

CSI2110 Fall 2017

Final Exam

Prof. Lucia Moura and Prof. Robert Laganriere

December 14, 9 :30h – 3 hours

40 points (40% of the final mark)

LAST NAME: _____

First Name: _____

Student Number: _____

Signature _____

Instructions :

Closed book exam. No calculators allowed. Please fill all multiple choice questions on scantron paper. There are 15 pages in this exam.

In all the questions, when asked for a big-Oh, please provide the best possible value.

All the logarithms are given in base 2.

QUESTION	MARKS OBTAINED
Questions 1-15 (multiple choice)	/15
Question 16 (AVL tree)	/2
Question 17 (AVL tree)	/2
Question 18 (BFS, DFS, graph repr)	/6
Question 19 (Dijkstra)	/4
Question 20 (Sorting)	/2
Question 21 (Mergesort)	/3
Question 22 (Trees)	/6
TOTAL	/40

Question 1 [1 point] What is the big-oh characterization of the following function?

$$f(N) = 1 + 2 + 4 + 8 + \dots + 2^{N-1} + 2^N$$

- A) $O(N^2)$ B) $O(\log N)$ C) $O(N)$ D) $O(2^N)$ E) $O(N^N)$

Question 2 [1 point] If a function $f(N)$ is $O(N)$ and a second function $g(N)$ is $O(N^2)$ what would be the big-Oh of the function $f(N)+g(N)$?

- A) $O(N)$ B) $O(N^2)$ C) $O(N^3)$ D) $O(\log N)$ E) Impossible to determine.

Question 3 [1 point] The following function returns the index of the two numbers having the maximum product value. What is the big-oh complexity of this algorithm in terms of the number of multiplication operations ?

```
1.  int x = -1; int y = -1;
2.  int maxProd = -1;
3.  for (i=0; i<n-1; i++)
4.      for (j=i+1; j<n; j++) {
5.          int prod = tab[i] * tab[j]);
6.          if (prod > maxProd) {
7.              maxProd = prod;
8.              x=i;
9.              y=j;
10.         }
11. return x, y;
```

- A) $O(N^2)$ B) $O(\log N)$ C) $O(N)$ D) $O(2^N)$ E) $O(N^N)$

Question 4 [1 point] Draw the binary tree where each node contains a letter and its inorder traversal prints ZLCXE and its postorder traversal prints ZCLEX:

Which one is the result of a preorder traversal of the tree you just draw?

- A) XLEZC B) XELCZ C) LEXCZ D) ZLCXE E) XLZCE

Question 5 [1 point]

Consider the **max heap** that uses the following array $a[] = [10, 5, 3, 1, 4, 2]$ to store its elements. Which is the array after one `removeMax()` operation?

- A) [5, 3, 1, 4, 2, -]
B) [2, 5, 3, 1, 4, -]
C) [5, 4, 3, 1, 2, -]
D) [5, 2, 3, 1, 4, -]
E) [5, 4, 3, 2, 1, -]

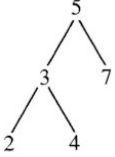

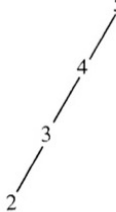
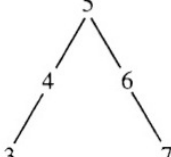
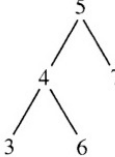
Question 6 [1 point]

Consider the **max heap** that uses the following array $a[] = [10, 5, 3, 1, 4, 2]$ to store its elements. Which is the array after inserting key 9 into the heap?

