Final - CSC212

1 semester 2022/2023

دعواتكم ♡

Data Structures CSC 212 | **Final Exam - Spring 2022**

Date: 04/06/2022 | Duration: 3 hours

Student ID: | Name:

1	2	3.1	3.2	4	5	6	7	8	Total

1.1 ___
1.2 ___
1.3 ___
1.4 ___
1.5 ___
1.6 ___

2.1 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2.2 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2.3 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2.4 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2.5 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2.6 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2.7 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2.8 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2.9 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2.10 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2.11 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2.12 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

3.a.1 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.a.2 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.a.3 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.a.4 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.a.5 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.1 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.2 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.3 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.4 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.5 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.6 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.7 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.8 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.9 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.10 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.11 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3.b.12 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

4.a.1 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
4.a.2 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
4.a.3 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ Ⓕ
4.a.4 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
4.b Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
4.c Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
4.d Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

5.1 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
5.2 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
5.3 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
5.4 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
5.5 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
5.6 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
5.7 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

6.1 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
6.2 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
6.3 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
6.4 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
6.5 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

7.a.1 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
7.a.2 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
7.a.3 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
7.b.1 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
7.b.2 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
7.b.3 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
7.b.4 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

8.1 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
8.2.a ___
8.2.b ___
8.2.c ___
8.2.d ___
8.3.a ___
8.3.b ___
8.3.c ___
8.3.d ___
8.4 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
8.5 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
8.6 Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

King Saud University	College of Computer and Information Sciences Department of Computer Science
Data Structures CSC 212	**Final Exam - Spring 2022**
Date: 04/06/2022	Duration: 3 hours

Question 1 ..... 12 points

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

A. LinkedList	B. HashTable	C. DoubleLinkedList	D. LinkedQueue	E. BT	F. ArrayStack
G. ArrayQueue	H. BST	I. LinkedPQueue	J. BPlusTree	K. HeapPQueue	L. Graph.

1. A text editor supporting multi-step undo/redo functionality. F
2. Processing a large dataset whose size is larger than the size of main memory. J
3. Check if a string is a palindrome (reads the same in both directions, such as *level* or *radar*). C
4. Validating users' passwords without needing to store them in clear text. B
5. Organizing categories of products in an e-commerce website. H
6. Navigating a robot vacuum cleaner around a house. L

Question 2 ..... 12 points

Write the method `arrangeList` which receives a list of integers l and a number n. The method should return a new list containing all the integers in l that are greater than or equal to n first followed by all the integers in l that are less than n. The list l must not change at the end of the method.

```
public static LinkedList<Integer> arrangeList(LinkedList<Integer> l, int n) {
  List<Integer> x = new LinkedList<Integer>();
  List<Integer> y = new LinkedList<Integer>();
  if (...)
    System.out.println("List is empty");
  else {
    ...
    while (...) {
      if(l.retrieve() >= n)
        x.insert(l.retrieve());
      else
        ...
      ...
    }
    if(...)
      x.insert(l.retrieve());
    else
      y.insert(l.retrieve());
    if (...) {
      ...
      while (...) {
        x.insert(y.retrieve());
        ...
      }
      ...
    }
  }
  return ...
}
```

Example 1. *if List l: [1 5 8 4 3 6 7 2 9] and* $n = 5$.

Then, the method should return the list: [5 8 6 7 9 1 4 3 2]

B 1. Line 4:

(A) `l.hasNext()`
(B) `l.empty()`
(C) `l.full()`
(D) `l.last()`
(E) None

D 2. Line 7:

(A) `y.findNext();`
(B) `x.findFirst();`
(C) `l.findNext();`
(D) `l.findFirst();`
(E) None

D 3. Line 8:

(A) `l.retrieve()>= n`
(B) `l.retrieve()!= n`
(C) `l.last()`
(D) `!l.last()`
(E) None

B 4. Line 12:

(A) `l.findNext();`
(B) `y.insert(l.retrieve());`
(C) `y.insert(x.retrieve());`
(D) `l.retrieve();`
(E) None

B 5. Line 13:

(A) `x.findNext();`
(B) `l.findNext();`
(C) `x.remove();`
(D) `l.remove();`
(E) None

B 6. Line 15:

(A) `l.retrieve()== n`
(B) `l.retrieve()>=n`
(C) `x.retrieve()>=n;`
(D) `l.retrieve()!=n;`
(E) None

D 7. Line 19:

(A) `!x.empty()`
(B) `x.retrieve()!= null`
(C) `y.empty()`
(D) `!y.empty()`
(E) None

B 8. Line 20:

(A) `x.findFirst();`
(B) `y.findFirst();`
(C) `l.findNext();`
(D) `l.findFirst();`
(E) None

C 9. Line 21:

(A) `l.last()`
(B) `l.retrieve()!= null`
(C) `!y.last()`
(D) `!x.last()`
(E) None

C 10. Line 23:

(A) `l.retrieve();`
(B) `y.retrieve();`
(C) `y.findNext();`
(D) `l.findNext();`
(E) None

D 11. Line 25:

(A) `y.insert(l.retrieve());`
(B) `x.insert(l.retrieve());`
(C) `y.insert(x.retrieve());`
(D) `x.insert(y.retrieve());`
(E) None

B 12. Line 28:

(A) `l;`

(B) `x;`

(C) `l.retrieve();`

(D) `y;`

(E) None

Question 3 ..... 14 points

(a) Write the method `private int f(BTNode<T> t, T e, int k)`, member of `BT`, which returns the number of nodes in the level `k` of the subtree `t` having data equal to `e`. The root of the subtree (`t`) is at level 0.

```
1 private int f(BTNode<T> t, T e, int k) {
2   if(...)
3     return ...;
4   if (...)
5     return ...;
6   return ...;
7 }
```

C 1. Line 2:

(A) `if (t.data == e)`

(B) `if (t != null)`

(C) `if (t == null)`

(D) `if (e.equals(t.data))`

(E) None

A 2. Line 3:

(A) `return 0;`

(B) `return k;`

(C) `return 1;`

(D) `return f(t.left,e,k)&&f(t.right,e,k);`

(E) None

B 3. Line 4:

(A) `if (!e.equals(t.data))`

(B) `if (k == 0 && e.equals(t.data))`

(C) `if (e.equals(t.data))`

(D) `if (k == 1 && e.equals(t.data))`

(E) None

C 4. Line 5:

(A) `return f(t.left,e,k)+f(t.right,e,k);`

(B) `return 0;`

(C) `return 1;`

(D) `return 1+f(t.left,e,k)+f(t.right,e,k);`

(E) None

B 5. Line 6:

(A) `return f(t.left,e,k)+f(t.right,e,k);`

(B) `return f(t.left,e,k-1)+f(t.right,e,k-1);`

(C) `return f(t,e,k+1)+f(t,e,k+1);`

(D) `return f(t.left,e,k+1)+f(t.right,e,k+1);`

(E) None

(b) Repeat the same questions as above, but this time as a user.

```
public static <T> int f(BT<T> b, T e, int k) {
  if (...)
    ...;
  ...;
  ...;
}
private static <T> int rf(BT<T> b, T e, int k) {
  if (...)
    return ...;
  int n=0;
  if (...) {
    ...;
    ...;
  }
```

```
15    if (...) {
16      ...;
17      ...;
18    }
19  }
```

B 1. Line 2:

(A) `if (b.full())`

(B) `if (b.empty())`

(C) `if (!b.empty())`

(D) `if (e.equals(b.retrieve()))`

(E) None

C 2. Line 3: (A) `return 1;`

(B) `return rf(b,e,k);`

(C) `return 0;`

(D) `return k;`

(E) None

C 3. Line 4:

(A) `return rf(b,e,k);`

(B) `return e.equals(b.retrieve());`

(C) `b.find(relative.Root);`

(D) `b.find(relative.Parent);`

(E) None

C 4. Line 5:

(A) `return rf(b, e, k - 1);`

(B) `return rf(b,e,k+1);`

(C) `return rf(b,e,k);`

(D) `return rf(b,e,0);`

(E) None

D 5. Line 8:

(A) `if (k==1)`

(B) `if (e.equals(b.retrieve()))`

(C) `if (k==1 && e.equals(b.retrieve()))`

(D) `if (k==0 && e.equals(b.retrieve()))`

(E) None

C 6. Line 9:

(A) `return 0;`

(B) `return 1+rf(b.left,e,k)+rf(b.right,e,k);`

(C) `return 1;`

(D) `return e.equals(b.retrieve());`

(E) None

B 7. Line 11:

(A) `if (b.find(Relative.Parent)){`

(B) `if (b.find(Relative.LeftChild)){`

(C) `if (b.find(Relative.Root)){`

(D) `if (b.left != null){`

(E) None

C 8. Line 12:

(A) `n-=rf(b.left,e,k);`

(B) `n-=rf(b,e,k);`

(C) `n+=rf(b,e,k-1);`

(D) `n+=rf(b,e,k+1);`

(E) None

B 9. Line 13:

(A) `b.find(Relative.RightChild);`

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

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

(D) `return n+1;`

(E) None

B 10. Line 15:

(A) `if (b.right != null){`

(B) `if (b.find(Relative.RightChild)){`

(C) `if (b.find(Relative.Parent)){`

(D) `if (b.find(Relative.Root)){`

(E) None

D 11. Line 16:

(A) `n+=rf(b,e,k+1);`

(B) `n-=rf(b,e,k);`

(C) `n-=rf(b.right,e,k);`

(D) `n+=rf(b,e,k-1);`

Ⓔ None

A 12. Line 17:

Ⓐ `b.find(Relative.Parent);`

Ⓑ `b.find(Relative.LeftChild);`

Ⓒ `return n+1;`

Ⓓ `b.find(Relative.Root);`

Ⓔ None

Question 4 ..... 14 points

(a) Consider the following heap represented as an array: 2, 7, 5, 8, 20, 10, 12. Choose the correct answer for every operation (all operations are done on the above heap).

A 1. Heap after inserting 1: Ⓐ 1, 2, 5, 7, 20, 10, 12, 8 Ⓑ 1, 2, 5, 7, 20, 10, 8, 12 Ⓒ 2, 5, 7, 20, 10, 12, 8, 1 Ⓓ 2, 5, 7, 20, 10, 12, 1, 8 Ⓔ None

B 2. Heap after inserting 3 then 4: Ⓐ 2, 3, 4, 5, 20, 10, 12, 8, 7 Ⓑ 2, 3, 5, 4, 20, 10, 12, 8, 7 Ⓒ 2, 3, 4, 5, 8, 7, 20, 10, 12 Ⓓ 2, 3, 4, 5, 8, 10, 12, 7, 20 Ⓔ None

5,7,10,8,20,11,12 E 3. Heap after inserting 11 then deleting one key: Ⓐ 11, 2, 7, 5, 8, 20, 10, 12 Ⓑ 5, 3, 4, 20, 10, 12, 8, 7 Ⓒ 5, 7, 11, 8, 20, 10, 12 Ⓓ 5, 7, 10, 8, 20, 12, 11 Ⓔ None

C 4. Heap after deleting two keys: Ⓐ 2, 7, 5, 8, 20 Ⓑ 2, 5, 7, 20, 8 Ⓒ 7, 8, 10, 12, 20 Ⓓ 7, 10, 8, 12, 20 Ⓔ None

C (b) Suppose we have two heaps (5, 9, 6) and (7, 8, 10) represented as arrays and a key 12, what will be the resultant heap after merging them? Ⓐ 12, 5, 9, 6, 7, 8, 10 Ⓑ 5, 9, 6, 12, 7, 8, 10 Ⓒ 5, 6, 7, 9, 12, 8, 10 Ⓓ 5, 9, 6, 7, 8, 10, 12 Ⓔ None

A (c) What is the result of a bottom-up min-heap construction of the array 5, 11, 2, 7, 16, 15, 4? Ⓐ 2, 7, 4, 11, 16, 15, 5 Ⓑ 5, 11, 2, 7, 16, 15, 4 Ⓒ 2, 7, 5, 11, 16, 15, 4 Ⓓ 2, 4, 7, 11, 16, 15, 5 Ⓔ None

$h = \lceil \log 10 + 1 \rceil = 4$ C (d) What is the height of a heap containing 10 elements (an empty heap has height 0)? Ⓐ 3 Ⓑ 10 Ⓒ 4 Ⓓ 5 Ⓔ None.

Question 5 ..... 14 points

Choose the most appropriate answer:

Remark 1. *Follow the the conventions: The height of an empty tree is 0. When necessary, replace with the smallest key in the right sub-tree.*

A 1. Which of the following is true for the best case run time for findkey in AVL trees: Ⓐ O(log n). Ⓑ O(n). Ⓒ O(1). Ⓓ O(n log n). Ⓔ None.

A 2. Which of the following is true for the average case run time for findkey in BST trees: Ⓐ O(log n). Ⓑ O(n). Ⓒ O(1). Ⓓ O(n log n). Ⓔ None.

A 3. The cost of one rotation in an AVL tree is:

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

D 4. After inserting the key 5 in the AVL , the tree becomes:

(A) , (B) (C) , (D) (E) None

A 5. After inserting the key 20 in the AVL , the tree becomes:

(A) (B) (C) (D) (E) None

C 6. After deleting the key 12 from the AVL , the tree becomes:

(A) (B) (C) (D) (E) None

A 7. After deleting the key 10 from the AVL , the tree becomes:

(A) (B) (C) (D) (E) None

Question 6 ..... 14 points

Choose the most appropriate answer:

A 1. What is the average retrieval time when n keys hash to the same position? (A) $O(n)$ (B) $O(\log n)$ (C) $O(1)$ (D) $O(n^2)$ (E) None

A 2. The best case running time for finding a key in a hash table with external chaining is: (A) $O(1)$ (B) $O(\log n)$ (C) $O(n \log n)$ (D) $O(n)$ (E) None

B 3. Consider a hash table of size 7, with starting index zero, and a hash function $(3k + 4)\%7$. What is the content of the table when the sequence 1, 3, 8, 10 is inserted into the table using linear rehashing with $c = 1$? Note that '·' denotes an empty location. (A) 8, ·, ·, ·, ·, ·, 10 (B) 1, 8, 10, ·, ·, ·, 3 (C) 1, ·, ·, ·, ·, ·,3 (D) 1, 10, 8, ·, ·, ·, 3 (E) None

C 4. Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function $k\%10$, which of the following statements are true? (1) 9679, 1989, 4199 hash to the same value. (2) 1471,

6171 hash to the same value. (3) All elements hash to the same value. (4) Each element hashes to a different value. (A) (1) only (B) (2) only (C) (1) and (2) only (D) (3) and (4) (E) None

C 5. Consider the following hash function: select the two rightmost digits then apply mod 7 on the corresponding number. Which of the following couples of keys cause a collision?

