**FINAL EXAM**

* **Due** No due date

* **Points** 80

* **Questions** 40

* **Available** after May 17 at 7pm
* **Time Limit** None

Attempt History

|  | **Attempt** | **Time** | **Score** |
| --- | --- | --- | --- |
| **LATEST** | [Attempt 1](https://dvc.instructure.com/courses/27283/quizzes/50378/history?version=1) | 72 minutes | 76 out of 80 |

Score for this quiz: **76** out of 80

Submitted May 17 at 8:12pm

This attempt took 72 minutes.

**Question 1**

**2 / 2 pts**

**Draw the binary search tree containing the following 11 values in the order shown, and answer the**

                              15  7  9  21  44  30  33  29  10  1  17 (in that order).

Which is the in-order successor of 15?

**Correct!**



17



10



9



7

**Question 2**

**2 / 2 pts**

**Draw the binary search tree containing the following 11 values in the order shown, and answer the.**

                              15  7  9  21  44  30  33  29  10  1  17 (in that order).

**Which is the in-order predecessor of 44?**



21



30



29

**Correct!**



33

**Question 3**

**2 / 2 pts**

**Draw the binary search tree containing the following 11 values in the order shown, and answer the**

                              15  7  9  21  44  30  33  29  10  1  17 (in that order).

How many levels are there in the tree



4



6



3

**Correct!**



5

**Question 4**

**2 / 2 pts**

**Examine the following Print function and determine which traversal it is.**

                               void Print(TreeNode<ItemType>\* tree);

                              {

                                    if (tree != NULL)

                                    {

                                       Print(tree->left);

                                       tree->info.PrintItem();

                                       Print(tree->right);

                                    }

                  }

**Correct!**



inorder



by level



preorder



postorder

**Question 5**

**2 / 2 pts**

**Examine the following binary search tree and answer the question. The numbers on the nodes are labels so that we can talk about the nodes; they are not values in the key members of the items in the tree.**

**If an item is to be inserted into the tree whose key data member is less than the key data member in node 1 but greater than the key data member in node 5, where would it be inserted?**

**Correct!**



node 5's right child



node 4's right child



node 7's left child



node 7's lright child

**Question 6**

**2 / 2 pts**

**When a node with no children is deleted, what replaces the pointer to the deleted node?**



the node's left subtree



the node's rigth subtree



the child

**Correct!**



NULL

**Question 7**

**2 / 2 pts**

**When a node with only one child is deleted, what replaces the pointer to the deleted node?**



the node's left subtree

**Correct!**



the child



NULL



the node's right subtree

**Question 8**

**2 / 2 pts**

**When a value in a node with two children is deleted, what replaces the deleted value?**



Randomize all values and replace the node with the mean value



It is illegal in C++ top delete a node that has two children



the with the value of the node's predecessor

**Correct!**



with the left most child on its right subtree, or the right most child on its left subtree.

**Question 9**

**2 / 2 pts**

**In a binary tree stored in an array called treenodes, the left child of treenodes[index] is located in**

**Correct!**



treenodes[index + index + 1]



treenodes[index + index]



There is no enough information to answer this question



treenodes[index + index + 2]

**Question 10**

**2 / 2 pts**

**In a binary tree stored in an array, the parent node of treenodes[index] is located in**

**Correct!**



treenodes[(index-1) / 2]



treenodes[(index + index) / 2]



No one knows



treenodes[index + index /2]

**Question 11**

**2 / 2 pts**

**Is the tree bellow** a binary search tree?



it is a binary tree in C language but not in c++ language



yes



depends on the compiler

**Correct!**



no

**Question 12**

**2 / 2 pts**

**In a max heap, ReheapDown restores the heap property after**

**Correct!**



deleting a node



displaying the tree contents



inserting a node



counting nodes

**Question 13**

**2 / 2 pts**

**When the values in the heap are moved into the array, which sequence below is correct (the first value is in [0]).**



There is no unique mapping from the heap into the array representation.



