	King Saud University	College of Computer and Information Sciences Department of Computer Science
	Data Structures CSC 212	**Final Exam - Fall 2020**
	Date: 28/12/2020	Duration: 3 hours

Question 1 ..... 12 points

Choose the most appropriate data structure for each of the following tasks.

A. LinkedList	**B.** ArrayList	**C.** DoubleLinkedList	**D.** LinkedQueue	**E.** AVL	**F.** Stack
G. BT.	**H.** BST.	**I.** LinkedPQueue.	**J.** BPlusTree.	**K.** HeapPQueue.	**L.** Graph.

1. An application that analyzes the communication patterns on a online social network. Graph
2. An application in a restaurant that keeps track of all tables assigned to every waiter. Linked Queue
3. An algorithm that decides on the order of surgical procedures at a hospital. Linked Priority Queue
4. An application that keeps track of people that have been vaccinated and those who have not. B Plus Tree
5. A video play list that allows forward and backward navigation between the videos. Double Linked List
6. An algorithm that receives a character and returns its ASCII code. AVL

Question 2 ..... 12 points

Write the method **public static int ls(List<Boolean> l)** which takes as input a non-empty list `l` of Booleans and returns the length of the longest contiguous sequence of **true** in `l`.

Example 1. *If* $l = \{0,1,1,0,0,1,1,1,1,0,1\}$, *ls(l) returns 4.*

```
1  public static int ls(List<Boolean> l) {
2    int maxL = ...;
3    ...
4    while (true) {
5      ...
6      while (true) {
7        if (...)
8          ...
9        else
10         ...
11       if (...)
12         break;
13       else
14         ...
15     }
16     if (...)
17       ...
18     if (...)
19       break;
20     else
21       ...
22   }
23   return maxL;
24 }
```

1. Line 2:
   - (A) `int maxL = -1;`
   - (B) `int maxL = 1;`
   - (C) `int maxL = l.retrieve();`
   - (D) `int maxL = 0;`
   - (E) None
2. Line 3:
   - (A) `l.insert(true);`
   - (B) `l.update(false);`
   - (C) `l.findNext();`
   - (D) `maxL = l.length();`
   - (E) None L.findFirst

3. Line 5:
   - (A) `int cpt = 1;`
   - (B) `int cpt = -1;`
   - (C) `int cpt = 0;`
   - (D) `int cpt = l.retrieve();`
   - (E) None

4. Line 7:
   - (A) `if (l.retrieve()> cpt)`
   - (B) `if (l.retrieve()> maxL)`
   - (C) `if (l.retrieve())`
   - (D) `if (l.last())`
   - (E) None

5. Line 8:
   - (A) `maxL++;`
   - (B) `cpt--;`
   - (C) `cpt++;`
   - (D) `l.findFirst();`
   - (E) None

6. Line 10:
   - (A) `continue;`
   - (B) `break;`
   - (C) `l.findFirst();`
   - (D) `l.finNext();`
   - (E) None

7. Line 11:
   - (A) `if (l.retrieve())`
   - (B) `if (l.last())`
   - (C) `if (maxL == 0)`
   - (D) `if (cpt == 0)`
   - (E) None

8. Line 14:
   - (A) `maxL = 0;`
   - (B) `maxL = cpt;`
   - (C) `l.findFirst();`
   - (D) `l.findNext();`
   - (E) None

9. Line 16:
   - (A) `if (cpt == maxL)`
   - (B) `if (cpt < maxL)`
   - (C) `if (cpt > maxL)`
   - (D) `if (cpt == 0)`
   - (E) None

10. Line 17:
    - (A) `cpt = maxL;`
    - (B) `maxL++;`
    - (C) `maxL = cpt;`
    - (D) `cpt++;`
    - (E) None

11. Line 18:
    - (A) `if (maxL < 0)`
    - (B) `if (l.last())`
    - (C) `if (maxL > 0)`
    - (D) `if (l.retrieve())`
    - (E) None