(A) 3848 and 4756 (B) 3973 and 1258 (C) 162 and 35476 (D) All of the above. (E) None of the above.

Question 7 ..... 14 points

(a) Choose the most appropriate answer:

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.*

6*5 C 1. In a B+ tree with $l = m = 6$, the maximum number of keys at level 1 (root at level 0):

(A) 29 keys (B) 30 keys (C) 5 keys (D) 6 keys (E) None

A 2. Consider a B+-tree in which the maximum number of keys in a non-leaf node is 5. What is the minimum number of keys in any non-root node?

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

A 3. B+-trees and AVL trees have the same worst case time complexity for insertion and deletion:

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

(b) 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 $m = 3$:

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

, the **root** of tree becomes:

(A)

(B)

(C)

(D)

(E) None

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

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

(A)

(B)

(C)

(D)

(E) None

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

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

(A)

(B)

(C)

(D)

(E) None

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

, the **root** of the tree

becomes:

(A) [7 |] (B) [5 |] (C) [7 | 9] (D) [7 | 13] (E) None

Question 8 ..... 6 points

Given the following graph (G) adjacency matrix, answer the questions below.

	A	B	C	D	E	F
A	0	0	0	0	1	0
B	0	0	1	1	0	1
C	0	1	0	1	0	0
D	0	1	1	0	0	0
E	1	0	0	0	0	0
F	0	1	0	0	0	0

B 1. The total number of edges in G is: (A) 3 (B) 5 (C) 7 (D) 10 (E) None.

2. Which of the following sequences are paths in this graph? Answer by T (true) or F (false).

   (a) (B, C, D, B, F) T

   (b) (D, B, C, F) F

   (c) (B, D, C, F, A) F

   (d) (C, D, B, F) T

3. Answer by T (true) or F (false).

   (a) G contains only two connected subgraphs. F

   (b) There are n-1 edges in G, where n is the number of nodes. T

   (c) G is a tree. F

   (d) G is a connected graph. F

D 4. Which of the following is true for this graph?

(A) The graph is connected. (B) The graph has no cycles. (C) The graph is not connected and has no cycles. (D) The graph has cycles. (E) None.

A 5. In terms of graph representation and space requirement, which statement is true about G?

(A) Adjacency list is more efficient than adjacency matrix. (B) Adjacency matrix is more efficient than adjacency list. (C) Both representations require the same space for G. (D) Both representations require no space. (E) None.

B 6. The number of cycles in this graph is:

(A) 0. (B) 1. (C) 2. (D) 3. (E) None.

King Saud University	College of Computer and Information Sciences Department of Computer Science
Data Structures CSC 212	**Final Exam - Spring 2021**
Date: 24/04/2021	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. ArrayQueue.	**H**. BST.	**I**. LinkedPQueue.	**J**. BPlusTree.	**K**. HeapPQueue.	**L**. Graph.

1. An electronic book reader (e-reader) that allows users to easily flip through pages. C
2. An online store that displays items in a least-expensive to most-expensive order. I
3. A text processing program that prints out strings in a reversed order. F
4. A database management system that stores high-demand magazine articles. J
5. A social network platform that verifies whether a given username is available or taken. E
6. A food delivery application that helps its drivers deliver multiple orders in the shortest time possible. L

Question 2 ..... 12 points

Write the method `greaterThanK` which receives a list of integers l and a number k. The method should return a new list containing all the integers in l that are greater than k. If there is no integer in l higher than the value k, then the method should display an appropriate message and returns an empty list.

Example 1. *if List l: [1 2 3 4 5 6 7 8 9] and* $k = 5$.

Then, the method should return the list: [6 7 8 9]

```
public static LinkedList<Integer> greaterThanK(LinkedList<Integer> l, int k) {
  LinkedList<Integer> nl = new LinkedList<Integer>();
  if (...)
    System.out.println("List␣is␣empty");
  else {
    ...
    while (...){
      if(...)
        nl.insert(...);
      ...
    }
    if(...)
      nl.insert(...);
    if (...)
      System.out.println("no␣value␣in␣L␣is␣greater␣than␣K");
  }
  return ...;
}
```

B 1. Line 3:

(A) `l.hasNext()`
(B) `l.empty()`
(C) `l.full()`
(D) `l.last()`
(E) None

D 2. Line 6:

(A) `nl.findNext();`
(B) `nl.findFirst();`
(C) `l.findNext();`
(D) `l.findFirst();`
(E) None

D 3. Line 7:

(A) `l.retrieve()>= k`
(B) `l.retrieve()!= k`
(C) `l.last()`
(D) `!l.last()`
(E) None

B 4. Line 8:

(A) `l.retrieve()>= k`
(B) `l.retrieve()>k`
(C) `nl.retrieve();`
(D) `l.retrieve();`
(E) None

D 5. Line 9:

(A) `nl.retrieve()`
(B) `l.retrieve()!= null`
(C) `k`
(D) `l.retrieve()`
(E) None

B 6. Line 10:

(A) `nl.findNext();`
(B) `l.findNext();`
(C) `nl.remove();`
(D) `l.remove();`
(E) None

B 7. Line 12:

(A) `l.retrieve()>= k`
(B) `l.retrieve()>k`
(C) `nl.retrieve();`
(D) `l.retrieve();`
(E) None

C 8. Line 13:

(A) `nl.retrieve()`
(B) `l.retrieve()!= null`
(C) `l.retrieve()`
(D) `k`
(E) None

D 9. Line 14:

(A) `l.empty()`
(B) `l.full()`
(C) `nl.full()`
(D) `nl.empty()`
(E) None

B 10. Line 17:

(A) `l;`
(B) `nl;`
(C) `l.retrieve();`
(D) `n`
(E) None

Question 3 ..... 14 points

(a) Write the method **public void** `trim(int l)`, member of the class `BT`, which removes all nodes that are at level `l` or deeper. We follow the convention that the root is at level 0. Assume that `l` > 0.

```
1  public void trim(int l) {
2    ...
3  }
4  private ... recTrim(...) {
5    if (...)
6      ...
7    if (...) {
8      ...
9      ...
10   } else {
11     ...
12     ...
13   }
14 }
```

B 1. Line 2:

(A) `recTrim(root);`

(B) `recTrim(root, l);`

(C) `recTrim(null);`

(D) `recTrim(root, root);`

(E) None

A 2. Line 4:

(A) `private void recTrim(BTNode<T> t, int l){`

(B) `private boolean recTrim(BTNode<T> t){`

(C) `private void recTrim(BTNode<T> t){`

(D) `private void recTrim(BTNode<T> t1, BTNode<T> t2){`

(E) None

B 3. Line 5:

(A) `if (t.data == l)`

(B) `if (t == null)`

(C) `if (l == null)`

(D) `if (t1 == null || t2 == null)`

(E) None

C 4. Line 6:

(A) `return t1.data.equals(l)|| t2.data.equals(l);`

(B) `return true;`

(C) `return;`

(D) `recTrim(t);`

(E) None

A 5. Line 7:

(A) `if (l == 1){`

(B) `if (t1.data == l || t2.data == l){`

(C) `if (t.data == l){`

(D) `if (l == 0){`

(E) None

D 6. Line 8:

(A) `t.data = null;`

(B) `t1.left = null;`

(C) `t.left = t.right;`

(D) `t.left = null;`

(E) None

D 7. Line 9:

(A) `t2.right = null;`

(B) `t.data = null;`

(C) `t.right = t.left;`

(D) `t.right = null;`

(E) None

D 8. Line 11:

(A) `recTrim(t.left);`

(B) `recTrim(t);`

(C) `recTrim(t1.left, t1.right);`

(D) `recTrim(t.left, l - 1);`

(E) None

A 9. Line 12:

(A) `recTrim(t.right, l - 1);`

(B) `recTrim(t.right);`

(C) `recTrim(t2.left, t2.right);`

(D) `recTrim(t);`

(E) None

(b) Write the method **public static** `<T>` **void** `setNull(BT<T> bt, int l)`, user of the class `BT`, which sets to null the data of all nodes that are at level `l` or deeper. We follow the convention that the root is at level 0.

```
1  public static <T> void setNull(BT<T> bt, int l) {
2    if (...)
3      ...
4    ...
5    ...
6  }
7  private static <T> ... recSetNull(...) {
8    if (...)
9      ...
10   if (...) {
11     ...
12     ...
13   }
14   if (...) {
15     ...
16     ...
17   }
18 }
```

D 1. Line 2:

(A) **if** `(bt.retrieve()== l)`

(B) **if** `(bt.full())`

(C) **if** `(bt.find(l))`

(D) **if** `(bt.empty())`

(E) None

C 2. Line 3:

(A) **return** `bt.level();`

(B) `bt.update(l);`

(C) **return**;

(D) **return** `bt.retrieve()== l;`

(E) None

bt.Find(relative.root) E 3. Line 4:

(A) `recSetNull(bt);`

(B) `recSetNull(bt, l);`

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

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

(E) None

B 4. Line 5:

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

(B) `recSetNull(bt, l);`

(C) `bt.find(l);`

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

(E) None

B 5. Line 7:

(A) **private static** `<T>` **void** `recSetNull(BT<T> bt){`

(B) **private static** `<T>` **void** `recSetNull(BT<T> bt, int l){`

(C) **private static** `<T>` **boolean** `recSetNull(BT<T> bt){`

(D) **private static** `<T>` **void** `recSetNull(int l){`

(E) None

A 6. Line 8:

(A) **if** `(l <= 0)`

(B) **if** `(bt == null)`

(C) **if** `(bt.level()== l)`

(D) **if** `(l > 0)`

(E) None

B 7. Line 9:

(A) `bt.remove();`

(B) `bt.update(null);`

(C) `bt.update(l);`

(D) `bt.insert(null);`

(E) None

D 8. Line 10:

(A) `if (bt.find(Relative.Parent)){`

(B) `if (bt.retrieve()== l)){`

(C) `if (bt.find(Relative.Root)){`

(D) `if (bt.find(Relative.LeftChild)){`

(E) None

D 9. Line 11:

(A) `recSetNull(bt);`

(B) `recSetNull(l - 1);`

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

(D) `recSetNull(bt, l - 1);`

(E) None

D 10. Line 12:

(A) `recSetNull(l);`

(B) `recSetNull(bt);`

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

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

(E) None

D 11. Line 14:

(A) `if (bt.find(Relative.Parent)){`

(B) `if (bt.retrieve()== l)){`

(C) `if (bt.find(Relative.Root)){`

(D) `if (bt.find(Relative.RightChild)){`

(E) None

B 12. Line 15:

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

(B) `recSetNull(bt, l - 1);`

(C) `recSetNull(bt);`

(D) `recSetNull(l - 1);`

(E) None

A 13. Line 16:

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

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

(C) `recSetNull(l);`

(D) `recSetNull(bt);`

(E) None

Question 4 ..... 14 points

Choose the most appropriate answer.

E 1. Consider an array A representing a min-heap with 1024 nodes, where the root is stored at position 1. Which of the following is **necessarily** true?

(A) $A[3] \leq A[1024]$. (B) $A[3] \geq A[1024]$. (C) $A[3] \neq A[1024]$. (D) $A[3] == A[1024]$. (E) None.

D 2. Suppose T is a binary tree with 128 nodes and height 9 (an empty tree has height 0). Which of the following is **necessarily** false?

(A) T is a complete tree. (B) T is a min-heap. (C) T is a max-heap. (D) A, B, and C. (E) None.

E 3. Which traversal algorithm visits the nodes of a max-heap in decreasing order?

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

B 4. You have a min-heap of size 63 with the key 1 at root. You insert another key 1, and sift-up stops immediately without doing any swaps. After this insert, the number of keys equal to 1 in the heap is:

(A) 2. (B) At least 7. (C) At most 7. (D) At most 6. (E) None

E 5. Given the heap $H = [2, 9, 8, 13, 11, 17, 14]$, what is H after inserting 3 and then 7?

(A) 2, 3, 8, 7, 13, 9, 11, 14, 17 (B) 2, 3, 8, 7, 11, 17, 14, 9, 13 (C) 2, 3, 7, 8, 9, 11, 13, 14, 17

e

(D) 2, 3, 7, 9, 11, 8, 14, 17, 13 (E) None

9

D 6. Given the heap $H = [1, 5, 2, 8, 7, 16, 10]$, what is H after deleting one key?

(A) 2, 10, 8, 5, 7, 16 (B) 2, 5, 7, 8, 10, 16 (C) 2, 5, 8, 7, 16, 10 (D) 2, 5, 10, 8, 7, 16
(E) None

B 7. Given the heap $H = [2, 4, 7, 9, 17, 10, 15, 11]$, what is H after deleting two keys?

(A) 2, 4, 7, 9, 17, 10. (B) 7, 9, 10, 11, 17, 15. (C) 7, 9, 17, 10, 15, 11. (D) 7, 9, 10, 11, 15, 17.
(E) None

Question 5 ..... 14 points

Choose the most appropriate answer:

Remark 1. *Follow the the conventions:*

- *The height of an empty tree is 0.*
- *When necessary, replace with the smallest key in the right sub-tree.*

D 1. The maximum height of an AVL tree with 7 nodes is:

(A) 1. (B) 2. (C) 3. (D) 4. (E) 5.

min = 0
Max = 2

A 2. The minimum number of rotations caused after inserting a node into an AVL tree with n nodes and height h is (a single rotation is counted 1; a double rotation is counted 2):

(A) 0. (B) 1. (C) 2. (D) h (E) n.

C 3. The worst case run time for delete in an AVL tree is:

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

B 4. After inserting the key 17 in the AVL , the tree becomes:

(A)

(B)

(C)

(D) (E) None

D 5. After inserting the key 20 in the AVL , the tree becomes:

(A)

(B)

(C)

(D) (E) None

D 6. After deleting the key 4 from the AVL , the tree becomes:

A 7. After deleting the key 10 from the AVL , the tree becomes:

Question 6 ..... 14 points

Choose the most appropriate answer:

B 1. What is a hash function?

