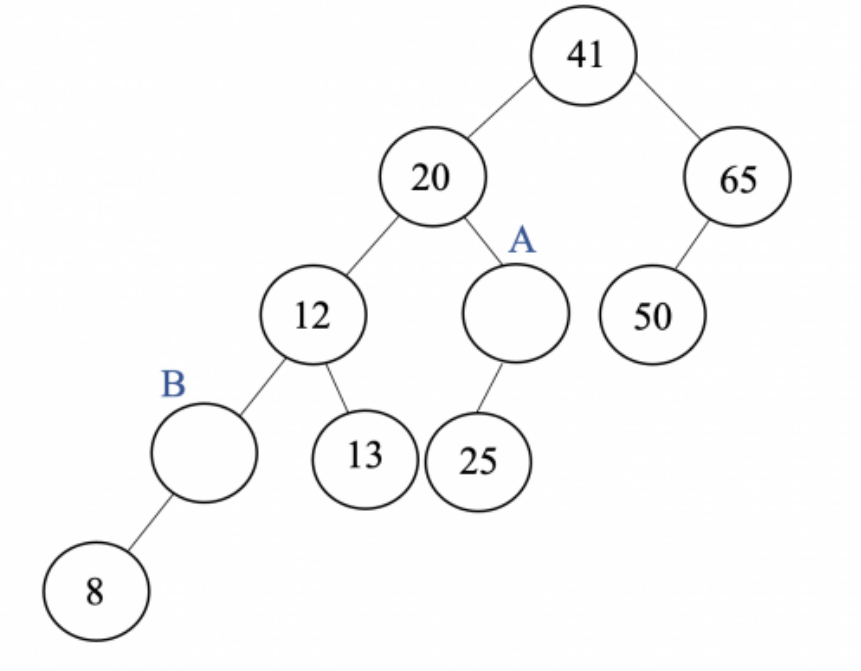
TRUE FALSE
○ ○

Q14. 1.10) Which of the following data structures you think is the most suitable fot the task of checking whether you have encountered a particular object before during a process or not? [1 point]

○ Stack
○ Queue
○ Binary Search Tree
○ Hashtable

Q15. 2) Short Answer Questions (15 points):

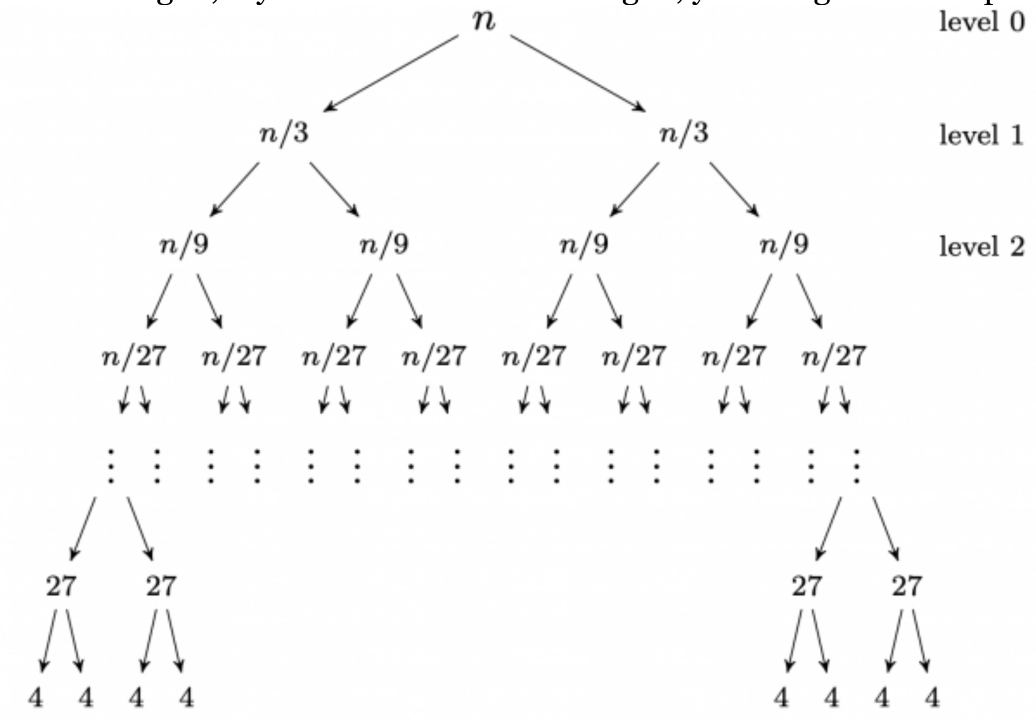
Q15.1. 2.1) [2 point] Consider the below BST and answer the following questions:



Q15.1.1. - What is the exact range of valid integer values possible for node A? [1 point]

Q15.1.2. - What is the exact range of valid integer values possible for node B? [1 point]

Q16. 2.2) [4 points] Consider the below tree and answer the following questions (not in terms of big-O; if you answer in terms of big-O, you will get half the point):



Q16.1.
- What is the input size per node in terms of n at any level m? [0.5 point]

Q16.2.
- What is the number of nodes at any level m? [0.5 point]

Q16.3.
- What is the total number of operations per recursive call in level m? [2 point]

Q16.4.
- What is the height of this tree? [1 point]

Q17. 2.3) [5 points] You have the following sequence of keys: [12, 8, 18, 28, 34, 10, 14, 32, 9, 40, 36, 38] to be inserted sequentially into an empty Red-Black tree. Answer the following questions:

Q17.1. - How many double red violations did you encounter and fix while building the tree? [2 point]

Q17.2. - How many trinode restructuring operations did you need to apply to fix double red violations? Specify the types of the trinode restructuring operations and after the insertion of which keys. [1.5 point]

Q17.3. - How many recoloring operations did you need to apply to fix double red violations? Specify after the insertion of which keys. [1.5 point]

Q18. 2.4) [4 points] Assume that you have a min-heap of 4 nodes, containing entries with keys: 6, 12, 14, and 18. Answer the following questions:

Q18.1. - List the structure of every possible min-heap that could match this description. For each structure, list the keys in each level (starting from level zero) in order from the left to the right. [2 point]

Q18.2. - For each of the structures you listed in the previous question, show what happens with removeMin( ) applied 4 times (describe the heap after each removeMin operation in terms of the keys in each level (starting from level zero) in order from the left to the right). [2 point]

Q21. 3) Algorithm Analysis (5 points):

Q22. 3.1) [2 points] Consider the following C++ implementation to answer the next subquestions:

```
void insertElement(int new_element, int list_length, vector<int> &vect){
   int indx = 0;
   bool flag;
   bool flag = false;

   while ((indx < list_length) && !flag){
      if (new_element > vect[indx]){
         indx++;}
      else {
         flag = true; } }

   for (int i=list_length; i>indx; i--){
      vect[i] = vect[i-1]; }

   vect[indx] = new_element; }
}
```

Q22.1. - What does this function do? [1 point]

Q22.2. - What is the running time of the best case scenario in terms of big-O? [0.5 point]

Q22.3. - What is the running time of the worst case scenario in terms of big-O? [0.5 point]

Q22.4. - How can we improve the running time of this function in the worst case scenario? [1 bonus point]

Q23.

3.2) [3 points] Consider the following C++ implementation to answer the next subquestions:

```
int func(int n) {
   if (n == 2) {
      return 2;}
   else {
      return n/log(n)+ 2*func(n/2);}
}
```

Q23.1. - What is the base case? [0.5 point]

Q23.2. - What is the recurrence case? [0.5 point]

Q23.3. - What is the running time of func(n) in the worst case scenario in terms of big-O? [2 point]

Q23.

No More Questions