12. Line 21:
    - (A) `l.findFirst();`
    - (B) `l.findNext();`
    - (C) `cpt++;`
    - (D) `maxL++;`
    - (E) None

Question 3 ..... 14 points

(a) Write a constructor for the class `BT` which builds the tree from its array representation `a`. The root element is at position 1, and the left and right children of a node at position `i` are located at positions `2*i` and `2*i+1` respectively. If a position contains **null**, then the corresponding node does not exist.

```
1  public BT(T[] a) {
2      current = root = ...;
3  }
4  private BTNode<T> rfa(T[] a, int i) {
5      if (...)
6          return ...;
7      BTNode<T> nnode = ...;
8      ...
9      ...
10     ...
11 }
```

1. Line 2:
   - (A) `rfa(a, 0)`
   - (B) `null`
   - (C) `rfa;`
   - — (D) `rfa(a, 1)`
   - (E) None
2. Line 5:
   - (A) `a[i] < 0`
   - — (B) `i >= a.length || a[i] == null`
   - (C) `i < a.length && a[i] < 0`
   - (D) `a[i] != null && i > a.length`
   - (E) None
3. Line 6:
   - (A) `current;`
   - — (B) `null`
   - (C) `i`
   - (D) `a[i/2]`
   - (E) None
4. Line 7:
   - (A) `root;`
   - (B) `a[i];`
   - (C) `new BTNode<T>(i)`
   - — (D) `new BTNode<T>( a[i] )`
   - (E) None
5. Line 8:
   - (A) `current = current->left;`
   - (B) `current.left = rfa(a, i);`
   - — (C) `nnode.left = rfa(a, i*2);`
   - (D) `current.right = rfa(a,i*2);`
   - (E) None
6. Line 9:
   - (A) `current = current->right;`
   - — (B) `nnode.right = rfa(a,i*2+1);`
   - (C) `current.right = rfa(a, i*2+1);`
   - (D) `current.left = rfa(a, i*2+1);`
   - (E) None
7. Line 10:
   - (A) `return current;`
   - (B) `return current.rfa();`
   - (C) `return null;`
   - — (D) `return nnode;`
   - (E) None

(b) Repeat the same question as user of BT.

```
1  public static <T> BT<T> fa(T[] a) {
2    BT<T> bt = new BT<T>();
3    if (...) {
4      ...
5      ...
6    }
7    return bt;
8  }
9  private static <T> void rfa(BT<T> bt, T[] a,
       int i) {
10   if (...) {
11     ...
12     ...
13     ...
14   }
15   if (...) {
16     ...
17     ...
18     ...
19   }
20 }
```

1. Line 3:
   - — (A) `if (a.length > 1 && a[1] != null){`
   - (B) `if (a.length > 0 || a[0] != null){`
   - (C) `if (a.length > 0 && a[0] != null){`
   - (D) `if (a.length > 1){`