(A) A function that allocates memory to keys (B) A function that computes the location of the key in the array (C) A function that creates an array (D) A function that computes the location of the values in the array (E) None

C 2. A hash table of size 10 uses open addressing with hash function h(k)=k mod 10, and linear re-hashing with $c = 1$. The figure below shows the table after inserting 6 keys. Which one of the following choices gives a possible order in which the key values could have been inserted in the table?

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

(A) 46, 42, 34, 52, 23, 33 (B) 34, 42, 23, 52, 33, 46 (C) 46, 34, 42, 23, 52, 33 (D) 42, 46, 33, 23, 34, 52 (E) None of the above.

C 3. Given the following keys (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function key mod 10, which of the following statements are true?

i. 9679, 1989, 4199 hash to the same value. ii. 1471, 6171 hash to the same value. iii. All keys hash to the same value. iv. Each key hashes to a different value.

(A) i only (B) ii only (C) i and ii only (D) iii or iv (E) None of the above.

C 4. What is the worst case search time of a hashing using separate chaining?

(A) $O(1)$ (B) $O(\log n)$ (C) $O(n)$ (D) $O(n \log n)$ (E) None of the above.

C 5. Consider the following hash function: select the two rightmost digits then apply mod 7 on the corresponding number. Which of the following couples of keys cause a collision?

(A) 3848 and 4756 (B) 3973 and 1258 (C) 162 and 35476 (D) All of the above. (E) None of the above.

Question 7 ..... 14 points

Choose the most appropriate answer (in questions 2 to 7 below, $m = l = 3$):

B 1. The leaves of a B+ tree of order 4 can contain the following number of elements (suppose $l = m$):

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

C 2. After inserting the key 7 in the B+ tree

, the **root** of tree becomes:

(A)

(B)

(C)

(D)

(E) None

A 3. After inserting the key 14 in the B+ tree , the **root** of the tree becomes:

(A)

(B)

(C)

(D)

(E) None

C 4. After inserting the key 10 in the B+ tree , the **root** of the tree becomes:

(A)

(B)

(C)

(D)

(E) None

B 5. After deleting the key 9 from the B+ tree , the **root** of the tree becomes:

(A)

(B)

(C)

(D)

(E) None

D 6. After deleting the key 12 from the B+ tree , the **root** of the tree becomes:

(A)

(B)

(C)

(D)

(E) None

A 7. After deleting the key 11 from the B+ tree , the **root** of the tree becomes:

(A)

(B)

(C)

(D)

(E) None

Question 8 ..... 6 points

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

1. What is the least number of edges a graph with 50 nodes must have to be connected?
   (A) 50. (B) 51. (C) 49. (D) 100. (E) None.

B 2. When searching for a node (not too far from the source node) in a graph, which of the following is true?
   (A) DFS is more appropriate than BFS. (B) BFS is more appropriate than DFS. (C) DFS is not suitable at all for this case. (D) BFS is not suitable at all for this case. (E) None.

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

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

1. Which of the following sequences are paths in this graph? Answer by T (true) or F (false).
   (a) (A, D, B, E, F) T
   (b) (D, B, F, A) F
   (c) (E, B, F, C, D) F
   (d) (C, F, B, D, A) T

2. Answer by T (true) or F (false).
   (a) The number of edges in this graph is 6. F T
   (b) The graph can accept new unique edges. T
   (c) In the graph, (A, C, F, B, A) is a cycle. F

(d) The graph is connected. T

D 3. Which of the following is true for this graph ?

(A) The graph is disconnected. (B) The graph has no cycles. (C) The graph is connected and has no cycles. (D) The graph has cycles. (E) None.

C 4. The BFS traversal of this graph starting from node C is (insert neighbors in the data structure in increasing alphabetic order):

(A) C, F, D, B, E, A. (B) C, A, D, B, E, F.
(C) C, A, F, D, B, E. (D) C, F, E, B, D, A.
(E) None.

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 }
```

D 1. Line 2:

- (A) `int maxL = -1;`
- (B) `int maxL = 1;`
- (C) `int maxL = l.retrieve();`
- (D) `int maxL = 0;`
- (E) None

E 2. Line 3:

- (A) `l.insert(true);`
- (B) `l.update(false);`
- (C) `l.findNext();`
- (D) `maxL = l.length();`
- (E) None L.findFirst

L.FindFirst():

C 3. Line 5:

(A) `int cpt = 1;`

(B) `int cpt = -1;`

(C) `int cpt = 0;`

(D) `int cpt = l.retrieve();`

(E) None

E 4. Line 7:

(A) `if (l.retrieve()> cpt)`

(B) `if (l.retrieve()> maxL)`

(C) `if (l.retrieve())`

(D) `if (l.last())`

L.retrive ==0 (E) None

C 5. Line 8:

(A) `maxL++;`

(B) `cpt--;`

(C) `cpt++;`

(D) `l.findFirst();`

(E) None

B 6. Line 10:

(A) `continue;`

(B) `break;`

(C) `l.findFirst();`

(D) `l.finNext();`

(E) None

B 7. Line 11:

(A) `if (l.retrieve())`

(B) `if (l.last())`

(C) `if (maxL == 0)`

(D) `if (cpt == 0)`

(E) None

D 8. Line 14:

(A) `maxL = 0;`

(B) `maxL = cpt;`

(C) `l.findFirst();`

(D) `l.findNext();`

(E) None

C 9. Line 16:

(A) `if (cpt == maxL)`

(B) `if (cpt < maxL)`

(C) `if (cpt > maxL)`

(D) `if (cpt == 0)`

(E) None

C 10. Line 17:

(A) `cpt = maxL;`

(B) `maxL++;`

(C) `maxL = cpt;`

(D) `cpt++;`

(E) None

B 11. Line 18:

(A) `if (maxL < 0)`

(B) `if (l.last())`

(C) `if (maxL > 0)`

(D) `if (l.retrieve())`

(E) None

B 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 }
```

D 1. Line 2:

- (A) rfa(a, 0)
- (B) null
- (C) rfa;
- — (D) rfa(a, 1)
- (E) None

B 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

B 3. Line 6:

- (A) current;
- — (B) null
- (C) i
- (D) a[i/2]
- (E) None

D 4. Line 7:

- (A) root;
- (B) a[i];
- (C) new BTNode<T>(i)
- — (D) new BTNode<T>(a[i])
- (E) None

C 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

B 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

D 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 }
```

A 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

D 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

B 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

B 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

A 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

D 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

B 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

D 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

B 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

D 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

C 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.

E 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!

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

Ⓔ None.

A 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.

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

D 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

3,5,8,6,11,20,14,12,10 E 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

D 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

D 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

A 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?

Ⓐ 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

After 2nd key

Question 5 ..... 14 points

(a) (4 points)

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:

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

Ⓐ 1. Ⓑ 2. Ⓒ 3. Ⓓ 4. Ⓔ 5.

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

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

10,11,14,12,20

(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):

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

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

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

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

Question 6 ..... 14 points

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

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

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

E 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?

(A) 9 and 101 (B) 162 and 370 (C) 523 and 259 (D) All of the above. (E) 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:

B (a) 5: (A) 0 (B) 1 (C) 2 (D) 3 (E) None

D (b) 22: (A) 0 (B) 1 (C) 2 (D) 3 (E) None

		Probs
0	22	3
1	18	2
2	31	2
3		
4	16	1
5	5	1

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

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

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

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

C (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):

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

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

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

A (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:

A 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

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

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

A 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 $m = 3$:

B 1. After inserting the key 7 in the B+ tree [3|5|9], the **root** of tree becomes:

(A) [5|7] (B) [7|] (C) [5|] (D) [9|] (E) None

A 2. After inserting the key 10 in the B+ tree, the **root** of the tree becomes:

(A) [7|10] (B) [7|9] (C) [8|12] (D) [7|12] (E) None

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

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

Ⓐ

Ⓑ

Ⓒ

Ⓓ

Ⓔ None

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

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

Ⓐ

Ⓑ

Ⓒ

Ⓓ

Ⓔ None

Question 8 ..... 6 points

Nodes اذا حسب ال B
edges اذا حسب ال C

(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.

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

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
   there is cycle

   (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

D 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.

D 4. The DFS traversal of this graph starting from A is (insert neighbors in the data structure in increasing alphabetic order): A B C

   Ⓐ 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.

King Saud University	College of Computer and Information Sciences Department of Computer Science
Data Structures CSC 212	**Final Exam - Spring 2020**
Date: 02/05/2020	Duration: 3 hours

Do not answer in this sheet and do not submit it.

Question 1 ..... 20 points

(a) Choose the correct frequency for every line as well as the total O of the following code:

```
1 sum = 1;
2 for (i = 1; i <= n; i++) {
3   sum+= i;
4   for (j = i; j >= 2; j--)
5     sum--;}
```

1. Line 1: (A) 1 (B) 2 (C) 0 (D) n (E) $2n$
2. Line 2: (A) n (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n-2$
3. Line 3: (A) n (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n-2$
4. Line 4: (A) n^2 (B) $n(n-1)/2$ (C) $(2n+1)/2$ (D) $(2n-1)/2$ (E) $n(n+1)/2$
5. Line 5: (A) n^2 (B) $n(n-1)/2$ (C) $(2n+1)/2$ (D) $(2n-1)/2$ (E) $n(n+1)/2$
6. Total O: (A) 1 (B) n (C) n^2 (D) $n\log(n)$ (E) n^3

(b) Choose the correct frequency for every line as well as the total O of the following code:

```
1 count = 0;
2 for (i = 1; i < n+1; i++)
3   count ++;
4 for (j = 0; j <= count; j++)
5   k = j+1;
```

1. Line 1: (A) 0 (B) 1 (C) 2 (D) n (E) n^2
2. Line 2: (A) n (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n-2$
3. Line 3: (A) n (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n-2$
4. Line 4: (A) $count+1$ (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n(n+1)/2$
5. Line 5: (A) $count+1$ (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n(n-1)/2$
6. Total O: (A) 1 (B) n (C) n^2 (D) $n\log(n)$ (E) n^3

(c) Choose the correct frequency for every line as well as the total O of the following code:

```
1 int i = 1;
2 while (i < n) {
3   i++;
4   if (i > 7) break;
5 }
```

1. Line 1: (A) 1 (B) 0 (C) i (D) n (E) $n+1$
2. Line 2: (A) 8 (B) 7 (C) n (D) $n-1$ (E) $n+1$
3. Lines 3: (A) n (B) $n-1$ (C) 6 (D) 7 (E) 8
4. Lines 4: (A) n (B) $n-1$ (C) 6 (D) 7 (E) 8
5. Tightest Total O: (A) 1 (B) n (C) $\log(n)$ (D) n^2 (E) 2^n

(d) Choose the correct answer:

1. $n^7 + n^4 + n^2 + \log n$ is : (A) $O(n^2)$ (B) $O(n^4)$ (C) $O(n^7)$ (D) $O(\log(n))$ (E) None
2. $2^n + n!$ is : (A) $O(n^2)$ (B) $O(2^n)$ (C) $O(n!)$ (D) $O(n^n)$ (E) None
3. $n + \log n^3 + 6$ is : (A) $O(n)$ (B) $O(\log n^3)$ (C) $O(n \log n)$ (D) $O(n^3)$ (E) None
4. The time complexity of inserting an element in a heap of n elements is:
   (A) $O(n^2)$ (B) $O(n)$ (C) $O(2^n)$ (D) $O(\log(n))$ (E) None
5. $n^2 + n \log n^4$ is : (A) $O(n)$ (B) $O(n^2)$ (C) $O(n \log(n))$ (D) $O(n^4)$ (E) None
6. $n^2 + 1000n$ is : (A) $O(n)$ (B) $O(n^2)$ (C) $O(n \log(n))$ (D) $O(nn^2)$ (E) None
7. $n^4 \log n + n!$ is : (A) $O(n!)$ (B) $O(n^4)$ (C) $O(n^5)$ (D) $O(\log(n))$ (E) None
8. Algorithm A is $O(n)$, and Algorithm B is $O(2n)$. Given the same input:
   (A) A always finishes before B. (B) B always finishes before A. (C) A and B finish at the same time. (D) B requires double the time taken by A. (E) None

Question 2 ..... 20 points

(a) Write the method `removeOddElems`, member of the class `LinkedList`, that removes all the elements having an odd position (the position of the first element is 0). The method signature is: **public void** `removeOddElems()`.

Example 1. *If* $l : A \to B \to C \to D \to E$, *then* `l.oddElems()` *returns:* $A \to C \to E$.

C 1. Line 1:

Ⓐ **while** `(head != null)`

Ⓑ **if** `(head.next != null)`

Ⓒ **if** `(head == null)`

Ⓓ **if** `(head != null)`

Ⓔ None

E 2. Line 2:

Ⓐ **return** `head;`

Ⓑ **return false;**

Ⓒ **return null;**

Ⓓ **return** `current;`

return: Ⓔ None

A 3. Line 3:

Ⓐ `Node<T> p = head;`

Ⓑ `Node<T> p = head.next;`

Ⓒ `Node<T> p = current;`

Ⓓ `Node<T> p = head.next.next;`

Ⓔ None

E 4. Line 4:

Ⓐ **while** `(p.next != null){`

Ⓑ **while** `(p != null){`

Ⓒ **while** `(p.next.next != null){`

Ⓓ **while** `(current.next != null){`

while(P!=null && P.Next!=null) Ⓔ None

C 5. Line 5:

Ⓐ `p.next = p;`

Ⓑ `p = p.next;`

Ⓒ `p.next = p.next.next;`

Ⓓ `current.next = p;`

Ⓔ None

A 6. Line 6:

(A) `p = p.next;}`

(B) `p.next.next = current.next;}`

(C) `p = p.next.next;}`

(D) `p.next.next = p.next;}`

(E) None

(b) Consider the function `f` below, member of `LinkedList`:

