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

(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 <code>LinkedList.remove</code> . ____ | 7. The worst case of <code>BST.remove</code> . ____ |
| 2. The worst case of <code>Heap.remove</code> . ____ | 8. The worst case of heap sort. ____ |
| 3. The worst case of <code>ArrayList.remove</code> . ____ | 9. The best case of heap sort. ____ |
| 4. The best case of <code>BST.insert</code> . ____ | 10. The best case of binary search. ____ |
| 5. The worst case of <code>ArrayQueue.enqueue</code> . ____ | 11. The worst case of <code>ArrayStack.push</code> . ____ |
| 6. The best case of <code>AVL.find</code> . ____ | 12. The worst case of <code>ArrayQueue.serve</code> . ____ |

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

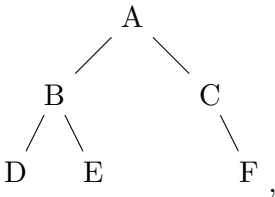
- | | | | |
|--------------------------------|-------------------------------|------------------------------------|------------------------------|
| A. <code>LinkedList</code> . | B. <code>ArrayList</code> . | C. <code>DoubleLinkedList</code> . | D. <code>LinkedList</code> . |
| E. <code>LinkedPQueue</code> . | F. <code>LinkedStack</code> . | G. <code>BT</code> . | H. <code>BST</code> . |
| I. <code>AVL</code> . | J. <code>BPlusTree</code> . | K. <code>HeapPQueue</code> . | L. <code>Graph</code> . |

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. ____
2. An algorithm takes a list of n points $(x_i, y_i), i = 1 \dots n$ and returns the k nearest points to a given point (x_0, y_0) . ____
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. ____
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). ____

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 }
```

- Line 2:
 - (A) `if (!l.first()){`
 - (B) `if (l == null){`
 - (C) `if (!l.full()){`
 - (D) `if (!l.empty()){`
 - (E) None
 - Line 3:
 - (A) `current = head;`
 - (B) `l.current = head;`
 - (C) `l.findNext();`
 - (D) `l.findLast();`
- (E) None
 - Line 4:
 - (A) `while (!l.first()){`
 - (B) `while (l.next != null){`
 - (C) `while (l.last()){`
 - (D) `while (!l.last()){`
 - (E) None
 - 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

• Line 6:

(A) `l.findNext();`

(B) `findNext();`

(C) `current = current.next;`

(D) `l.findFirst();`

(E) None

• 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 }
```

• 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

• 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

• 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

• Line 5:

(A) `p = p.left;`

(B) `return p.data;`

(C) `p = root;`

(D) `p = p.parent;`

(E) None

• Line 7:

(A) `p = p.parent;`

(B) `p = p.right;`

(C) `return p.data;`

(D) `p = root;`

(E) None

• 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 }

```

• Line 2:

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

• Line 6:

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

• Line 7:

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

• 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

• Line 9:

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

• Line 10:

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

• 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

• Line 13:

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

- 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
- 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 }
```

- Line 2:
 - (A) if (t.data == e)
 - (B) if (t != null)
 - (C) if (t == null)
 - (D) if (e.equals(t.data))
 - (E) None
 - 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
 - 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
 - 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
 - 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.

```
1 public static <T> int f(BT<T> b, T e, int k) {
2     if (...)
3         ...;
4     ...;
5     ...;
6 }
7 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 }

```

- Line 2:

☐ (A) if (b.full())
☐ (B) if (b.empty())
☐ (C) if (!b.empty())
☐ (D) if (e.equals(b.retrieve()))
☐ (E) None

- Line 3: ☐ (A) return 1;

☐ (B) return rf(b,e,k);
☐ (C) return 0;
☐ (D) return k;
☐ (E) None

- 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

- 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

- 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

- 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

- 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

- 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

- Line 13:

☐ (A) b.find(Relative.RightChild);
☐ (B) b.find(Relative.Parent);
☐ (C) b.find(Relative.Root);
☐ (D) return n+1;
☐ (E) None

- Line 15:

☐ (A) if (b.right != null){
☐ (B) if (b.find(Relative.RightChild)){

- | | |
|---|---|
| <p>Ⓒ if (b.find(Relative.Parent)){</p> <p>Ⓓ if (b.find(Relative.Root)){</p> <p>Ⓔ None</p> <p>• Line 16:</p> <p>Ⓐ n+=rf(b,e,k+1);</p> <p>Ⓑ n-=rf(b,e,k);</p> <p>Ⓒ n-=rf(b.right,e,k);</p> <p>Ⓓ n+=rf(b,e,k-1);</p> | <p>Ⓔ None</p> <p>• Line 17:</p> <p>Ⓐ b.find(Relative.Parent);</p> <p>Ⓑ b.find(Relative.LeftChild);</p> <p>Ⓒ return n+1;</p> <p>Ⓓ b.find(Relative.Root);</p> <p>Ⓔ None</p> |
|---|---|

Question 4.....12 points

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

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.
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) $O(n^2)$.
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) $O(n^2)$.
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).

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
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
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
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:

1. The maximum height of an AVL tree with 4 nodes is:
(A) 1. (B) 2. (C) 3. (D) 4. (E) 5.
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) $O(n^2)$.
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 .
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):

1. After inserting the key 17 in the AVL

12
10 14
13 15

14
12 15
10 13

12
10 15
13 17

10
12 15
13 14

10
12 15
13 17

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

2. After inserting the key 20 in the AVL

10
3 25
15 35

15
3 25
10 20 35

10
3 15 25 35

20
10 25
3 35

15
10 25
3 35

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

3. After deleting the key 11 from the AVL

8
5 15
3 7 10

8
5 10
3 7 15

7
5 10
3 8 15

7
3 10
5 8 15

7
5 10
3 8 15

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

5
3 8
4 7 30
6

4. After deleting the key 30 from the AVL , the tree becomes:

(A) 6
4 8
3 5 7

(B) 2
4 7
3 5 8

(C) 5
3 7
4 6 8

(D) 7
5 8
3 6
4

(E) None

Question 6 12 points

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

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

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

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

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					
Number of probes					

2. External chaining. Fill in the following table:

Key	22	15	12	27	18
Index of the list					

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					
Index of next element					

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:

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
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
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
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):

1. After inserting the key 12 to the B+ tree
becomes:

5

9

1

2

5

6

7

9

10

11

, the **root** of the tree

(A)

5

9

 (B)

5

9

11

 (C)

9

 (D)

6

10

 (E) None

2. After inserting the key 4 to the B+ tree
becomes:

5

9

1

2

3

5

6

7

9

10

, the **root** of the tree

(A)

4

 (B)

5

9

 (C)

4

7

 (D)

7

9

 (E) None

3. After deleting the key 1 from the B+ tree
becomes:

5

9

1

2

5

6

7

9

10

, the **root** of the tree

(A)

6

9

 (B)

7

 (C)

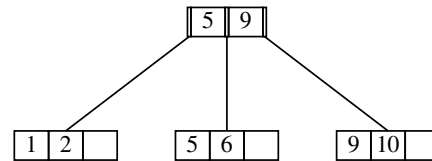
6

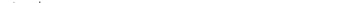
 (D)

5

7

 (E) None



4. After deleting the key 1 from the B+ tree , the **root** of the tree becomes:

- Ⓐ

2	9
---	---

 Ⓑ

5	9
---	---

 Ⓒ

6	
---	--

 Ⓓ

9	
---	--

 Ⓔ None

Question 8 6 points

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

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.

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

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

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

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

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

- (a) (a, c, e, f) _____

- (b) (a, c, b, d, f, e) _____

- (c) (a, b, e, c) _____

- (d) (f, e, b, d) _____

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

- (a) The graph is connected. _____

- (b) The number of edges in the graph is 8.

- (c) (b, e, f, b) is a cycle. _____

- (d) The number of 1s in the adjacency matrix of this graph is 18. ____

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) $a, b, f, c, d, e.$

- Ⓔ $a, b, c, f, e, d.$

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 .

- Ⓔ a, b, f, c, d, e .