5, 12, 15, 20, 35, 35, 36, 40

**Correct!**



40, 35, 36, 20, 15, 12, 35, 5



5, 20, 15, 12, 35, 35, 36, 40

**Question 14**

**2 / 2 pts**

**If 21 is inserted into the heap, is ReheapUp necessary?**



no



depends on the compiler

**Correct!**



yes



depends on Dijkstra's algorith

**Question 15**

**2 / 2 pts**

**If 40 is removed, what value will replace it after ReheapDown function is called?**



5

**Correct!**



36



35



depends on the compiler

**Question 16**

**2 / 2 pts**

The binary search tree shown below was constructed by inserting a sequence of items  
into an empty tree. Which of the following input sequences will NOT produce this binary search tree?

Which of the following input sequences will not produce this binary search tree?



5 3 4 9 12 7 8 6 20



5 9 3 7 6 8 4 12 20

**Correct!**



5 9 3 6 7 8 4 12 20



5 9 7 8 6 12 20 3 4



5 9 7 3 8 12 6 4 20

**Question 17**

**2 / 2 pts**

The following method/function sorts an array of floating-point numbers. What is the big O of the  algorithm ?:

void sort(double a[], int SIZE) {  
        int i, j, n;  
       double temp;  
       n = SIZE;  
        for (i = 1; i < n; i++) {  
             temp = a[i];  
              j = i;  
              while (j >= 1 && a[j-1] > temp) {  
                    a[j] = a[j-1];  
                     j--;  
             }  
             a[j] = temp;  
            }  
          }

**Correct!**



O(n^2)



O(n\*log(n))



O(log(n))



O(n)

**Question 18**

**2 / 2 pts**

Here is an array that has just been partitioned by the first step of quicksort:  
3 0 2 4 5 8 7 6 9  
Which of the following statements is correct?



7 could be the pivot, but 5 could not be.



Either 5 or 7 could be the pivot.

**Correct!**



5 could be the pivot, but 7 could not be.



Neither 5 nor 7 could be the pivot.

**Question 19**

**2 / 2 pts**

You are given an empty hash table of size 7 that uses closed hashing. The following  
sequence of keys is to be inserted:  
15  17  8  23  3  5

You insert these keys using linear probing and the following hash function:

h(x) = x % 7;

What would the contents of the hash table contain after inserting the above values.



|  |
| --- |
| index: 0       1       2       3       4       5       6 |
| key   :15      8       17    23      3       5 |



|  |
| --- |
| index: 0       1       2       3       4       5       6 |
| key   : 8      17       5       23      3      23 |

**Correct!**



|  |
| --- |
| index: 0       1       2       3       4       5       6 |
| key   :         15      8       17    23      3       5 |



|  |
| --- |
| index: 0       1       2       3       4       5       6 |
| key   :         8      17       5       23      3      23 |

**Question 20**

**2 / 2 pts**

You are given an empty hash table of size 7 that uses closed hashing. The following  
sequence of keys is to be inserted:  
15  17  8  23  3  5

You insert these keys using **quadratic** probing and the following hash function:

h(x) = x % 7;

What would the contents of the hash table contain after inserting the above values.



|  |
| --- |
| index:  0       1       2       3       4       5       6 |
| key   :  23     15     8       17     3       5 |

**Correct!**



|  |
| --- |
| index:  0       1       2       3       4       5       6 |
| key   :          15      8       17     3       5     23 |



|  |
| --- |
| index:  0       1       2       3       4       5       6 |
| key   :  15      8               17     3       5     23 |



|  |
| --- |
| index:  0       1       2       3       4       5       6 |
| key   :  17        15      8            3       5     23 |

**Question 21**

**2 / 2 pts**

For a well-designed hash table, searching requires \_\_\_\_\_ on average.



O(n\*logN)



O(logN)



)(N)

**Correct!**



O(1)

**Question 22**

**2 / 2 pts**

Perform a depth-first traversal of the graph shown below, starting with vertex C.  
Select the smallest edge first when appropriate. Which of the listed sequences is the resulting path

**Correct!**



CBADFE



CFBEBDA



CEBDAF



CBDABF

**Question 23**

**0 / 2 pts**

Perform a breath-first traversal of the graph shown below, starting with vertex C.  
Select the smallest edge first when appropriate. Which of the listed sequences is the resulting path

**You Answered**



CBFEADE



CFEBAD

**Correct Answer**



CBEFAD



CFEBDA

**Question 24**

**2 / 2 pts**

The array below is to be sorted in ascending order.

