	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`. ___
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 `ArrayQuue`. ___

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

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

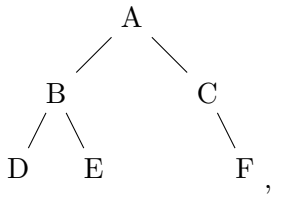
A. LinkedList.	**B**. ArrayList.	**C**. DoubleLinkedList.	**D**. LinkedQueue.
E. LinkedPQueue.	**F**. LinkedStack.	**G**. BT.	**H**. BST.
I. AVL.	**J**. BPlusTree.	**K**. HeapPQueue.	**L**. Graph.

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. ___
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)){`

Ⓒ `if (b.find(Relative.Parent)){`

Ⓓ `if (b.find(Relative.Root)){`

Ⓔ None

- Line 16:

Ⓐ `n+=rf(b,e,k+1);`

Ⓑ `n-=rf(b,e,k);`

Ⓒ `n-=rf(b.right,e,k);`

Ⓓ `n+=rf(b,e,k-1);`

Ⓔ None

- Line 17:

Ⓐ `b.find(Relative.Parent);`

Ⓑ `b.find(Relative.LeftChild);`

Ⓒ `return n+1;`

Ⓓ `b.find(Relative.Root);`

Ⓔ None

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?
   Ⓐ 1. Ⓑ 2. Ⓒ 3. Ⓓ 4. Ⓔ 5.
2. The best case run time for an insert in a heap represented as array is:
   Ⓐ $O(1)$. Ⓑ $O(\log n)$. Ⓒ $O(n)$. Ⓓ $O(n \log n)$ Ⓔ $O(n^2)$.
3. The worst case run time for a remove in a heap represented as array is:
   Ⓐ $O(1)$. Ⓑ $O(\log n)$. Ⓒ $O(n)$. Ⓓ $O(n \log n)$ Ⓔ $O(n^2)$.
4. Suppose all the keys in a heap have the same value. Then:
   Ⓐ The keys will be served in order of arrival (FIFO). Ⓑ The keys will be served in reverse order of arrival (LIFO). Ⓒ The keys will be served in pre-order. Ⓓ The keys will be served in in-order. Ⓔ 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:
   Ⓐ 3, 5, 6, 9, 7, 8, 10, 15, 11, 12, 4 Ⓑ 3, 4, 6, 9, 5, 8, 10, 15, 11, 12, 7 Ⓒ 3, 6, 4, 9, 5, 8, 10, 15, 11, 12, 7 Ⓓ 3, 4, 5, 6, 9 7, 8, 10, 15, 11, 12 Ⓔ None
2. Heap after inserting 2 then 13: Ⓐ 2, 3, 6, 9, 5, 8, 10, 11, 12, 7, 13, 15 Ⓑ 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15 Ⓒ 2, 3, 6, 9, 5, 8, 10, 11, 12, 7, 15, 13 Ⓓ 2, 3, 6, 9, 5, 8, 10, 15, 11, 12, 7, 13 Ⓔ None
3. Heap after deleting one key: Ⓐ 5, 6, 9, 7, 8, 10, 15, 11, 12 Ⓑ 5, 7, 6, 9, 8, 10, 15, 11, 12 Ⓒ 5, 7, 6, 9, 12, 8, 10, 15, 11 Ⓓ 5, 6, 7, 8, 9, 10, 11, 12, 15 Ⓔ None
4. Heap after deleting two keys: Ⓐ 6, 7, 8, 9, 12, 11, 10, 15 Ⓑ 6, 7, 8, 9, 10, 11, 12, 15 Ⓒ 6, 7, 9, 12, 8, 10, 15, 11 Ⓓ 6, 7, 9, 12, 8, 10, 11, 15 Ⓔ None

Question 5 ..... 12 points

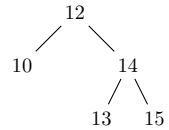
(a) (4 points)

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.*

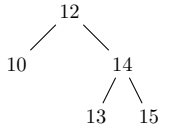
Choose the most appropriate answer:

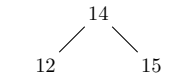
1. The maximum height of an AVL tree with 4 nodes is:
   Ⓐ 1. Ⓑ 2. Ⓒ 3. Ⓓ 4. Ⓔ 5.
2. The cost of one rotation in an AVL tree is:
   Ⓐ $O(1)$. Ⓑ $O(\log n)$. Ⓒ $O(n)$. Ⓓ $O(n \log n)$ Ⓔ $O(n^2)$.
3. The maximum number of rotations caused by an insert in an AVL tree with n nodes and height h is (a single rotation is counted 1; a double rotation is counted 2):
   Ⓐ 1. Ⓑ 2. Ⓒ h. Ⓓ $2h$ Ⓔ n.
4. The worst case height of an AVL tree with n nodes is:
   Ⓐ $O(1)$. Ⓑ $O(\log n)$. Ⓒ $O(n)$. Ⓓ $O(n \log n)$ Ⓔ $O(n^2)$.