- A) [10, 5, 3, 1, 4, 2, 9]
B) [10, 9, 5, 1, 4, 2, 3]
C) [10, 9, 1, 4, 5, 2, 3]
D) [10, 5, 9, 1, 4, 2, 3]
E) none of the above

Question 7 [1 point]

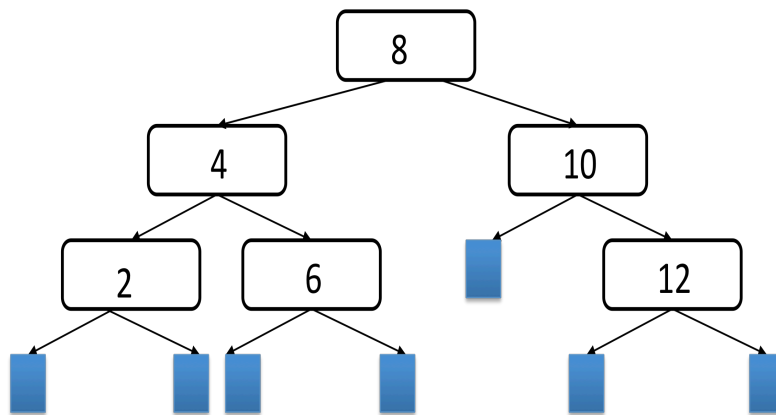
Which ones of the following is a **binary search tree** ? (Here no dummy leaves are used)

I	II	III	IV	V
				

- A) only I, II, III, IV
- B) only I, II, IV, V
- C) only I, II, V
- D) only I and V
- E) all of the trees are binary search trees

Question 8 [1 point]

Which of the following keys when inserted into the following AVL tree will cause a rebalancing operation?



- A) Insert 1
- B) Insert 7
- C) Insert 9
- D) Insert 11
- E) None of the above

Question 9 [1 point] Consider the following hash function $h(k) = k \bmod 9$, and a hash table of size 9, where collisions are resolved using **linear probing**. What is the table resulting from the insertions (in the following order) into an initially empty table: 11, 2, 3, 7, 13, 12, 9

- A)

Index:	0	1	2	3	4	5	6	7	8
Hash Table	11	2	3	7	13	12	9		
- B)

Index:	0	1	2	3	4	5	6	7	8
Hash Table	9		2	3	11	13	12	7	
- C)

Index:	0	1	2	3	4	5	6	7	8
Hash Table	9		11	2	3	12	13	7	
- D)

Index:	0	1	2	3	4	5	6	7	8
Hash Table	9		11	2	3	13	12	7	
- E)

Index:	0	1	2	3	4	5	6	7	8
Hash Table	9	11	2	3	13	12		7	

Question 10 [1 point] Consider the following hash function $h(k) = k \bmod 9$, and a hash table of size 9.

Index:	0	1	2	3	4	5	6	7	8
Hash Table			11	3	13	5		7	

In which index of the hash table we must insert key $k=12$ if collisions are resolved using **quadratic probing**? Remember that in quadratic probing we attempt to enter keys into positions: $h_j(k) = (h(k) + j^2) \bmod 9$, for $j=0,1,2,\dots$

- A) Index 0
 B) Index 1
 C) Index 6
 D) Index 8
 E) It will be impossible to insert key $k=12$.

Question 11 [1 point] Select the answer that gives the most precise estimate of the **worst-case time complexity** of Mergesort, Heapsort and Quicksort, respectively.

	Mergesort	Heapsort	Quicksort
A)	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
B)	$O(n \log n)$	$O(\log n)$	$O(n \log n)$
C)	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
D)	$O(n^2)$	$O(n \log n)$	$O(n \log n)$
E)	none of the above		

Question 12 [1 point] Suppose you have an array of n elements sorted in **increasing** order. You now use a sorting algorithm to sort them in **increasing** order (the algorithm does not know the input is in increasing order and runs as usual)

This question is about the most precise big-Oh complexity for the problem above using:

- Quicksort (with pivot in partition being the first element of the sequence)
- Insertion sort
- Selection sort

	Quicksort (pivot is first)	Insertion sort	Selection sort
A)	$O(n \log n)$	$O(n)$	$O(n)$
B)	$O(n^2)$	$O(n)$	$O(n^2)$
C)	$O(n \log n)$	$O(n^2)$	$O(n^2)$
D)	$O(n^2)$	$O(n^2)$	$O(n^2)$
E)	none of the above		

Question 13 [1 point] Consider a class of graphs, called Ring, with n vertices where each vertex is adjacent to exactly 2 other vertices. Consider a graph traversal with time complexity $\Theta(n+m)$ where n is the number of vertices and m is the number of edges in the graph. What is the time complexity of the graph traversal on the given type of graph as a function of n ?