```
public void f(int n) {
  Node<T> p = head; Node<T> q =  null;
  for (int i = 0; i < n; i++) {
    q = p;
    p = p.next;
  }
  if (p != null) {
    while (p.next != null)
      p = p.next;
    p.next = head;
    head = q.next;
    q.next = null;
  }
}
```

Choose the correct result in each of the following cases:

C 1. The list `l`: A, B, C, D, E, after calling `l.f(2)`, `l` becomes:

(A) A, B (B) D, E, A, B, C (C) C, D, E, A, B (D) A, D, E, B, C (E) None

E 2. The list `l`: A, B, C, D, E, after calling `l.f(0)`, `l` becomes:

(A) *empty* (B) A, B, C, D, E (C) B, C, D, E, A (D) B, C, D, E (E) None

B 3. The list `l`: A, B, C, D, E, after calling `l.f(5)`, `l` becomes:

(A) *empty* (B) A, B, C, D, E (C) E, D, C, B, A (D) A, D, E, B, C (E) None

D 4. The list `l`: A, B, C, D, E, after calling `l.f(1)`, `l` becomes:

(A) A (B) E, A, B, C, D (C) C, D, E, A, B (D) B, C, D, E, A (E) None

Question 3 ..... 20 points

Write the method **public** `<T>` **boolean** `isReverse(Queue<T> q1, Queue<T> q2)`, user of the ADT `Queue`. The method checks if `q2` is the reverse of `q1`. The two queues must not change after the call.

1. Line 1:

   (A) **if** `(q1.length()== q2.length())`

   (B) **if** `((q1.length()== 0)&& (q2.length()==0))`**return true**;

   (C) **if** `(q1.length()!= q2.length())`**return false**;

   (D) **if** `((q1.length()== 0 )|| (q2.length()== 0))`**return true**;

   (E) None

2. Line 2:

   (A) `Stack<T> x1 =` **new** `Stack<T>(), x2 =` **new** `Stack<T>();`

   (B) `Queue<T> x1 =` **new** `LinkedQueue<T>(), x2 =` **new** `LinkedQueue<T>();`

   (C) `Stack<T> x1 =` **new** `LinkedStack<T>(), x2 =` **new** `LinkedStack<T>();`

   (D) `Stack<T> x =` **new** `LinkedStack<T>();`

   (E) None

3. Line 3:

   (A) **while** `(!q1.empty())`

   (B) **while** `(q1.length()!= 0)`

   (C) **while** `(x1.length()!= 0)`

   (D) **while** `(!x.full ())`

   (E) None

4. Line 4:

   (A) `x.push(q1.retrieve());`

   (B) `x1.push(current.data);`

   (C) `x1.push(q1.serve());`

   (D) `x1.enquque(q1.serve());`

   (E) None

5. Line 5:

   (A) **while** `(!x.empty()){`

   (B) **while** `(x1.size()>0){`

   (C) **boolean** `eq =` **false**;

   (D) **boolean** `eq =` **true**;

   (E) None

6. Line 6:

   (A) `T e1 = x1.pop(), e2 = x2.pop();`

Ⓑ `T e1 = x.pop(), e2 = q2.serve();`

Ⓒ **while** `(!x1.empty()){`

Ⓓ **while** `(x1.size()>0){`

Ⓔ None

7. Line 7:

Ⓐ **if** `(!e1.equals(e2))`**break**;

Ⓑ `T e1 = x1.pop(), e2 = q1.serve();`

Ⓒ **if** `(!e1.equals(e2))`

Ⓓ `T e1 = x1.pop(), e2 = q2.serve();`

Ⓔ None

8. Line 8:

Ⓐ **if** `(!e1.equals(e2))`**continue**;

Ⓑ **if** `(!e1.equals(e2))eq =` **false**;

Ⓒ `x2.push(e1); q2.enqueue(e1); }`

Ⓓ **return false**;

Ⓔ None

9. Line 9:

Ⓐ `x.push(e1); q2.enqueue(e2); }`

Ⓑ `x2.push(e1); q2.enqueue(e2); }`

Ⓒ `x1.push(e1); q2.enqueue(e1); }`

Ⓓ **return false**

Ⓔ None

10. Line 10:

Ⓐ **while** `(!q2.empty())`

Ⓑ **while** `(!x2.empty())`

Ⓒ **while** `(!q1.full())`

Ⓓ **while** `(!x.last())`

Ⓔ None

11. Line 11:

Ⓐ `q1.enqueue(x1.pop());`

Ⓑ `q1.enqueue(x2.pop());`

Ⓒ `q2.enqueue(x.pop());`

Ⓓ `q2.enqueue(x2.pop());`

Ⓔ None

12. Line 12:

(A) **return false;**

(B) **return true;**

(C) **return** `e1.equals(e2);`

(D) **return** `eq;`

(E) None

Question 4 ..... 20 points

A player wants to escape a maze where at most two options are available at each step: go straight or turn. This maze can be represented by a binary tree where the data is a character that can take four values: 'B' for begin (only at the root), 'S' for go straight, 'T' for turn, or 'X' meaning this is an exit. Exits are located at leaf nodes only, but not all leaf nodes are exits, they could be dead ends.

Example 2. *In this maze, you see that you reach the exit by moving:* `T,T,X` *staring from the root.*

This is the class `MNode`:

```
class MNode {
  public char data;
  public MNode left, right;
  ...
}
```

(a) Write the method **private boolean** `follow(MNode t, String path)`, which tests if the path indicated by `path` and starting from `t` is valid. A path is valid if its directions are available (not necessarily leading to an exit).

Example 3. *In the maze shown above, the paths: "TT", "TTX", "ST" are valid, whereas the paths "SS" and "TST" are not valid.*

```
1  private boolean follow(MNode t, String path) {
2    MNode p = t;
3    int i = 0;
4    while (...) {
5      char c = path.charAt(i++);
6      if (...)
7        ...;
8      else if (...)
9        ...;
10     else
11       return ...;
12   }
13   return ...;
14 }
```

A 1. Line 4:

Ⓐ **while** `(p!=null && i<path.length()){`

Ⓑ **while** `(p==null){`

Ⓒ **while** `(i==path.length()){`

Ⓓ **while** `(i<path.length()){`

Ⓔ None

B 2. Line 6:

(A) **if** `(p.left.data==c)`

(B) **if** `(p.left!=null && p.left.data==c)`

(C) **if** `(p.data==c)`

(D) **if** `(p.left!=null)`

(E) None

B 3. Line 7:

(A) `p.data=c;`

(B) `p=p.left;`

(C) `p.left=p;`

(D) **return false;**

(E) None

C 4. Line 8:

(A) **else if** `(p.data==c)`

(B) **else if** `(p.right!=null)`

(C) **else if** `(p.right!=null && p.right.data==c)`

(D) **else if** `(p.right.data==c)`

(E) None

A 5. Line 9:

(A) `p=p.right;`

(B) `p.right=p;`

(C) `p.data=c;`

(D) **return false;**

(E) None

B 6. Line 11:

(A) **return** `p.data!='X';`

(B) **return false;**

(C) **return** `p.data=='X';`

(D) **return** `t.data=='X';`

(E) None

B 7. Line 13:

(A) **return** `p==null;`

(B) **return true;**

(C) **return** `p.data=='T'||p.data=='S';`

(D) **return** `p.data!='X';`

(E) None

(b) Write the method **private boolean** `escape(MNode t)`, which searches preorder for an exit starting at `t` and

returns true if it finds one.

```
1  private boolean escape(MNode t){
2    LinkStack<MNode> s = new LinkStack<MNode>();
3    MNode p = ...;
4    while (...) {
5      if (...)
6        ...;
7      if (...)
8        ...;
9      else if (...)
10       break;
11     else
12       ...;
13   }
14   return ...;
15 }
```

B 1. Line 3:

(A) `MNode p=root;`

(B) `MNode p=t;`

(C) `MNode p=t.right;`

(D) `MNode p=t.left;`

(E) None

E 2. Line 4:

(A) **while** `(p.data!='X'){`

(B) **while** `(!s.empty()){`

(C) **while** `(p.data!='S'){`

(D) **while** `(p.data=='X'){`

p!=null && p.data != 'x' (E) None

B 3. Line 5:

(A) **if** `(p.right==`**null**`)`

(B) **if** `(p.right!=`**null**`)`

(C) **if** `(p.right.data=='T')`

(D) **if** `(p.right.data=='S')`

(E) None

D 4. Line 6:

(A) `s.push('S');`

(B) `s.push(p);`

(C) `s.push(p.left);`

(D) `s.push(p.right);`

(E) None

D 5. Line 7:

(A) **if** `(p.left.data=='T')`

(B) **if** `(p.left.data!='S')`

(C) **if** `(p.left==null)`

(D) **if** `(p.left!=null)`

(E) None

A 6. Line 8:

(A) `p=p.left;`

(B) `s.push(p.left);`

(C) `p=p.right;`

(D) `s.push('T');`

(E) None

D 7. Line 9:

(A) **else if** `(s.pop()=='X')`

(B) **else if** `(p.data == 'S')`

(C) **else if** `(st.full())`

(D) **else if** `(s.empty())`

(E) None

A 8. Line 12:

(A) `p=s.pop();`

(B) `s.push(p);`

(C) `p=s.push();`

(D) `p=t;`

(E) None

E 9. Line 14:

(A) **return** `p.data='X';`

(B) **return** `p.data!='X';`

(C) **return** `p!=null;`

(D) **return** `p.data=='S';`

return P.data ==='X' (E) None

Question 5 ..... 10 points

(a) Choose the most appropriate answer.

E 1. In a min heap:

Ⓐ All keys at level k are smaller than all keys in level $k+1$. Ⓑ The largest key is always at the last level. Ⓒ The key of the left child of any node is smaller than the key of its right child. Ⓓ All of the above. Ⓔ None of the above.

C 2. The worst case run time for bottom-up heap construction (min or max) is:

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

A 3. The worst case run time for retrieving (without removing) the minimum key from a min heap is:

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

A 4. Suppose we are in the bottom-up heap constructing phase of heap-sort. So far, the array looks like this: 16, 14, 15, 10, 12, 27, 28. How many heap-down operations were done on the root?

Ⓐ 1. Ⓑ 2. Ⓒ 3 Ⓓ 4. Ⓔ None.

(b) Consider the following heap represented as an array: 20, 18, 12, 10, 11, 5, 2, 6, 4, 3. Choose the correct answer for every operation (all operations are done on the above heap).

A 1. Heap after inserting 19:

Ⓐ 20, 19, 12, 10, 18, 5, 2, 6, 4, 3, 11 Ⓑ 20, 19, 12, 11, 18, 5, 2, 6 ,4, 11, 3 Ⓒ 20, 18, 12, 10, 11, 5, 2, 6, 4, 3, 19 Ⓓ 20, 18, 12, 10, 19, 5, 2, 6, 4, 3,11 Ⓔ None

C 2. Heap after inserting 1 **then** 23: Ⓐ 20,18, 23, 10, 11, 12, 2, 6, 4, 3, 1, 5 Ⓑ 20, 18, 12, 10, 11, 5, 2, 6, 4, 3, 1, 23 Ⓒ 23, 18, 20, 10, 11, 12, 2, 6, 4, 3, 1, 5 Ⓓ 23, 20, 18, 10, 11, 12, 2, 6, 4, 3, 1, 5 Ⓔ None

D 3. Heap after deleting one key: Ⓐ 20, 18, 12, 10, 11, 5, 2, 6, 4 Ⓑ 3, 18, 12, 10, 11, 5, 2, 6, 4 Ⓒ 12, 18, 5, 10, 11, 3, 2, 6, 4 Ⓓ 18, 11, 12, 10, 3, 5, 2, 6, 4 Ⓔ None

E 4. Heap after deleting two keys: Ⓐ 4, 18, 12, 10, 11, 5, 2, 6 Ⓑ 18, 11, 12, 10, 4, 5, 2, 6 Ⓒ 12, 4, 5, 10, 11, 3, 2, 6 Ⓓ 12, 11, 4, 10, 3, 5, 2, 6 Ⓔ None

Question 6 ..... 10 points

(a) Choose the most appropriate answer:

B 1. Adjacency matrix takes $O(n^2)$ memory space, when is it appropriate to use it instead of adjacency list:

Ⓐ The graph is sparse. Ⓑ The graph is dense. Ⓒ The graph is unweighted Ⓓ The number of edges is less than the number of nodes. Ⓔ None.

C 2. You apply DFS from a given node and you find out that all nodes were visited. This means:

Ⓐ The graph is directed. Ⓑ Some nodes have no edges. Ⓒ The graph is connected. Ⓓ The graph contains cycles. Ⓔ None.

(b) Given the following graph adjacency matrix, answer the questions below.

	A	B	C	D	E	F
A		3	3			5
B	3		1	5		
C	3	1				
D		5				2
E						
F	5			2		

1. Which of the following sequences are simple paths in this graph? Answer by T (true) or F (false).
   (a) $(C, B, D, E,)$ F
   (b) (C, B, A) T
   (c) (C, A, F, D, B) T
   (d) (B, A, F, D, B, C) F
2. Answer by T (true) or F (false).
   (a) The graph is connected. F
   (b) The number of edges in the graph is 6. T
   (c) (A, B, D, F, A) is a cycle. T
   (d) The shortest path from F to B is (F, A, B). T اذا بنعتمد على عدد ال nodes مو ال weight

C 3. The BFS traversal of this graph starting from A is (insert neighbors in the data structure in increasing alphabetic order):

Ⓐ A, D, B, C, F. Ⓑ A, B, C, D, F. Ⓒ A, B, C, F, D. Ⓓ A, F, B, C, D. Ⓔ A, C, B, F, D.

A 4. The DFS traversal of this graph starting from A is (insert neighbors in the data structure in increasing alphabetic order):

Ⓐ A, F, D, C, B. Ⓑ A, F, C, D, B. Ⓒ A, B, C, F, D. Ⓓ A, F, B, C, D. Ⓔ A, C, B, F, D.

King Saud University	College of Computer and Information Sciences Department of Computer Science
Data Structures CSC 212	**Final Exam - Spring 2019**
Date: 20/06/2019	Duration: 3 hours

Guidelines: No calculators or any other electronic devices are allowed in this exam.

Student ID:	Name:
Section:	Instructor:

1.1,1.2	1.3	2	3	4	5	6	7	8	Total

Question 1 ..... 20 points

(a) (6 points) Choose the most important reason among A to E for each of the following decisions:

A. Guarantees correctness. **B**. Improves worst-case run time. **C**. Improves the best-case run time.

D. Uses less memory. **E**. Simplifies the code.

1. Add a tail pointer to the class `LinkedQueue`. B
2. Use three values for the status of a cell in linear rehashing (*empty, occupied and deleted*). A
3. In the second phase of heap sort, put the removed element after the end of the heap. D
4. Use a heap instead of a linked priority queue in heap sort. B
5. Use a queue instead of a stack to store the nodes during breadth-first search of a graph. A
6. Use % (modulus) instead of an if-else in the methods `enqueue` and `serve` of `ArrayQuue`. E

(b) (6 points) Choose the run time from A to D for each of the following cases:

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

1. The best case of `LinkedList.remove`. A
2. The worst case of `Heap.remove`. B
3. The worst case of `ArrayList.remove`. C
4. The best case of `BST.insert`. A
5. The worst case of `ArrayQueue.enqueue`. A
6. The best case of `AVL.find`. A
7. The worst case of `BST.remove`. C
8. The worst case of heap sort. D
9. The best case of heap sort. D
10. The best case of binary search. A
11. The worst case of `ArrayStack.push`. A
12. The worst case of `ArrayQueue.serve`. A

(c) (8 points) Choose the most appropriate data structure for each of the following tasks.

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

1. An algorithm reads an unknown number of integers from a file, then goes through the numbers and removes any number that is equal to the sum of the one before it and the one after it. C
2. An algorithm takes a list of n points $(x_i, y_i), i = 1 \ldots n$ and returns the k nearest points to a given point (x_0, y_0). K
3. An algorithm takes as input a list of purchases in the form $(UserID, ItemID, Price)$ and computes the total amount spent by every user. I
4. An algorithm reads a list of facebook friendship relations in the form $(User_i, User_j)$ and finds out if a post made by a user can reach another user through a sequence of shares (with friends). L

Question 2 ..... 14 points

We can represent the path from the root to any node in a non empty binary tree using a string containing the letters 'L' and 'R'. The letter 'L' indicates going left, whereas 'R' indicates going right.

Example 1. *In this tree, the empty string corresponds to the root 'A', "LL" corresponds to 'D', "LR" corresponds to 'E' and "RR" corresponds to 'F'.*

1. Write the method **private static void** `concat(char c, List<String> l)`, member of `BT` which concatenates the character `c` at the start of every string in `l`. If the list is empty, nothing happens.