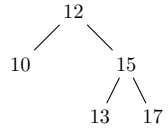
(b) (8 points) Choose the correct result in each of the following cases (follow the the convention of replacing with the smallest key in the right sub-tree when necessary):

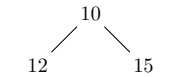
1. After inserting the key 17 in the AVL 

, the tree becomes:

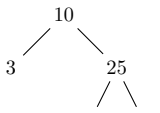
Ⓐ 

Ⓑ 

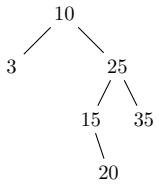
Ⓒ 

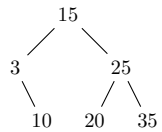
Ⓓ 

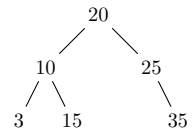
Ⓔ None

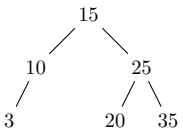
2. After inserting the key 20 in the AVL 

, the tree becomes:

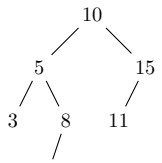
Ⓐ 

Ⓑ 

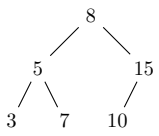
Ⓒ 

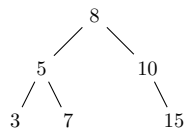
Ⓓ 

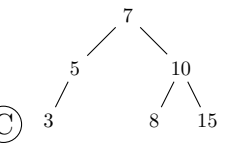
Ⓔ None

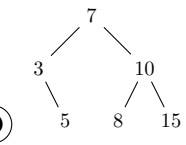
3. After deleting the key 11 from the AVL 

, the tree becomes:

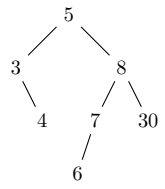
Ⓐ 

Ⓑ 

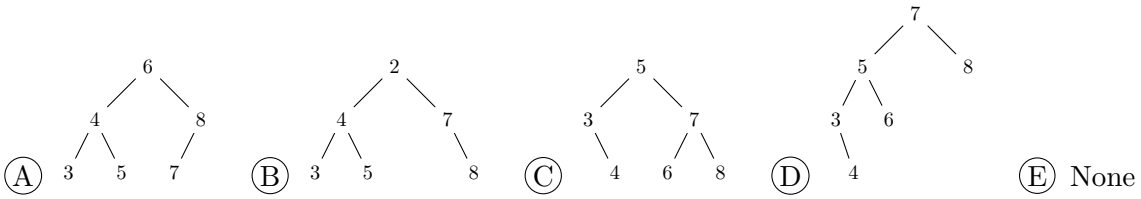
Ⓒ 

Ⓓ 

Ⓔ None

4. After deleting the key 30 from the AVL 

, the tree becomes:



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:
   Ⓐ $O(1)$ Ⓑ $O(\log n)$ Ⓒ $O(n)$ Ⓓ $O(n \log n)$ Ⓔ None
2. You want to store at most 16 keys in hash table using the % hash function. Choose the most appropriate table size:
   Ⓐ 14 Ⓑ 15 Ⓒ 16 Ⓓ 17 Ⓔ 18
3. How many keys can a hash table that uses folding on a single digit store if the key contains 4 digits?
   Ⓐ 4 Ⓑ 36 Ⓒ 37 Ⓓ 39 Ⓔ 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?
   Ⓐ 3848 and 4756 Ⓑ 3973 and 1258 Ⓒ 162 and 35476 Ⓓ All of the above. Ⓔ 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

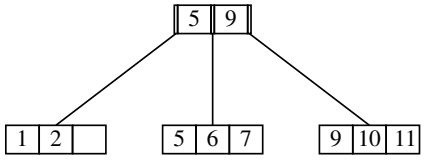
(a) (4 points)

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:
   Ⓐ 3 to 7 elements Ⓑ 4 to 7 elements Ⓒ 5 to 7 elements Ⓓ 0 to 7 elements Ⓔ 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:
   Ⓐ h Ⓑ $4h$ Ⓒ $\log h$ Ⓓ $3h$ Ⓔ None
3. The maximum number of data elements in a B+ tree with $m = l = 3$ and height 2 is:
   Ⓐ 3 Ⓑ 6 Ⓒ 9 Ⓓ 12 Ⓔ None
4. The height of a B+ tree containing n different keys is:
   Ⓐ $O(\log n)$ Ⓑ $O(n)$ Ⓒ $O(n \log n)$ Ⓓ $O(n^2)$ Ⓔ 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 

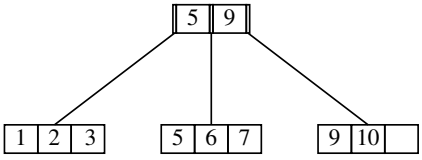
, the **root** of the tree becomes:
   Ⓐ 

Ⓑ 

Ⓒ 

Ⓓ 

Ⓔ None

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

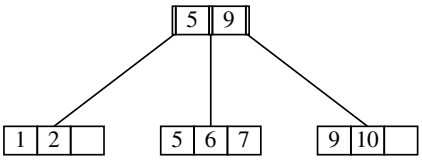
, the **root** of the tree becomes:
   Ⓐ 

Ⓑ 

Ⓒ 

Ⓓ 

Ⓔ None

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

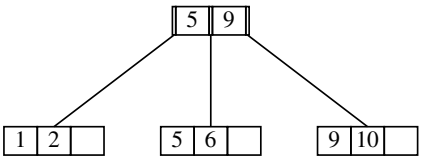
, the **root** of the tree becomes:
   Ⓐ 

Ⓑ 

Ⓒ 

Ⓓ 

Ⓔ None

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

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

Ⓐ 

Ⓑ 

Ⓒ 

Ⓓ 

Ⓔ 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:
   Ⓐ Preorder. Ⓑ Inorder. Ⓒ Postorder. Ⓓ BFS. Ⓔ 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	$\to b \to c$
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) ___
   (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) i 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, b, d, c, e, f. Ⓑ a, b, c, d, e, f.
   Ⓒ a, c, d, f, e, b. Ⓓ 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, b, d, c, f, e. Ⓑ a, b, d, c, e, f.
   Ⓒ a, b, c, d, e, f. Ⓓ a, c, d, f, e, b.
   Ⓔ a, b, f, c, d, e.