- A) $\Theta(n)$
- B) $\Theta(n \log n)$
- C) $\Theta(n^2)$
- D) $\Theta(n^3)$
- E) None of the above.

Question 14 [1 point] Consider procedure `stableSort(A,i)` which sorts a global array of strings `W` with 4 letters by simply using the letter at position `i` of each string `S` as the key for sorting in increasing order of `S[i]`. Moreover we know that the algorithm is **stable**. Consider the piece of code:

```
for i=0 to 3 do stableSort(A,i)
```

If before the piece of code the array has the Strings in the following order

`W=[DEBY, ANNA, ANDY, BOBI, MEBA, ANGU, IKBA]`

Then after the code is run the array contains:

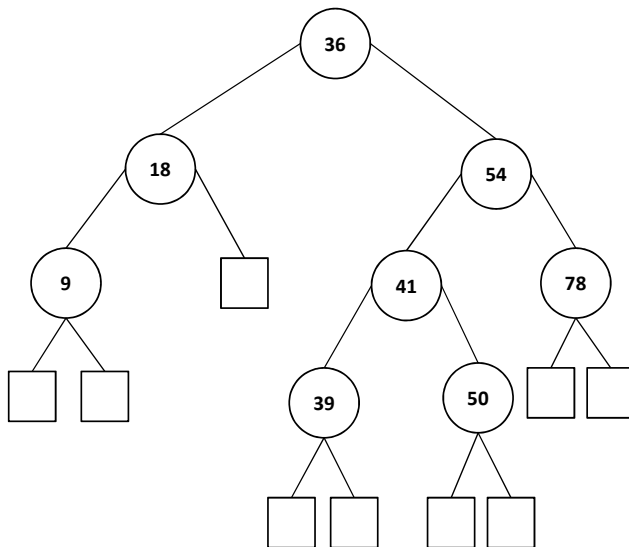
- A) `W=[DEBY, ANNA, ANDY, BOBI, MEBA, ANGU, IKBA]`
- B) `W=[ANNA, ANDY, ANGU, DEBY, IKBA, BOBI, MEBA]`
- C) `W=[ANDY, ANGU, ANNA, DEBY, IKBA, BOBI, MEBA]`
- D) `W=[MEBA, IKBA, ANNA, BOBI, ANGU, DEBY, ANDY]`
- E) None of the above.

Question 15 [1 point] All the alternatives are data structures that can be used to implement the MAP Abstract Data Type, with the exception of

- A) Hash table
- B) Binary Search Tree
- C) AVL tree
- D) Array list
- E) Double ended queue (Deque)

Question 16 [2 point]

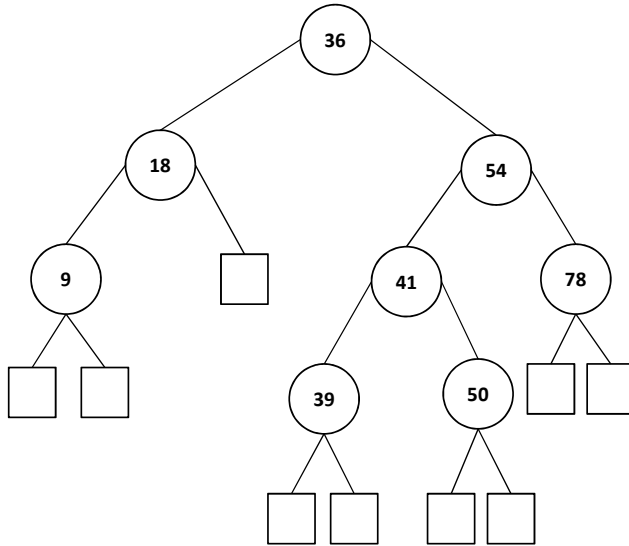
Consider the following AVL tree.