```
1  private static void concat(char c, List<String> l) {
2    if (...) {
3      ...;
4      while (...) {
5        ...;
6        ...;
7      }
8      ...;
9    }
10 }
```

D • Line 2:

- (A) **if** `(!l.first()){`
- (B) **if** `(l == null){`
- (C) **if** `(!l.full()){`
- (D) **if** `(!l.empty()){`
- (E) None

Return: E • Line 3:

- (A) `current = head;`
- (B) `l.current = head;`
- (C) `l.findNext();`
- (D) `l.findLast();`
- (E) None

D • Line 4:

- (A) **while** `(!l.first()){`
- (B) **while** `(l.next != null){`
- (C) **while** `(l.last()){`
- (D) **while** `(!l.last()){`
- (E) None

B • Line 5:

- (A) `l.update(c);`
- (B) `l.update(c + l.retrieve());`
- (C) `l.update(l.retrieve()+ c);`

(D) `l.update(l.retrieve());`

(E) None

A • Line 6:

(A) `l.findNext();`

(B) `findNext();`

(C) `current = current.next;`

(D) `l.findFirst();`

(E) None

A • Line 8:

(A) `l.update(c + l.retrieve());`

(B) `l.update(l.retrieve()+ c);`

(C) `l.update(c);`

(D) `l.update(l.retrieve());`

(E) None

2. Write the method **public** `T get(String path)`, member of `BT`, which returns the data of the node indicated by `path`. Assume that the tree is not empty and that `path` is valid.

```
1  public T get(String path) {
2    BTNode<T> p = ...;
3    for (...) {
4      if (...)
5        ...;
6      else
7        ...;
8    }
9    return ...;
10 }
```

C • Line 2:

(A) `BTNode<T> p = null;`

(B) `BTNode<T> p = root.left;`

(C) `BTNode<T> p = root;`

(D) `BTNode<T> p = root.right;`

(E) None

D • Line 3:

(A) **for**(**int** `i=0; i<size; i++){`

(B) **for**(**int** `i=0; i<path.length(); i--){`

(C) **for**(**int** `i=path.length()-1; i>=0; i--){`

(D) **for**(**int** `i=0; i<path.length(); i++){`

(E) None

D • Line 4:

(A) **if** `(p.left != null)`

(B) **if** `(p.data == 'L')`

(C) **if** `(p.right != null)`

(D) **if** `(path.charAt(i)= 'L')`

(E) None

A • Line 5:

(A) `p = p.left;`

(B) **return** `p.data;`

(C) `p = root;`

(D) `p = p.parent;`

(E) None

B • Line 7:

(A) `p = p.parent;`

(B) `p = p.right;`

(C) **return** `p.data;`

(D) `p = root;`

(E) None

D • Line 9:

(A) **return** `p.left.data;`

(B) **return** `p.right.data;`

(C) **return** `p;`

(D) **return** `p.data;`

(E) None

3. Write the method **public** `List<String> leafPaths()`, member of `BT`, which returns the paths to all leaf nodes as strings. Use the method **private static void** `concat(char c, List<String> l)` above. If the tree is

empty, empty list is returned. Note that in line 16 below, we are calling the method `concatLists(l1, l2)` which returns the concatenation of the two lists `l1` and `l2`.

```
1  public List<String> leafPaths() {
2    return ...;
3  }
4  private List<String> rf(BTNode<T> t) {
5    List<T> l = new LinkedList<String>();
6    if (...)
7      ...;
8    if (...) {
9      ...;
10     ...;
11   }
12   List<String> l1 = ...;
13   ...;
14   List<String> l2 = ...;
15   ...;
16   return concatLists(l1, l2);
17 }
```

D • Line 2:

- (A) `return rf(t);`
- (B) `return leafPaths(root);`
- (C) `return rf(root.left)+rf(root.right);`
- (D) `return rf(root);`
- (E) None

D • Line 6:

- (A) `if (t.data == 'R')`
- (B) `if (t != null)`
- (C) `if (t.data == 'L')`
- (D) `if (t == null)`
- (E) None

D • Line 7:

- (A) `return l.retrieve();`
- (B) `return 'L';`
- (C) `return root;`
- (D) `return l;`
- (E) None

A • Line 8:

- (A) `if (t.left == null && t.right == null){`
- (B) `if (t.left != null && t.right != null){`
- (C) `if (t.left != null || t.right != null){`
- (D) `if (t.left == null || t.right == null){`
- (E) None

D • Line 9:

- (A) `l.update("");`
- (B) `l.findFirst();`
- (C) `l.insert("L");`
- (D) `l.insert("");`
- (E) None

A • Line 10:

- (A) `return l;`
- (B) `l.insert("R");`
- (C) `l.findFirst();`
- (D) `l.findNext();`
- (E) None

C • Line 12:

- (A) `List<String> l1 = rf(root.left);`
- (B) `List<String> l1 = rf(t);`
- (C) `List<String> l1 = rf(t.left);`
- (D) `List<String> l1 = rf(left);`
- (E) None

C • Line 13:

- (A) `concat('L', l);`
- (B) `concat('L', new LinkedList<String>());`
- (C) `concat('L', l1);`
- (D) `concat('L', rf(t));`
- (E) None

C • Line 14:

(A) `List<String> l2 = rf(right);`

(B) `List<String> l2 = rf(t);`

(C) `List<String> l2 = rf(t.right);`

(D) `List<String> l2 = rf(root.right);`

(E) None

C • Line 15:

(A) `concat('R', rf(t));`

(B) `concat('R', new LinkedList<String>());`

(C) `concat('R', l2);`

(D) `concat('R', l);`

(E) None

Question 3 ..... 12 points

(a) Write the method **private int** `f(BTNode<T> t, T e, int k)`, member of `BT`, which returns the number of nodes in the level `k` of the subtree `t` having data equal to `e`. The root of the subtree (that is `t`) is at level 0.

```
1 private int f(BTNode<T> t, T e, int k) {
2   if(...)
3     return ...;
4   if (...)
5     return ...;
6   return ...;
7 }
```

C • Line 2:

(A) **if** `(t.data == e)`

(B) **if** `(t != null)`

(C) **if** `(t == null)`

(D) **if** `(e.equals(t.data))`

(E) None

A • Line 3:

(A) **return** `0;`

(B) **return** `k;`

(C) **return** `1;`

(D) **return** `f(t.left,e,k)&&f(t.right,e,k);`

(E) None

B • Line 4:

(A) **if** `(!e.equals(t.data))`

(B) **if** `(k == 0 && e.equals(t.data))`

(C) **if** `(e.equals(t.data))`

(D) **if** `(k == 1 && e.equals(t.data))`

(E) None

C • Line 5:

(A) **return** `f(t.left,e,k)+f(t.right,e,k);`

(B) **return** `0;`

(C) **return** `1;`

(D) **return** `1+f(t.left,e,k)+f(t.right,e,k);`

(E) None

B • Line 6:

(A) **return** `f(t.left,e,k)+f(t.right,e,k);`

(B) **return** `f(t.left,e,k-1)+f(t.right,e,k-1);`

(C) **return** `f(t,e,k+1)+f(t,e,k+1);`

(D) **return** `f(t.left,e,k+1)+f(t.right,e,k+1);`

(E) None

(b) Repeat the same questions as above, but this time as a user.

```
public static <T> int f(BT<T> b, T e, int k) {
  if (...)
    ...;
  ...;
  ...;
}
private static <T> int rf(BT<T> b, T e, int k) {
```