(E) None

2. Line 4:

(A) `bt.insert(a[0], relative.Root);`

(B) `rfa(bt, a, 1);`

(C) `bt.insert(a[ar.length/2], relative.Root);`

(D) `bt.insert(a[1], relative.Root);`

(E) None

3. Line 5:

(A) `rfa(bt, a, ar.length/2);`

(B) `rfa(bt, a, 1);`

(C) `rfa(bt, a, 0);`

(D) `bt.insert(a[1], relative.Parent);`

(E) None

4. Line 10:

(A) `if (a[2*i] != null){`

(B) `if (2*i < a.length && a[2*i] != null){`

(C) `if (2*i < a.length){`

(D) `if (2*i < a.length || a[2*i] != null){`

(E) None

5. Line 11:

(A) `bt.insert(a[2*i], Relative.LeftChild);`

(B) `bt.insert(a[i/2], Relative.LeftChild);`

(C) `rfa(bt, a, 2*i);`

(D) `bt.find(Relative.Parent);`

(E) None

6. Line 12:

(A) `bt.insert(a[i+1], Relative.LeftChild);`

(B) `bt.insert(a[2*i], Relative.LeftChild);`

(C) `bt.find(Relative.Parent);`

(D) `rfa(bt, a, 2*i);`

(E) None

7. Line 13:

(A) `rfa(bt, a, 2*i);`

(B) `bt.find(Relative.Parent);`

(C) `bt.find(Relative.Root);`

(D) `rfa(bt, a, i/2);`

(E) None

8. Line 15:

(A) `if (2*i+1 < a.length){`

(B) `if (a[2*i+1] != null){`

(C) `if (2*i+1 < a.length || a[2*i+1] != null){`

(D) `if (2*i+1 < a.length && a[2*i+1] != null){`

(E) None

9. Line 16:

(A) `bt.find(Relative.Parent);`

(B) `bt.insert(a[2*i+1], Relative.RightChild);`

(C) `bt.insert(a[(i+1)/2], Relative.RightChild)`
`;`

(D) `rfa(bt, a, 2*i+1);`

(E) None

10. Line 17:

(A) `bt.find(Relative.Parent);`

(B) `bt.insert(a[i+2], Relative.RightChild);`

(C) `bt.insert(a[2*i+1], Relative.RightChild);`

(D) `rfa(bt, a, 2*i+1);`

(E) None

11. Line 18:

(A) `rfa(bt, a, 2*i+1);`

(B) `bt.find(Relative.Root);`

(C) `bt.find(Relative.Parent);`

(D) `rfa(bt, a, (i+1)/2);`

(E) None

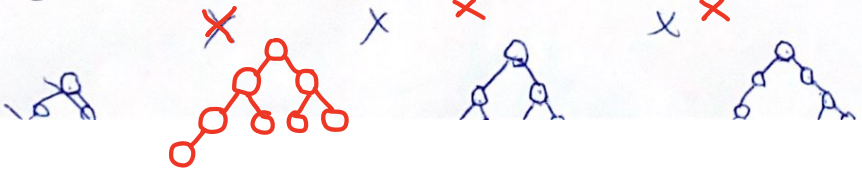
Question 4 ..... 14 points

Choose the most appropriate answer.

$2^1 - 1$ 2222 = 16 − 1 = 15

1. Suppose T is a binary tree with 8 nodes and height 4. Which of the following is surely true?

(A) T is a complete tree. (B) T is a min-heap. (C) T is a max-heap. (D) T is not a heap.



$2^6 - 1 = 63$ = complete!

Ⓔ None.

$2^{\text{something}} - 1$ = complete tree $64-1=63$

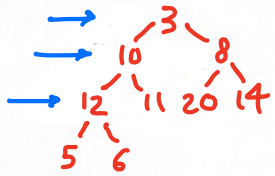
2. Suppose T is a binary tree with 63 nodes and height 6. Which of the following is surely true?

Ⓐ T is a complete tree. Ⓑ T is a min-heap. Ⓒ T is a max-heap. Ⓓ T is not a heap.
Ⓔ None.

{3,10,8,12,11,20,14,5,6}

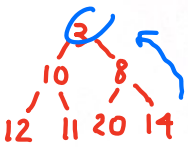
3. Given a complete binary tree T and its array representation A, which traversal algorithm visits the nodes of T in the same order as they have in the array A?

Ⓐ Pre-order Ⓑ In-order Ⓒ Post-order Ⓓ BFS Ⓔ DFS Ⓕ None



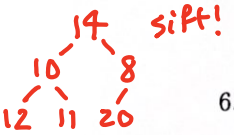
4. Given the heap $H = [3, 10, 8, 12, 11, 20, 14]$, what is H after inserting 5 and then 6?

Ⓐ 3, 4, 5, 8, 6, 12, 10, 11, 14, 20 Ⓑ 3, 5, 8, 6, 11, 20, 14, 10, 12 Ⓒ 3, 5, 6, 8, 10, 11, 12, 14, 20
Ⓓ 3, 5, 6, 10, 11, 8, 14, 20, 12 Ⓔ None



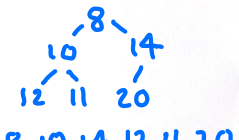
5. Given the heap $H = [3, 10, 8, 12, 11, 20, 14]$, what is H after deleting one key?