Insert key 38 using the algorithm seen in class and show the resulting tree :

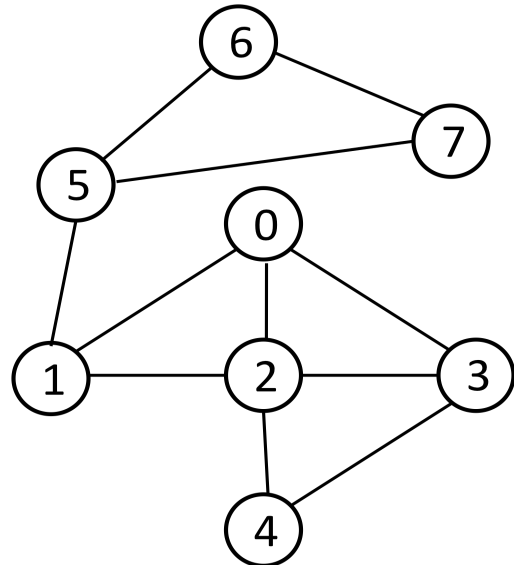
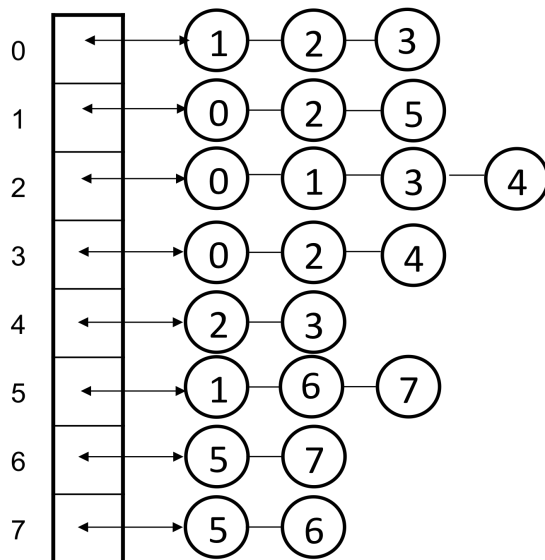
Question 17 [2 point]

Consider the following AVL tree.



Delete key 9 using the algorithm seen in class and show the resulting tree :

Question 18 [6 points] Consider the graph given below and consider the given adjacency list representation of the graph.



Taking into account the order of the vertices in each adjacency list, and **starting the search from vertex 1**,

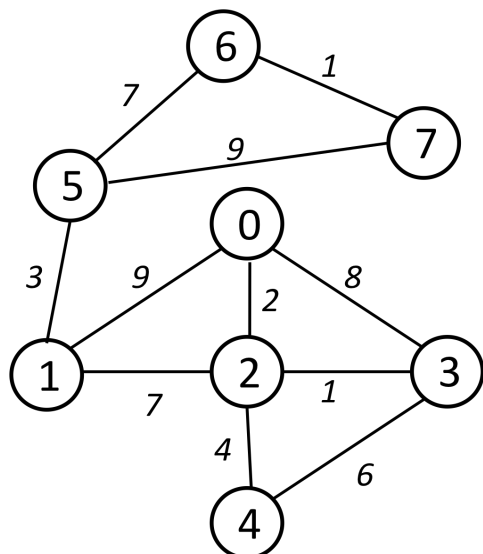
1.) give, in order, the list of visited nodes in the case of a **depth-first search** starting from vertex 1:

2.) give, in order, the list of visited nodes in the case of a **breath-first search** starting from vertex 1:

3.) Below draw the **adjacency matrix** representation of the graph :

Question 19 [4 points]

Use Dijkstra's algorithm to obtain a tree of shortest paths starting for starting vertex 3 for the graph below.