```
8   if (...)
9     return ...;
10  int n=0;
11  if (...) {
12    ...;
13    ...;
14  }
15  if (...) {
16    ...;
17    ...;
18  }
19 }
```

B • Line 2:

(A) `if (b.full())`

(B) `if (b.empty())`

(C) `if (!b.empty())`

(D) `if (e.equals(b.retrieve()))`

(E) None

C • Line 3: (A) `return 1;`

(B) `return rf(b,e,k);`

(C) `return 0;`

(D) `return k;`

(E) None

C • Line 4:

(A) `return rf(b,e,k);`

(B) `return e.equals(b.retrieve());`

(C) `b.find(relative.Root);`

(D) `b.find(relative.Parent);`

(E) None

C • Line 5:

(A) `return rf(b, e, k - 1);`

(B) `return rf(b,e,k+1);`

(C) `return rf(b,e,k);`

(D) `return rf(b,e,0);`

(E) None

D • Line 8:

(A) `if (k==1)`

(B) `if (e.equals(b.retrieve()))`

(C) `if (k==1 && e.equals(b.retrieve()))`

(D) `if (k==0 && e.equals(b.retrieve()))`

(E) None

C • Line 9:

(A) `return 0;`

(B) `return 1+rf(b.left,e,k)+rf(b.right,e,k);`

(C) `return 1;`

(D) `return e.equals(b.retrieve());`

(E) None

B • Line 11:

(A) `if (b.find(Relative.Parent)){`

(B) `if (b.find(Relative.LeftChild)){`

(C) `if (b.find(Relative.Root)){`

(D) `if (b.left != null){`

(E) None

C • Line 12:

(A) `n-=rf(b.left,e,k);`

(B) `n-=rf(b,e,k);`

(C) `n+=rf(b,e,k-1);`

(D) `n+=rf(b,e,k+1);`

(E) None

B • Line 13:

(A) `b.find(Relative.RightChild);`

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

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

(D) `return n+1;`

(E) None

B • Line 15:

(A) `if (b.right != null){`

(B) `if (b.find(Relative.RightChild)){`

(C) `if (b.find(Relative.Parent)){`

(D) `if (b.find(Relative.Root)){`

(E) None

D • Line 16:

(A) `n+=rf(b,e,k+1);`

(B) `n-=rf(b,e,k);`

(C) `n-=rf(b.right,e,k);`

(D) `n+=rf(b,e,k-1);`

(E) None

A • Line 17:

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

(B) `b.find(Relative.LeftChild);`

(C) `return n+1;`

(D) `b.find(Relative.Root);`

(E) None

Question 4 ..... 12 points

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

A 1. What is the number of nodes in the last level of a heap containing 1024 nodes?

(A) 1. (B) 2. (C) 3. (D) 4. (E) 5.

A 2. The best case run time for an insert in a heap represented as array is:

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

B 3. The worst case run time for a remove in a heap represented as array is:

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

E 4. Suppose all the keys in a heap have the same value. Then:

(A) The keys will be served in order of arrival (FIFO). (B) The keys will be served in reverse order of arrival (LIFO). (C) The keys will be served in pre-order. (D) The keys will be served in in-order. (E) None.

(b) (8 points) Consider the following heap represented as an array: 3, 5, 6, 9, 7, 8, 10, 15, 11, 12. Choose the correct answer for every operation (all operations are done on the above heap).

B 1. Heap after inserting 4:

(A) 3, 5, 6, 9, 7, 8, 10, 15, 11, 12, 4 (B) 3, 4, 6, 9, 5, 8, 10, 15, 11, 12, 7 (C) 3, 6, 4, 9, 5, 8, 10, 15, 11, 12, 7 (D) 3, 4, 5, 6, 9 7, 8, 10, 15, 11, 12 (E) None

D 2. Heap after inserting 2 then 13: (A) 2, 3, 6, 9, 5, 8, 10, 11, 12, 7, 13, 15 (B) 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15 (C) 2, 3, 6, 9, 5, 8, 10, 11, 12, 7, 15, 13 (D) 2, 3, 6, 9, 5, 8, 10, 15, 11, 12, 7, 13 (E) None

C 3. Heap after deleting one key: (A) 5, 6, 9, 7, 8, 10, 15, 11, 12 (B) 5, 7, 6, 9, 8, 10, 15, 11, 12 (C) 5, 7, 6, 9, 12, 8, 10, 15, 11 (D) 5, 6, 7, 8, 9, 10, 11, 12, 15 (E) None

A 4. Heap after deleting two keys: (A) 6, 7, 8, 9, 12, 11, 10, 15 (B) 6, 7, 8, 9, 10, 11, 12, 15 (C) 6, 7, 9, 12, 8, 10, 15, 11 (D) 6, 7, 9, 12, 8, 10, 11, 15 (E) None

Question 5 ..... 12 points

(a) (4 points)

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:

C 1. The maximum height of an AVL tree with 4 nodes is:

(A) 1. (B) 2. (C) 3. (D) 4. (E) 5.

A 2. The cost of one rotation in an AVL tree is:

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

min = 0
Max = 2

B 3. The maximum number of rotations caused by an insert in an AVL tree with n nodes and height h is (a single rotation is counted 1; a double rotation is counted 2):

(A) 1. (B) 2. (C) h. (D) $2h$ (E) n.

B 4. The worst case height of an AVL tree with n nodes is:

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

(b) (8 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):

B 1. After inserting the key 17 in the AVL

, the tree becomes:

(A)

(B)

(C)

(D)

(E) None

D 2. After inserting the key 20 in the AVL

, the tree becomes:

(A)

(B)

(C)

(D)

(E) None

B 3. After deleting the key 11 from the AVL

, the tree becomes:

(A)

(B)

(C)

(D)

(E) None

C 4. After deleting the key 30 from the AVL

, the tree becomes:

(A)

(B)

(C)

(D)

(E) None

Question 6 ..... 12 points

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

C 1. The insert operation in a hash table using linear rehashing has a worst case run time:

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

it should be Prime # D 2. You want to store at most 16 keys in hash table using the % hash function. Choose the most appropriate table size:

(A) 14 (B) 15 (C) 16 (D) 17 (E) 18

36+1=37 C 3. How many keys can a hash table that uses folding on a single digit store if the key contains 4 digits?

(A) 4 (B) 36 (C) 37 (D) 39 (E) 40

E 4. Consider the following hash function: select the two rightmost digits then apply % 11 on the corresponding number. Which of the following couples of keys cause a collision?

(A) 3848 and 4756 (B) 3973 and 1258 (C) 162 and 35476 (D) All of the above. (E) None of the above.

(b) (8 points) Use the hash function $H(key) = key\%5$ to store the sequence of keys $22, 15, 12, 27, 18$ in a hash table of size 5. Use the following collision resolution strategies:

1. Linear rehashing (c=1). Fill in the following table:

Key	22	15	12	27	18
Position	2	0	3	4	1
Number of probes	1	1	2	3	4

2. External chaining. Fill in the following table:

Key	22	15	12	27	18
Index of the list	2	0	2	2	3

3. Coalesced chaining with cellar size 2 and address region size 5. Fill in the following table (put -1 if there is no next element):

Key	22	15	12	27	18
Position	2	0	6	5	3
Index of next element	6	-1	5	-1	-1

Try :)

Question 7 ..... 12 points

(a) (4 points)

Remark 2. *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. Recall also that 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:

B 1. The leaves of a B+ tree with $l = 7$ can contain the following number of data elements:

(A) 3 to 7 elements (B) 4 to 7 elements (C) 5 to 7 elements (D) 0 to 7 elements (E) None

D 2. In a B+ tree with $m = 4$, $l = 3$, and height h, the worst number of comparisons required to find a key is:

(A) h (B) $4h$ (C) $\log h$ (D) $3h$ (E) None

C 3. The maximum number of data elements in a B+ tree with $m = l = 3$ and height 2 is:

(A) 3 (B) 6 (C) 9 (D) 12 (E) None

A 4. The height of a B+ tree containing n different keys is:

(A) $O(\log n)$ (B) $O(n)$ (C) $O(n \log n)$ (D) $O(n^2)$ (E) None

(b) (8 points) Choose the correct result in each of the following cases (when possible, always borrow and transfer to the left):

C 1. After inserting the key 12 to the B+ tree

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

(A)

(B)

(C)

(D)

(E) None

5,7 E 2. After inserting the key 4 to the B+ tree

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

(A)

(B)

(C)

(D)

(E) None

A 3. After deleting the key 1 from the B+ tree

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

(A)

(B)

(C)

(D)

(E) None

D 4. After deleting the key 1 from the B+ tree

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

(A)

(B)

(C)

(D)

(E) None

Question 8 ..... 6 points

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

A 1. If you apply DFS on a binary tree, the nodes will be visited in the same order as:

(A) Preorder. (B) Inorder. (C) Postorder. (D) BFS. (E) None.

D 2. You apply BFS from a given node and you find out that **not** all nodes were visited. This means:

(A) Your BFS implementation has a bug. (B) Some nodes have no edges. (C) The graph contains cycles. (D) The graph is disconnected. (E) None.

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

a	$\to b \to c$
b	$\to a \to c \to e \to d$
c	$\to a \to b \to d$
d	$\to b \to c \to e \to f$
e	$\to b \to d \to f$
f	$\to e \to d$

1. Which of the following sequences are paths in this graph? Answer by T (true) or F (false).
   (a) (a, c, e, f) F
   (b) (a, c, b, d, f, e) T
   (c) (a, b, e, c) F
   (d) (f, e, b, d) T
2. Answer by T (true) or F (false).
   (a) The graph is connected. T
   (b) The number of edges in the graph is 8. F
   (c) (b, e, f, b) i a cycle. F
   (d) The number of 1s in the adjacency matrix of this graph is 18. T

B 3. The BFS traversal of this graph starting from a is (insert neighbors in the data structure in increasing alphabetic order):

(A) a, b, d, c, e, f. (B) a, b, c, d, e, f.
(C) a, c, d, f, e, b. (D) a, b, f, c, d, e.
(E) a, b, c, f, e, d.

D 4. The DFS traversal of this graph starting from a is (insert neighbors in the data structure in increasing alphabetic order):

(A) a, b, d, c, f, e. (B) a, b, d, c, e, f.
(C) a, b, c, d, e, f. (D) a, c, d, f, e, b.
(E) a, b, f, c, d, e.

King Saud University	College of Computer and Information Sciences Department of Computer Science
Data Structures CSC 212	**Final Exam - Fall 2019**
Date: 21/12/2019	Duration: 3 hours

Guidelines: No calculators or any other electronic devices are allowed in this exam.

Student ID:	Name:
Section:	Instructor:

1.1	1.2	2	3	4	5	6	7	8	Total

Question 1 ..... 14 points

(a) (6 points) Choose the run time from A to D for each of the following cases:

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

1. The worst case of `DoubleLinkedList.remove`. A
2. The worst case of `LinkedPQ.enqueue`. C
3. The worst case of `BST.insert`. C
4. The best case of `DoubleLinkedList.remove`. A
5. The best case of `Heap.insert`. A
6. The worst case of `ArrayList.findNext`. A
7. The best case of `LinkedPQ.enqueue`. A
8. The worst case of `AVL.find`. B
9. The worst case of `ArrayStack.pop`. A
10. The worst case of `Heap.insert`. B
11. The worst case of `Heap sort`. D
12. The best case of `Heap.remove`. A

(b) (8 points) Choose the most appropriate data structure for each of the following tasks.

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

مومتأكده

1. An algorithm reads an unknown number of integers from a file, then depending on an input from the user computes one of the following: sum, product, min, max or average of the numbers. C
2. An algorithm reads the list of flights operated by an airline company in the form $(City_i, City_j)$, meaning there is a flight between the two cities. The algorithm must check whether its is possible to fly from a given city to another using only the flights of this company. L
3. An algorithm spell-checks the user input by comparing the input text to a set of pre-stored words. I
4. A hospital wants to manage the waiting list in the emergency service. Cases with the same level of severity are treated according to the order of arrival. E

Scanned with CamScanner

X Question 2 ..... 12 points

(a) (6 points) Fill each entry of the table below by T (for true) or F (for false).

	Time (worst case): $O(n \log n)$	Comparison-based	In-place	Stable
Merge sort				
Quick sort				
Radix sort				

(b) (6 points) Consider the following array where keys are integer and data is of type string: $A = \{(3, B), (5, D), (3, A), (2, E), (7, E), (5, B), (1, F)\}$. We want to sort this array in increasing order. Choose the result produced by the given algorithm from the following options:

Ⓐ $\{(1, F), (2, E), (3, B), (3, A), (5, D), (5, B), (7, E)\}$

Ⓑ $\{(1, F), (2, E), (3, A), (3, B), (5, B), (5, D), (7, E)\}$

Ⓒ $\{(1, F), (2, E), (3, A), (3, B), (5, D), (5, B), (7, E)\}$

Ⓓ $\{(1, F), (2, E), (3, B), (3, A), (5, B), (5, D), (7, E)\}$

Ⓔ None of the above.

1. Heap sort (in sift-down, swap left if children are equal): Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2. Bubble sort: Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

Question 3 ..... 12 points

(a) Write the method **private boolean** `f(BTNode<T> t, T e, int k)`, member of `BT`, which return true if `e` appears in `t` at a depth that is equal or greater than `k` (assume that `t` is at depth 0).

```
1 private boolean f(BTNode<T> t, T e, int k) {
2   if (...)
3     ...;
4   if (...)
5     ...;
6   return ...;
7 }
```

B • Line 2:

Ⓐ `if (e.equals(t.data))`

Ⓑ `if (t==null)`

Ⓒ `if (t.left==null && t.right==null)`

Ⓓ `if (k > 0)`

Ⓔ None

C • Line 3:

Ⓐ `return true;`

Ⓑ `return k;`

Ⓒ `return false;`

Ⓓ `return k == e.data;`

Ⓔ None

B • Line 4:

Ⓐ `if (k>0 || e.equald(t.data))`

Ⓑ `if (k<=0 && e.equald(t.data))`

Ⓒ `if (k>0 && e.equald(t.data))`

Ⓓ `if (k<0 && e.equald(t.data))`

Ⓔ None

Scanned with CamScanner

D • Line 5:

(A) `return k>0;`

(B) `return false;`

(C) `return e.equald(t.data);`

(D) `return true;`

(E) None

D • Line 6:

(A) `return f(t.left,e,k-1)&&f(t.right,e,k-1);`

(B) `return f(t.left,e,k)&&f(t.right,e,k);`

(C) `return f(t.left,e,k)||f(t.right,e,k);`

(D) `return f(t.left,e,k-1)||f(t.right,e,k-1);`

(E) None

(b) Repeat the same questions as above, but this time as a user.