Ⓐ 8, 14, 12, 10, 11, 20 Ⓑ 8, 10, 11, 12, 14, 20 Ⓒ 8, 10, 12, 11, 20, 14 Ⓓ 8, 10, 14, 12, 11, 20
Ⓔ None



6. Given the heap $H = [3, 6, 8, 9, 23, 10, 13, 11]$, what is H after deleting two keys :

Ⓐ 3, 6, 8, 9, 23, 10. Ⓑ 8, 9, 23, 10, 13, 11. Ⓒ 8, 9, 10, 11, 13, 23. Ⓓ 8, 9, 10, 11, 23, 13.
Ⓔ None

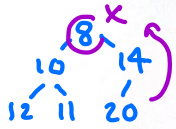


8,10,14,12,11,20

7. Given two heaps, $H_1 = [6, 10, 18, 20, 22, 24, 30]$ and $H_2 = [8, 12, 14, 19, 23]$, what is the result of merging them along with the key 13?

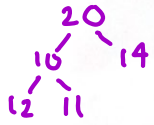
After 2nd key

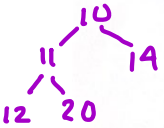
Ⓐ 6, 10, 8, 13, 18, 12, 14, 20, 22, 24, 30, 19, 23 Ⓑ 6, 8, 12, 14, 13, 18, 19, 23, 20, 22, 24, 30
Ⓒ 6, 8, 12, 13, 14, 18, 19, 20, 22, 23, 24, 30 Ⓓ 6, 8, 14, 10, 23, 12, 20, 22, 24, 30, 18, 19 Ⓔ None



Question 5 ..... 14 points

(a) (4 points)

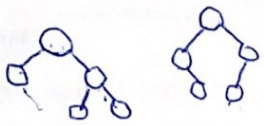




10,11,14,12,20

Remark 1. *In what follows the height of tree is the number of levels in the tree. Hence, an empty tree has height 0, whereas a tree with 1 node has height 1.*

Choose the most appropriate answer:



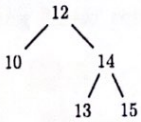
1. The maximum depth of an AVL tree with 5 nodes is:

Ⓐ 1. Ⓑ 2. Ⓒ 3. Ⓓ 4. Ⓔ 5.

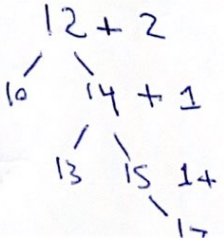
2. The worst case run time for insert in AVL tree is:

Ⓐ $O(1)$. Ⓑ $O(n)$. Ⓒ $O(\log n)$. Ⓓ $O(n \log n)$ Ⓔ $O(n^2)$.

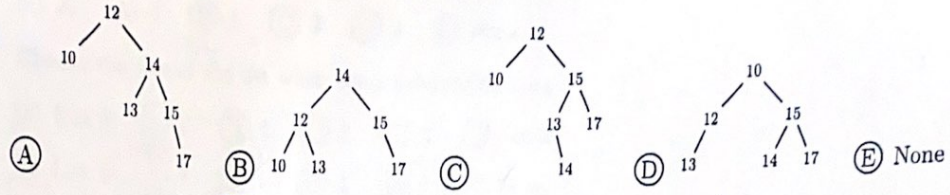
(b) (10 points) Choose the correct result in each of the following cases (follow the the convention of replacing with the smallest key in the right sub-tree when necessary):



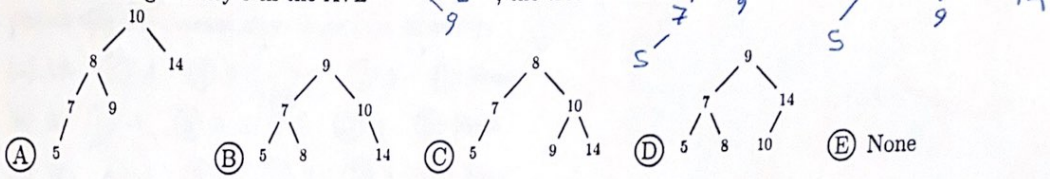
1. After inserting the key 17 in the AVL , the tree becomes:



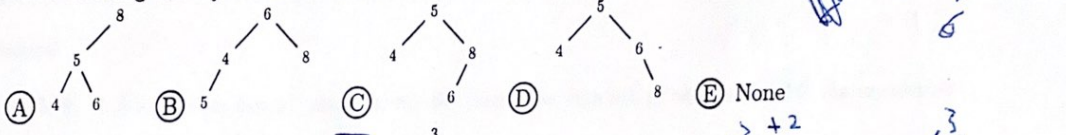
14



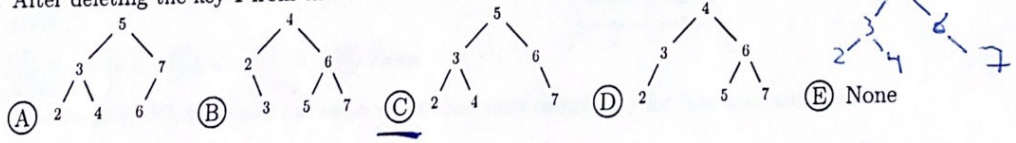
2. After inserting the key 9 in the AVL , the tree becomes:



3. After deleting the key 9 from the AVL , the tree becomes:



4. After deleting the key 1 from the AVL , the tree becomes:



Question 6 ..... 14 points

(a) (4 points) Choose the most appropriate answer:

1. The insert operation in a hash table using external chaining has a worst case run time:

Ⓐ $O(1)$ Ⓑ $O(\log n)$ Ⓒ $O(n)$ Ⓓ $O(n \log n)$ Ⓔ None

2. Consider the following hash function: The product of the rightmost digit and the leftmost digit then apply % 11 to the result. Which of the following couples of keys cause a collision?

Ⓐ 9 and 101 Ⓑ 162 and 370 Ⓒ 523 and 259 Ⓓ All of the above. Ⓔ None of the above.

(b) (10 points) Use the hash function $H(key) = key\%6$ to store the sequence of keys $16, 5, 22, 18, 31$ in a hash table of size 6. Use the following collision resolution strategies:

1. Choose the number of probes when using linear rehashing with c=1:

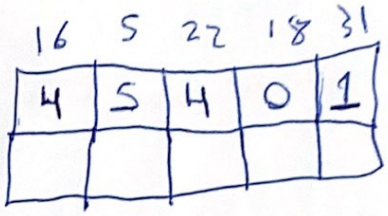
(a) 5: Ⓐ 0 Ⓑ 1 Ⓒ 2 Ⓓ 3 Ⓔ None

(b) 22: Ⓐ 0 Ⓑ 1 Ⓒ 2 Ⓓ 3 Ⓔ None

4	5	6	0	1		
1	1	3	1	1		

(c) 31: (A) 0 (B) 1 (C) 2 (D) 3 (E) None

2. Choose the size of the list when using external chaining.

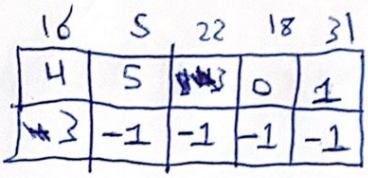


(a) List 0: (A) 0 (B) 1 (C) 2 (D) 3 (E) None

(b) List 2: (A) 0 (B) 1 (C) 2 (D) 3 (E) None

(c) List 4: (A) 0 (B) 1 (C) 2 (D) 3 (E) None

3. Choose the index of the next element when using coalesced chaining with cellar size 2 and address region size 4 (-1 means there is no next element):



(a) 16: (A) -1 (B) 0 (C) 3 (D) 5 (E) None

(b) 5: (A) -1 (B) 0 (C) 3 (D) 4 (E) None

(c) 22: (A) -1 (B) 1 (C) 2 (D) 4 (E) None

(d) 18: (A) -1 (B) 1 (C) 5 (D) 6 (E) None

Question 7 ..... 14 points

(a) (6 points)