17 53 71 62 36 46 41 23 12

After the**initial partition** step of the version of quicksort , with  
36 as the pivot, how would the array be ordered?



36 12 23 17 62 46 41 71 53



12 17 23 36 62 46 41 71 53



36 12 23 62 17 46 41 71 53

**Correct!**



17 12 23 36 62 46 41 71 53

**Question 25**

**2 / 2 pts**

The array below is to be sorted in ascending order.

17 53 71 62 36 46 41 23 12

After the initial iteration of bubble sort, how would the array be ordered?



7 36 62 41 46 41 23 12 17



17 53 62 46 36 41 23 12 71

**Correct!**



17 53 62 36 46 41 23 12 71



17 53 62 36 46 41 12 23 71

**Question 26**

**2 / 2 pts**

Reference the graph above . Step through Dijkstra’s algorithm to calculate the single-source shortest paths from A to every other vertex.

What is  the lowest-cast path from node A to node F

**Correct!**



: A to B to C to E to F



: A to F to C to E to B



: A to C to B to E to F

**Question 27**

**2 / 2 pts**

T/F Quicksort is claimed to have an expected running time of O(n log n), but it could  
be as slow as O(n^2) time if the partition function always picks as the pivot the largest or  
smallest element of the array section being sorted. That will cause the depth of the  
recursion to be O(n) instead of O(log n).

**Correct!**



True



False

**Question 28**

**2 / 2 pts**

T/F To  fix Quicksort so the expected time is O(n log n), one should use partition algorithm that partitions the array section into two equal-sized halves; one way to do this is to pick the pivot element randomly; another is to use an algorithm that estimates the median value in the array by, say, picking the median of the first, last, and a few intermediate elements of the array section. This will limit the depth of the recursive calls to O(log n) and, since each level does O(n) work, the total time will be O(n log n).

**Correct!**



True



False

**Question 29**

**2 / 2 pts**

**T/F Merge sort requires extra space.**

**Correct!**



True



False

**Question 30**

**2 / 2 pts**

**Heap sort requires extra space**



True

**Correct!**



False

**Question 31**

**2 / 2 pts**

**Which of the following lists the graph nodes in priority queue order (lowest value is highest priority)?**



F, C, D, B, G, A, E



A, B, D, E, F, G

**Correct!**



B, C, F, D, A, E, G

**Question 32**

**2 / 2 pts**

**The shortest distance from D to B is:**



15



4



2

**Correct!**



10

**Question 33**

**2 / 2 pts**

The graph is pictured in which of the following implementations?



Adjacency List



List of Edjes



Synchronized shortest path list

**Correct!**



Adjacency Matrix

**Question 34**

**2 / 2 pts**

T/F A full binary tree has all the leaf nodes on the same level, and every non-leaf node has two children.

**Correct!**



True



False

**Question 35**

**0 / 2 pts**

A heap must be a full binary tree

**You Answered**



True

**Correct Answer**



False

**Question 36**

**2 / 2 pts**

If a priority queue is being implemented using a linked list of values sorted by priority (in descending order), what is the Big-O complexity of the Dequeue operation?



O(N\*N)



O(logN)

**Correct!**



O(1)



O(N)

**Question 37**

**2 / 2 pts**

The NlogN sort that you should NOT use when the data is almost sorted.



Merge Sort



Bubble Sort



Insertion Sort

**Correct!**



Quick Sort

**Question 38**

**2 / 2 pts**

The NlogN sort that you should not use if memory is scarce.



Shell Sort

**Correct!**



Merge Sort



Heap Sort



Quick Sort

**Question 39**

**2 / 2 pts**

T/F In the HeapSort algorithm, the Big-O complexity of the best case, average case, and worst case

is  :  O(NlogN),   O(NlogN),   O(NlogN) respectively.

**Correct!**



True



False

**Question 40**

**2 / 2 pts**

What is the order in which vertices are visited by Dijkstra when the source vertex is A.



Cannot tell from the information given



A D C B E F G

**Correct!**



A C B E F D G



A D F G

Quiz Score: **76** out of 80