```
1  public static <T> boolean f(BT<T> bt, T e, int k) {
2    if (...)
3      return ...;
4    ...;
5    return ...;
6  }
7  private static <T> int rf(BT<T> bt, T e, int k) {
8    if (...)
9      return ...;
10   if (...) {
11     if (...)
12       return ...;
13     ...;
14   }
15   if (...) {
16     if (...)
17       return ...;
18     ...;
19   }
20   return ...;
21 }
```

A • Line 2:

(A) `if (bt.empty())`

(B) `if (k == 0)`

(C) `if (bt.full())`

(D) `if (!bt.full())`

(E) None

A • Line 3:

(A) `return false;`

(B) `return true;`

(C) `return e.equals(bt.retrieve());`

(D) `return k == 0;`

(E) None

E • Line 4:

(A) `bt.find(relative.LeftChild);`

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

(C) `bt.find(relative.RightChild);`

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

(E) None

A • Line 5:

(A) `return rf(bt,e,k);`

(B) `return rf(bt.left,e,k)||rf(bt.right,e,k);`

(C) `return rf(bt,e,k-1);`

(D) `return rf(bt,e,k+1);`

(E) None

B • Line 8:

(A) `if (k<=0)`

(B) `if (k<0 && e.equals(bt.retrieve()))`

(C) `if (k<=0 && e.equals(bt.retrieve()))`

(D) `if (k==0 || e.equals(bt.retrieve()))`

(E) None

A • Line 9:

(A) `return true;`

Scanned with CamScanner

(B) `return k < 0;`

(C) `return e.equals(bt.retrieve());`

(D) `return false;`

(E) None

C • Line 10:

(A) `if (bt.find(Relative.Parent)){`

(B) `if (bt.find(Relative.Root)){`

(C) `if (bt.find(Relative.LeftChild)){`

(D) `if (bt.find(Relative.LeftChild)!=nulll){`

(E) None

A • Line 11:

(A) `if (rf(bt,e,k-1))`

(B) `if (rf(bt,e,k))`

(C) `if (rf(bt,e,k+1))`

(D) `if (rf(bt,e,2*k))`

(E) None

B • Line 12:

(A) `return k>0;`

(B) `return true;`

(C) `return k<=0;`

(D) `return false;`

(E) None

B • Line 13:

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

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

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

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

(E) None

A • Line 15:

(A) `if (bt.find(Relative.RightChild)){`

(B) `if (bt.find(Relative.Root)){`

(C) `if (bt.find(Relative.RightChild)!=nulll){`

(D) `if (bt.find(Relative.Parent)){`

(E) None

C • Line 16:

(A) `if (rf(bt,e,2*k+1))`

(B) `if (rf(bt,e,k+1))`

(C) `if (rf(bt,e,k-1))`

(D) `if (rf(bt,e,k))`

(E) None

C • Line 17:

(A) `return k<=0;`

(B) `return false;`

(C) `return true;`

(D) `return k>0;`

(E) None

C • Line 18:

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

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

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

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

(E) None

B • Line 20:

(A) `return k<=0;`

(B) `return false;`

(C) `return k>=0;`

(D) `return k==0;`

(E) None

Question 4 ..... 14 points

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

E 1. In a min heap:

(A) All keys at level k are smaller than all keys in level $k+1$. (B) The largest key is always at the last level. (C) The key of the left child of any node is smaller than the key of its right child. (D) All of the above. (E) None of the above.

Scanned with CamScanner

C 2. The worst case run time for bottom-up heap construction is:

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

(b) (10 points) Consider the following heap represented as an array: 20, 18, 12, 10, 11, 5, 2, 6, 4, 3. Choose the correct answer for every operation (all operations are done on the above heap).

A 1. Heap after inserting 19:

Ⓐ 20, 19, 12, 10, 18, 5, 2, 6, 4, 3, 11 Ⓑ 20, 19, 12, 11, 18, 5, 2, 6 ,4, 11, 3 Ⓒ 20, 18, 12, 10, 11, 5, 2, 6, 4, 3, 19 Ⓓ 20, 18, 12, 10, 19, 5, 2, 6, 4, 3,11 Ⓔ None

C 2. Heap after inserting 1 **then** 23: Ⓐ 20,18, 23, 10, 11, 12, 2, 6, 4, 3, 1, 5 Ⓑ 20, 18, 12, 10, 11, 5, 2, 6, 4, 3, 1, 23 Ⓒ 23, 18, 20, 10, 11, 12, 2, 6, 4, 3, 1, 5 Ⓓ 23, 20, 18, 10, 11, 12, 2, 6, 4, 3, 1, 5 Ⓔ None

D 3. Heap after deleting one key: Ⓐ 20, 18, 12, 10, 11, 5, 2, 6, 4 Ⓑ 3, 18, 12, 10, 11, 5, 2, 6, 4 Ⓒ 12, 18, 5, 10, 11, 3, 2, 6, 4 Ⓓ 18, 11, 12, 10, 3, 5, 2, 6, 4 Ⓔ None

E 4. Heap after deleting two keys: Ⓐ 4, 18, 12, 10, 11, 5, 2, 6 Ⓑ 18, 11, 12, 10, 4, 5, 2, 6 Ⓒ 12, 4, 5, 10, 11, 3, 2, 6 Ⓓ 12, 11, 4, 10, 3, 5, 2, 6 Ⓔ None

Question 5 ..... 14 points

(a) (4 points)

Remark 1. *In what follows the depth of tree is the number of levels in the tree from the root to a leaf (counting number of nodes not edges). Hence, tree with 1 node has depth 0.*

Choose the most appropriate answer:

C 1. The maximum depth of an AVL tree with 8 nodes is:

Ⓐ 1. Ⓑ 2. Ⓒ 3. Ⓓ 4. Ⓔ 5.

A 2. The minimum number of rotations caused by an insert in an AVL tree with n nodes and height h is (a single rotation is counted 1; a double rotation is counted 2):

Ⓐ 0. Ⓑ 1. Ⓒ 2. Ⓓ h Ⓔ n.

(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):

A 1. After inserting the key 9 in the AVL

, the tree becomes:

Ⓐ

Ⓑ

Ⓒ

Ⓓ

Ⓔ None

C 2. After inserting the key 1 in the AVL

, the tree becomes:

Scanned with CamScanner

A 3. After deleting the key 2 from the AVL , the tree becomes:

A 4. After deleting the key 20 from the AVL , the tree becomes:

Question 6 ..... 14 points

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

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

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

E 2. How many keys can a hash table that uses folding on a single digit store if the key contains 3 digits?

Ⓐ 4 Ⓑ 8 Ⓒ 26 Ⓓ 27 Ⓔ 28

(b) (10 points) Use the hash function $H(key) = key\%5$ to store the sequence of keys 25, 13, 14, 23, 16 in a hash table of size 5. Use the following collision resolution strategies:

1. Linear rehashing (c=1):

Key	25	13	14	23	16
Position	0	3	4	1	2
Number of probes	1	1	1	4	2

2. External chaining:

Key	25	13	14	23	16
Index of the list	0	3	4	3	1

3. Coalesced chaining with cellar size 2 and address region size 5 (put -1 if there is no next element):

Key	25	13	14	23	16
Position	0	3	4	6	1
Index of next element	-1	6	-1	-1	-1

Scanned with CamScanner

Question 7 ..... 14 points

(a) (4 points)

Remark 2. *In what follows, a tree with 1 node has depth 1 and height 1. Recall also that 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:

C 1. The minimum number of children of a root node in a B+ tree is:

Ⓐ 0 Ⓑ 1 Ⓒ 2 Ⓓ $\lceil \frac{m}{2} \rceil$ Ⓔ None

A 2. The maximum number of data elements in a B+ tree with $m = l$ and height h is :

Ⓐ m^h Ⓑ 2^h Ⓒ h^m Ⓓ $h + m$ Ⓔ None

(b) (10 points) Choose the correct result in each of the following cases (when possible, always borrow and transfer to the left):

D 1. After inserting the key 4 in the B+ tree , the **root** becomes:

Ⓐ [4 7] Ⓑ [5 8] Ⓒ [4] Ⓓ [5] Ⓔ None

B 2. After inserting the key 10 in the B+ tree , the **root** becomes:

Ⓐ [5 10] Ⓑ [5 8] Ⓒ [6] Ⓓ [6 10] Ⓔ None

6,8 E 3. After deleting the key 2 from the B+ tree , the **root** becomes:

Ⓐ [8] Ⓑ [5 8] Ⓒ [6] Ⓓ [6 7] Ⓔ None

A 4. After deleting the key 5 from the B+ tree , the **root** becomes:

Ⓐ [8] Ⓑ [5 8] Ⓒ [6] Ⓓ [2] Ⓔ None

Scanned with CamScanner

Question 8 ..... 6 points

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

B 1. Adjacency matrix takes $O(n^2)$ memory space, when is it appropriate to use it instead of adjacency list:

Ⓐ The graph is sparse. Ⓑ The graph is dense. Ⓒ The graph is unweighted Ⓓ The number of edges is less than the number of nodes. Ⓔ None.

C 2. You apply DFS from a given node and you find out that all nodes were visited. This means:

Ⓐ The graph is directed. Ⓑ Some nodes have no edges. Ⓒ The graph is connected. Ⓓ The graph contains cycles. Ⓔ None.

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

	A	B	C	D	E	F
A		3	3			5
B	3		1	5		
C	3	1				
D		5				2
E						
F	5			2		

1. Which of the following sequences are simple paths in this graph? Answer by T (true) or F (false).
   (a) $(C, B, D, E,)$ F
   (b) (C, B, A) T
   (c) (C, A, F, D, B) T
   (d) (B, A, F, D, B, C) F
2. Answer by T (true) or F (false).
   (a) The graph is connected. F
   (b) The number of edges in the graph is 6. T
   (c) (A, B, D, F, A) is a cycle. T
   (d) The shortest path from F to B is (F, A, B). T (لو مانحتبر ال weight)

C 3. The BFS traversal of this graph starting from A is (insert neighbors in the data structure in increasing alphabetic order):

Ⓐ A, D, B, C, F. Ⓑ A, B, C, D, F. Ⓒ A, B, C, F, D. Ⓓ A, F, B, C, D. Ⓔ A, C, B, F, D.

A 4. The DFS traversal of this graph starting from A is (insert neighbors in the data structure in increasing alphabetic order):

Ⓐ A, F, D, C, B. Ⓑ A, F, C, D, B. Ⓒ A, B, C, F, D. Ⓓ A, F, B, C, D. Ⓔ A, C, B, F, D.

Scanned with CamScanner

A **QUESTION 1**

2 points Saved

Choose the most appropriate data structure:
A browsing history in a web browser that allows users to click backward and forward buttons to navigate through previously visited webpages.

Double linked list Hash table heap AVL tree linked priority queue

---

C **QUESTION 2**

2 points Saved

Implement Undo feature of a Text Editor

AVL tree linked queue array stack linked list binary tree

C **QUESTION 3**

2 points Saved

Call Center phone system that holds incoming calls in an order, until a service representative is free.

- array list
- array stack
- linked queue
- AVL tree
- linked list

B **QUESTION 4**

2 points Saved

Bibliography system that arranges references in alphabetical order of authors' surnames

linked list heap binary tree linked stack array list

C **QUESTION 5**

2 points Saved

Friend Suggestion System for Social Networks that connects between its users and the people, places and things they interact with online.

binary tree double linked list graph hash table AVL tree

D **QUESTION 6**

2 points Saved

A signing up authentication system that asks users to create a secure password. Then, the system checks if the submitted password is not within the password blacklist (the list of the most commonly used passwords that can easily be hacked e.g. myPssword123).

AVL tree array queue heap Hash table linked stack

QUESTION 7

1 points Save Answer

write the method ***sortList*** which receives a list of integers *l* and a number *n*. The method should return a new list containing all the integers in *l* that are greater than or equal to ***n*** first,then all the integers in ***l*** that are less than **n**.

Example

if List l: [1 5 8 4 3 6 7 2 9] and **n = 5**.

Then, the method should return the list: [5 8 6 7 9 1 4 3 2]

```
public static LinkedList<Integer> sortList(LinkedList<Integer> l, int n) {
  LinkedList<Integer> x = new LinkedList<Integer>();
   LinkedList<Integer> y = new LinkedList<Integer>();
  if (..............)
    System.out.println("List␣is␣empty");
  else {
    ................
    while (...........)
      {
      if(l.retrieve()>=n)
        x.insert(l.retrieve());
         else
            ....................
      ................
    }
    if(................)
      x.insert(l.retrieve());
      else
         y.insert(l.retrieve());
    if (..........)
      {
         ...............
       while (........)
         {
        x.insert(y.retrieve());
           ..........................
         }
        ..........................
      }
  }
  return .....;
}
```

B **Line 4:**

- A) `l.hasNext()`
- B) `l.empty()`
- C) `l.full()`
- D) `l.last()`
- E) None

D **QUESTION 8**

1 points Save Answer

Line 7:

- A) `findNext()`
;
- B) `x.findFirst();`
- C) `l.findNext();`
- D) `l.findFirst();`
- E) None

D

QUESTION 9

1 points Saved

Line 8:

A) `l.retrieve()>= n`

B) `l.retrieve()!= n`

C) `l.last()`

D) `!l.last()`

E) None

B

QUESTION 10

1 points Save Answer

Line 13:

A) `l.findNext();`

B) `y.insert(l.retrieve());`

C) `y.insert(x.retrieve());`

D) `l.retrieve();`

E) None

B QUESTION 11

1 points Saved

Line 14:

A) `x.findNext();`

B) `l.findNext();`

C) `x.remove();`

D) `l.remove();`

E) None

B QUESTION 12

1 points Save Answer

Line 16:

A) `l.retrieve()==n`

B) `l.retrieve()>= n`

C) `x.retrieve()>= n`

D) `l.retrieve()!= n`

E) None

D QUESTION 13

1 points Save Answer

Line 20:

- A) `!x.empty()`
- B) `x.retrieve()!= null`
- C) `y.empty()`
- D) `!y.empty()`
- E) None

B QUESTION 14

1 points Save Answer

Line 22:

- A) `x.findFirst();`
- B) `y.findFirst();`
- C) `l.findNext();`
- D) `l.findFirst();`
- E) None

QUESTION 15

1 points Save Answer

Line 23:

A) `l.last()`

B) `l.retrieve()!= null`

C) `!y.last()`

D) `!x.last()`

E) None

QUESTION 16

1 points Save Answer

Line 26:

A) `l.retrieve();`

B) `y.retrieve();`

C) `y.findNext();`

D) `l.findNext();`

E) None

D QUESTION 17

1 points Save Answer

Line 28:

- A) `y.insert(l.retrieve());`
- B) `x.insert(l.retrieve());`
- C) `y.insert(x.retrieve());`
- D) `x.insert(y.retrieve());`
- E) None

B QUESTION 18

1 points Save Answer

Line 31:

- A) `l;`
- B) `x;`
- C) `l.retrieve();`
- D) `y;`
- E) None

QUESTION 19

1 points Save Answer

Write the method **private boolean evenN(BTNode t, int n),** member of **BT**, which returns true if an even number appears in **t** at a depth that is equal or greater than **n** (assume that t is at depth 0)

```
1 private boolean evenN(BTNode<T> t, int n) {
2   if (............)
3     return ............;
4   if (............)
5     return ............;
6   return ............;
7 }
```

B **Line 2:**

- A) `if ( t.data%2==0)`
- B) `if (t == null)`
- C) `if (t.left==null && t.right==null)`
- D) `if (n > 0)`
- E) None

QUESTION 20

1 points Saved

C **Line 3:**

- A) `return true;`
- B) `return n;`
- C) `return false;`
- D) `return n == (t.data%2);`
- E) None

B QUESTION 21

1 points Saved

Line 4:

- A) `if (n>0 || (t.data%2==0))`
- B) `if (n <= 0 && t.data%2==0)`
- C) `if (n>0 && (t.data%2==0))`
- D) `if (n<0 && (t.data%2==0))`
- E) None

D QUESTION 22

1 points Save Answer

Line 5:

- A) `return n>0;`
- B) `return false;`
- C) `return (t.data%2==0);`
- D) `return true;`
- E) None

D QUESTION 23

1 points Save Answer

Line 6:

- A) `return evenN(t.left,n-1)&&evenN(t.right,n-1);`
- B) `return evenN(t.left,n)&&evenN(t.right,n);`
- C) `return evenN(t.left,n)||evenN(t.right,n);`
- D) `return evenN(t.left, n - 1)|| evenN(t.right, n - 1);`
- E) None

QUESTION 24

0.5 points Save Answer

Repeat the same question as above, but this time as a user.