Remark 2. *A B+ tree has two parameters, m: the maximum number of children and l: the maximum number of elements in a leaf node.*

Choose the most appropriate answer:

1. Consider a B+-tree in which the $m = l = 5$. What is the minimum number of keys in any non-root node :

(A) 2 (B) 1 (C) 4 (D) 3 (E) None

2. B+-tree and AVL tree have the same worst case time complexity for insertion and deletion:

(A) *True* (B) *False*

3. The best case running time for checking if a key exists in a B+-tree is:

(A) $O(1)$ (B) $O(\log n)$ (C) $O(n)$ (D) $O(n \log n)$ (E) None

(b) (8 points) Choose the correct result in each of the following cases (when possible, always borrow and transfer to the left). The order of the tree is $m = 3$:

1. After inserting the key 7 in the B+ tree

, the **root** of tree becomes:

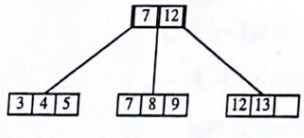
(A) 

(B) 

(C) 

(D) 

(E) None

2. After inserting the key 10 in the B+ tree 

, the **root** of the tree becomes:

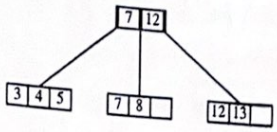
(A) 

(B) 

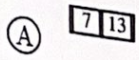
(C) 

(D) 

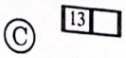
(E) None

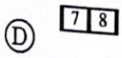
3. After deleting the key 12 from the B+ tree 

becomes: , the **root** of the tree

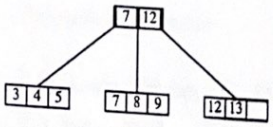
Ⓐ 

Ⓑ 

Ⓒ 

Ⓓ 

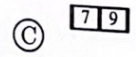
Ⓔ None

4. After deleting the key 12 from the B+ tree 

becomes: , the **root** of the tree

Ⓐ 

Ⓑ 

Ⓒ 

Ⓓ 

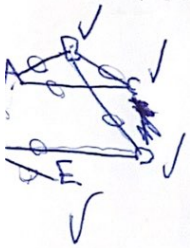
Ⓔ None

Question 8 ..... 6 points

(a) (2 points) Choose the most appropriate answer:

1. What is the maximum length of any simple path in a graph containing 100 nodes?
   Ⓐ 10. Ⓑ 100. Ⓒ 99. Ⓓ 101. Ⓔ None.
2. Which of the following is true for DFS and BFS graph traversal ?
   Ⓐ Both DFS and BFS use Stack. Ⓑ DFS uses Queue and BFS uses Stack. Ⓒ BFS uses Queue. Ⓓ DFS uses Stack. Ⓔ C and D.

(b) (4 points) Given the following graph adjacency list, answer the questions below.



A	$\to B \to C \to F$
B	$\to A \to C \to D$
C	$\to A \to B$
D	$\to B \to F$
E	$\to F$
F	$\to A \to D \to E$

1. Which of the following sequences are paths in this graph? Answer by T (true) or F (false).
   (a) (F, B, D, A) F
   (b) (A, B, D, F, A) T
   (c) (A, B, C, F, D) F
   (d) (F, A, B, C) T
2. Answer by T (true) or F (false).
   (a) The graph is a tree. F
   (b) The number of edges in the graph is 7. T
   (c) In the graph, (A, B, D, F, A) is a cycle. T
   (d) The neighbors of node C are A, B, and D. F
3. Which of the following is true for this graph ?
   Ⓐ The graph is a tree. Ⓑ The graph is acyclic. Ⓒ The graph is disconnected. Ⓓ The graph is connected. Ⓔ None.
4. The DFS traversal of this graph starting from A is (insert neighbors in the data structure in increasing alphabetic order):
   Ⓐ A, C, B, E, D, F. Ⓑ A, F, D, E, C, B.
   Ⓒ A, F, B, C, E, D. Ⓓ A, F, E, D, C, B.
   Ⓔ None.

AFEDCB


AFEDC