Fill the following table to show the updates of the distances associated to the vertices along its execution. We already filled the iteration when 3 is added to the cloud. The next vertex to join the cloud is vertex 2 and at this time edge (3,2) is added to the tree of shortest paths. You need to keep filling the table with distance updates until all vertices are added to the cloud. On the right column you show edges that join the tree of shortest paths at each iteration.

iteration	vertex joining the cloud	Dist[0]	Dist[1]	Dist[2]	Dist[3]	Dist[4]	Dist[5]	Dist[6]	Dist[7]	Edge Added to the tree of shortest paths
0	3	8	infinity	1	0	6	infinity	infinity	infinity	--
1	2									(3,2)
2										
3										
4										
5										
6										
7										

Question 20 [2 points]

A developer tries to implement a sorting algorithm but unfortunately it is **incorrect**:

```
1. public static void selection(int[] data){
2.     for (int i = 0; i < data.length - 1; i++) {
3.         int minIndex = i;
4.         for (int j = 0; j < data.length; j++) {
5.             if (data[j] < data[minIndex])
6.                 minIndex = j;
7.         }
8.         // swap data between i and minIndex
9.         int temp = data[i];
10.        data[i] = data[minIndex];
11.        data[minIndex] = temp;
12.    }
13. }
```

When the algorithm is tested with the following array:

10, 34, 2, 56, 7, 67, 88, 42

the following result is obtained:

34, 10, 56, 7, 67, 88, 2, 42

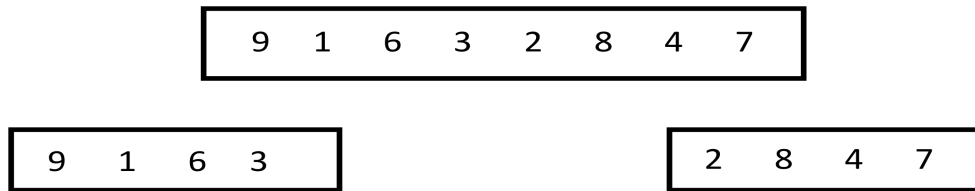
Correct the code by proposing a change to one and only one line of code.

In the space below give the line number and the modified line:

Question 21 [3 points] Mergesort

Draw the steps for a full execution of the Mergesort algorithm for the given sequence of numbers. You can use a similar diagram as you used in assignment 5 where the partition steps are shown first followed by the merge steps.

(Remark: Only the first partition step is shown.)



Question 22 [6 points] Consider a binary search tree and the piece of code below. For a node p in the binary search tree, $\text{left}(p)$ is the node that is the left child of p and $\text{right}(p)$ is the node that is the right child of p . The function $\text{compare}(x,y)$ is a standard comparator method that returns 0 if $x=y$, -1 if $x<y$, and 1 if $x>y$.

```
int private
Position<Entry<K,V>> Mystery(Position<Entry<K,V>> p, K key) {
    if (isExternal(p))
        return p; // case that algorithm Mystery is not successful
    int comp = compare(key, p.getElement());
    if (comp == 0)
        return p; // case that algorithm Mystery is successful
    else if (comp < 0)
        return Mystery(left(p), key);
    else
        return Mystery(right(p), key);
}
```

Part A) What does method $\text{Mystery}(p, \text{key})$ do?

Part B) For a call $\text{Mystery}(\text{root}, \text{key})$, where root is the root of a tree containing n nodes, what is the worst-case time complexity (big-Oh) of the algorithm in the following cases

The tree is an arbitrary binary search tree: _____

The tree is an AVL tree: _____

(page intentionally left blank)