```
public static <T> boolean evenN(BT<T> t, int n) {
  if (............)
    return ............;
  ............;
  return ............;
}
private static <T> boolean recEvenN(BT<T> t, int n) {
  if (............)
    return ............;
  if (............) {
    if (............)
      return ............;
    ............;
  }
  if (............) {
    if (............)
      return ............;
    ............;
  }
  return ............;
}
```

Line 2

- A) `if (t.empty())`
- B) `if (n == 0)`
- C) `if (t.full())`
- D) `if (!t.full())`
- E) None

QUESTION 25

0.5 points Save Answer

Line 3

- A) `return false;`
- B) `return true;`
- C) `return (t.retrieve()%2==0));`
- D) `return n == 0;`
- E) None

D QUESTION 26

1 points Save Answer

Line 4

- A) `t.find(Relative.LeftChild);`
- B) `t.find(Relative.Parent);`
- C) `t.find(Relative.RightChild);`
- D) `t.find(Relative.Root);`
- E) None

A QUESTION 27

1 points Save Answer

Line 5

- A) `return recEvenN(t, n);`
- B) `return recEvenN(t.left,n)||recEvenN(t.right,n);`
- C) `return recEvenN(t,n-1);`
- D) `return recEvenN(t,n+1);`
- E) None

C QUESTION 28

0.5 points Save Answer

Line 8

- A) `if (n<=0)`
- B) `if (n<0 && (t.retrieve()%2==0))`
- C) `if (n <= 0 && (t.retrieve()%2==0))`
- D) `if (n==0 || (t.retrieve()%2==0))`
- E) None

A QUESTION 29

0.5 points Save Answer

Line 9

- A) `return true;`
- B) `return n < 0;`
- C) `return t.retrieve()%2==0;`
- D) `return false;`
- E) None

C QUESTION 30

0.5 points Save Answer

Line 10

- A) `if (t.find(Relative.Parent)){`
- B) `if (t.find(Relative.Root)){`
- C) `if (t.find(Relative.LeftChild){`
- D) `if (t.find(Relative.LeftChild)!=nulll){`
- E) None

A QUESTION 31

0.5 points Save Answer

Line 11

- A) `if (recEvenN(t,n-1))`
- B) `if (recEvenN(t,n))`
- C) `if ((recEvenN(t,n+1))`
- D) `if ((recEvenN(t,2*n))`
- E) None

B QUESTION 32

0.5 points Save Answer

Line 12

- A) `return n>0;`
- B) `return true;`
- C) `return n<=0;`
- D) `return false;`
- E) None

B QUESTION 33

0.5 points Save Answer

Line 13

- A) `t.find(Relative.Root);`
- B) `t.find(Relative.Parent);`
- C) `t.find(Relative.LeftChild);`
- D) `t.find(Relative.RightChild);`
- E) None

A QUESTION 34

0.5 points Save Answer

Line 15

- A) `if (t.find(Relative.RightChild)){`
- B) `if (t.find(Relative.Root)){`
- C) `if (t.find(Relative.RightChild)!=nulll){`
- D) `if (t.find(Relative.Parent)){`
- E) None

C QUESTION 35

0.5 points Save Answer

Line 16

- A) `if (recEvenN(t,2*n+1))`
- B) `if (recEvenNrf(t,n+1))`
- C) `if (recEvenN(t,n-1))`
- D) `if (recEvenN(t,n))`
- E) None

QUESTION 36

0.5 points Save Answer

Line 17

- A) `return n<=0;`
- B) `return false;`
- C) `return true;`
- D) `return n>0;`
- E) None

QUESTION 37

0.5 points Save Answer

Line 18

- A) `t.find(Relative.RightChild);`
- B) `t.find(Relative.LeftChild);`
- C) `t.find(Relative.Parent);`
- D) `t.find(Relative.Root);`
- E) None

B QUESTION 38

1 points Save Answer

Line 20

- A) `return n<=0;`
- B) `return false;`
- C) `return n>=0;`
- D) `return n==0;`
- E) None

E **QUESTION 39**

3 points Save Answer

Choose the most appropriate answer
In a MIN heap:

- A) Any key at level k is smaller than or equal to all keys in level k + 1.
- B) The largest key isalways at the last level.
- C) The key of the left child of any node is smaller than the key of itsright child.
- D) All of the above.
- E) None of the above.

A **QUESTION 40**

3 points Save Answer

The best case run time for removing minimum value from a MIN heap is:

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

D **QUESTION 41**

2 points Save Answer

Consider the following heap represented as an array: **20, 30, 55, 50, 52**.
Choose the correct answer forevery operation (all operations are done on the above heap).
Heap after inserting **16**:

- A) 20, 30, 16, 50, 52, 55
- B) 20, 30, 55, 50, 52, 16
- C) 16, 20, 30,55, 50, 52
- D) 16, 30, 20, 50, 52, 55
- E) None

B **QUESTION 42**

2 points Save Answer

Heap after inserting **25**:

- A) 20, 30, 55, 50, 52, 25
- B) 20, 30, 25, 50, 52, 55
- C) 20, 25,30, 55, 50, 52
- D) 25, 20, 30, 55, 50, 52
- E) None

A **QUESTION 43**

2 points Save Answer

Heap after deleting one key:

- A) 20, 30, 55, 50
- B) 30, 50, 55, 52
- C) 20, 30, 50, 52
- D) 30, 52, 55, 50
- E) None

E **QUESTION 44**

2 points Save Answer

Heap after deleting two keys:

- A) 30, 50, 55
- B) 30, 50, 52
- C) 50, 55, 52
- D) 52, 50, 55
- E) None

C **QUESTION 45**

3 points Save Answer

Choose the most appropriate answer:
The maximum number of rotations caused by an insert in an AVL tree with ***n*** nodes and height ***h*** is :
(Note: a single rotation is counted 1; a double rotation is counted 2)

A) 0. B) 1. C) 2. D) ***h*** E) ***n***.

D **QUESTION 46**

3 points Save Answer

Consider the following tree :

If the balance of A is +2 and that of B is -1 .
What rotations must be done to balance the tree:

A) Single Left Rotation . B) Single Right Rotation. C) Double Right Left Rotation D) Double Left Right Rotation. E) No Rotation.

QUESTION 47

2 points Save Answer

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

○ A) ○ B) ○ C) ○ D) ○ E)

A **QUESTION 48**

2 points Save Answer

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

After inserting the key 20 in the AVL

, the tree becomes:

(A)

(B)

(C)

(D)

(E) None

○ A) ○ B) ○ C) ○ D) ○ E)

C **QUESTION 49**

2 points Save Answer

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

After deleting the key 12 from the AVL

, the tree becomes:

(A)

(B)

(C)

(D)

(E) None

○ A) ○ B) ○ C) ○ D) ○ E)

D **QUESTION 50**

2 points Save Answer

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

A) B) C) D) E)

C **QUESTION 51**

1 points Save Answer

Choose the most appropriate answer:
In the worst-case scenario, the time complexity of the Linear Rehashing technique is:

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

QUESTION 52

1 points Save Answer

Consider a company where an employee-ID is designed as follow: The first 3 digits reflect his department, the next 2 digits reflect his job-title, and the last 4 digits are randomly generated by the system.

The company can have at most 10000 employee.

Choose the **best** hash function from the following:

- A) H(key) = key mod 10000
- B) H(key) = (sum of the first 2 digits) *100
- C) H(key) =the result of folding the key at 2 digits
- D) All of the above.
- E) None of the above.

QUESTION 53

12 points Save Answer

Use the hash function **H(key) = key%5** to store the sequence of keys **10 , 13 , 18 , 2 , 3** in a hash table of size **5**.
Use the following collision resolution strategies:

1. Linear rehashing (**c=1**). Fill in the following table:

key	10	13	18	2	3
Position	0	3	4	2	1
Number of probes	1	1	2	1	4

2. External chaining. Fill in the following table:

key	10	13	18	2	3
Index of the list	0	3	3	2	3

3. Coalesced chaining with cellar size 2 and address region size 5 . Fill in the following table (put -1if there is no next element):

key	10	13	18	2	3
Position	0	3	6	2	5
Index of next element	-1	6	5	-1	-1

Try :)

C **QUESTION 54**

2 points Save Answer

Remark : In what follows, a tree with 1 node has depth 1 and height 1. Recall also that 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:

The minimum number of children of a root node in a B+ tree is:

○ A) 0 ○ B) 1 ○ C) 2 ○ D) $\lceil \frac{m}{2} \rceil$ ○ E) None

---

A **QUESTION 55**

2 points Save Answer

The maximum number of data elements in a B+ tree with **m = l** and height **h** is :

○ A) m^h ○ B) 2^h ○ C) h^m ○ D) $h + m$ ○ E) None

B **QUESTION 56**

2 points Save Answer

After inserting the key 9 in the B+ tree [1 | 5 | 8], the **root** of tree becomes:

(A) [1 |] (B) [8 |] (C) [9 |] (D) [5 | 9] (E) None

○ A) ○ B) ○ C) ○ D) ○ E)

B **QUESTION 57**

2 points Save Answer

After inserting the key 4 in the B+ tree

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

(A)

(B)

(C)

(D)

(E) None

○ A) ○ B) ○ C) ○ D) ○ E)

D **QUESTION 58**

2 points Save Answer

After inserting the key 10 in the B+ tree

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

(A)

(B)

(C)

(D)

(E) None

○ A) ○ B) ○ C) ○ D) ○ E)

E **QUESTION 59**

2 points Save Answer

After deleting the key 2 from the B+ tree

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

(A)

(B)

(C)

(D)

(E) None

○ A) ○ B) ○ C) ○ D) ○ E)

A||C **QUESTION 60**

2 points Save Answer

After deleting the key 5 from the B+ tree

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

(A)

(B)

(C)

(D)

(E) None

○ A) ○ B) ○ C) ○ D) ○ E)

B **QUESTION 61**

1 points Save Answer

Choose the most appropriate answer:
Adjacency list space requirement if e is the number of edges and n is the number of vertices is :

A) $O(n^2)$ B) O(e + n) C) O(n) D) O(log n) E) None.

C **QUESTION 62**

1 points Save Answer

The data structure requierd for BFS traversal on a graph is:

A) Tree. B) Array. C) Queue. D) Stack. E) None.

QUESTION 63

2 points Save Answer

Given the following graph adjacency matrix, answer the questions below.

	A	B	C	D	E	F
A			1		1	1
B			1	1	1	
C	1	1			1	
D		1			1	
E	1	1	1	1		
F	1					

Which of the following sequences are simplepaths in this graph? Answer by T (true) or F(false).

(a) (C , B , D , A , E) F

(b) (A , C , B) T

(c) (C , A , F , D , B) F

(d) (A , E , C , B , D) T

Answer by T (true) or F (false).

(a) The graph is connected. T

(b) The number of edges in the graph is 6. F

(c) There is a cycle in the graph. T

(d) The shortest path from F to B is (F , A , D ,B). F

C QUESTION 64

1 points Save Answer

The **BFS** traversal of this graph starting from **A** is :
(insert neighbors in the data structure in increasing alphabetic order):

- A) A , F , E , D , B , C.
- B) A , E , B , C , D , F.
- C) A , C , E , F , B , D.
- D) A , F , E , B , C , D.
- E) A , E , C , B , F , D.

A QUESTION 65

1 points Save Answer

The **DFS** traversal of this graph starting from **A** is :
(insert neighbors in the data structure in increasing alphabetic order):

- A) A , F , E , D , B , C.
- B) A , E , B , C , D , F.
- C) A , C , E , F , B , D.
- D) A , F , E , B , C , D.
- E) A , E , C , B , F , D.

FINAL DATA STRUCTURES (2020-S1)

QUESTION 1: Choose the most appropriate data structures

I. Adding orders in a restaurant by taking in customers' food requests. queue

II. Web server cache (it works by allowing users to add a new HTML page or go to a previous one or even going to the next). DLL

III. Tracing a packet traversal through nodes in a network topology. Graph

IV. Counting the frequency of words in a word document (any new word is deemed as frequency of 1) and its frequency is incremented each time it is found. AVL

V. Taking patient requests for covid19 vaccines, a patient can be healthy or non-healthy, requests are accepted based on their health history. Priority Queue

VI. The span Si of the stock's price on a given day i is defined as the maximum number of consecutive days just before the given day, for which the price of the stock on the current day is less than or equal to its price on the given day. For example, if an array of 7 days prices is given as {100, 80, 60, 70, 60, 75, 85}, then the span values for corresponding 7 days are {1, 1, 1, 2, 1, 4, 6}. Double Linked List

QUESTION 2: Write a method as a user for splitting a linked list (list) and return a new list that has been split at a given value (n). For example, calling split(linked list<integer> list ,int n) ; list = {1,2,3,4,5,6,7,8,9} and n = 6 returns the list {6,7,8,9}. Also, if the list is empty or the value has not been found display an appropriate message and return an empty list (the original list MUST not change).

```
public static LinkedList <Integer> split(LinkedList<Integer> list, int n)
{
...
}
```

QUESTION 3: Write the recursive BT member method by the following signature, boolean isPathSum (int k), Which is a function that returns true if one of the paths has the sum value of k, if its empty it has no path sum. For example, let bt be the following

so by calling isPathSum(18)
1+7+10 = true
and isPathSum(22) = false

```
public boolean isPathSum(int k)
{
...
}

public boolean rps(BTNode<Integer> t, int k)
{
..
}
```

Write the same method in recursive but now as a user

```
public static boolean isPathSum(BT<Integer> bt, int k)
{
...
}

public static boolean rps(BT<Integer> bt, int k)
{
...
}
```

SOME THEORETICAL QUESTIONS:

VII. What is the worst-case time complexity of retrieving a max value from a max-heap. O(1)

keys = parents
values/items = leaves

VIII. Given a b+ tree m = 6 what is the maximum number of keys at level 1 (root is at level 0).

level 0 →

level 1 →

(5x6) = 30

leaves can be 6 (6x6) = 36

IX. In an AVL tree what is the maximum number of nodes at level 2.

If root = 0 0 1 2 4 nodes! If root = 1 1 2 2 nodes!

X. In an AVL tree with 7 nodes what is the maximum depth.

Depth starts at 0, maximum depth = 3 (4 levels)

أحاول أطولها قدما اقدر
(must be balanced)

XI. What is the best-case complexity for insertion in coalesced chaining or external (I don't remember the type of rehashing) O(1)

XII. DFS uses what data structure. stack

QUESTION 5,6,7,8: AVL, B+, HASH, HEAP AND GRAPHS ARE EXACTLY THE SAME AS THE PREVIOUS EXAMS.

Q2:

```
public List<Integer> split(LinkedList<Integer> list, int n)
{
  List res = new LinkedList<Integer>();
  if(list.empty())
    s.o.p("empty List");
    return res;
  list.findFirst();
  int i=1;
  for( i=1; i< n && !list.last; i++)
    {
      list.findNext();
    }
  if(i == n){
    while(!list.last())
      res.insert(list.retrieve());
      list.findNext();
    res.insert(list.retrieve());
    return res;
  }
  else{
    s.o.p("not Found");
    return res;
  } }
```

i=1 ✓
i=2 ✓
i=3 ✓
i=4 ✓
i=5 ✓

الممسوحة ضوئيا بـ CamScanner

Question 3-a:-

```
public boolean ispathsum(int k){

    return ispathsum(root, k);
}

private boolean ispathsum(BTnode<Integer>p, int k)
{
    if(p == null)    return false;
    if(k - p.data == 0) return true;

    return ispathsum(p.left, k-p.data)
        || ispathsum(p.right, k-p.data);
}
```

20

20

ممكن يكتب بشكل آخر

```
if(k == p.data)    return true;
```

الممسوحة ضوئيا بـ CamScanner