Name/Surname: Id:

CENG 112 – DATA STRUCTURES

SPRING 2015-2016 / Final Exam

08.06.2016

- Exam duration is 120 minutes
- No written notes
- No electronic devices
- ...Good Luck...

	Q1	Q2	Q3	Q4	Total
Points	30	20	25	25	**100**
Grade					

}

Q1. (30 Points) Linked Lists and Recursion

Given the node definition below

```
public class Node {
      int data;
      Node next;
}
```

a) Fill in the method below so that it computes the maximum element of the given list in an **iterative** way:

Answer:

```
public int maxIterative(Node list) {









}
```

b) Fill in the method below so that it computes the maximum element of the given list in a **recursive** way:

Answer:

```
public int maxRecursive(Node list, int currentMax) {









}
```

c) Fill in the method below so that it creates a new list that contains as many elements as the longer one of list1 and list2, and each element is the sum of the corresponding elements of list1 and list2. If one list is shorter, add 0 for the missing elements. For example pairwiseSumIterative of {1, 2, 3} and {4, 5, 6} is a new list containing { 5, 7, 9 } and pairwiseSumIterative of { 1, 2, 3 } and { 10 } is a new list containing { 11, 2, 3 }.

Answer:

```
public Node pairwiseSumIterative(Node list1, Node list2) {









}
```

d) Implement a recursive version of the method above. You are allowed to change the parameter list, returns type and define your own data types to use in the implementation.

Answer:

Q2. (20 Points) Stacks and Tree Traversals

Given the node definition

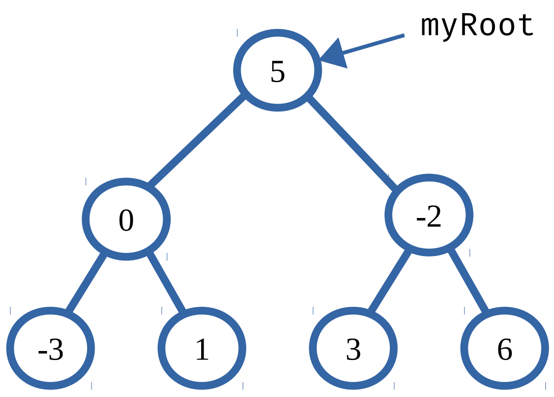
```
public class Node {
      int data;
      Node left;   // Left child
      Node right;  // Right child
}
```

and the method definition

```
public void printTree(Node root) {
      if (root == null || root.data < 0) {
            Stdout.println("deadend!");
            return;
      }

      StdOut.println("I'm on street " + root.data);
      printTree(root.left);
      printTree(root.right);
}
```

a) Write the output of the class printTree(myRoot) if myRoot refers to the root of the tree illustrated below:



<u>Answer:</u>

Name/Surname: Id:

b) Given the stack API given below, write an iterative version of the printTree() method by using an explicit stack to remove recursion:

```
public class Stack:
      public Stack();
      public void push(Node n);
      public Node pop();
      public int size();
      public boolean isEmpty();
```

Answer:

Q3. (25 Points) Searching with Trees

a) Draw the final Binary Search Tree (BST) after inserting the following letters in the given order: D, A, T, S, R, U, C

Answer:

b) What should be the order of insertion for the letters D, A, T, S, R, U, and C so that the resulting BST is perfectly balanced (s.t. all paths from root to leaf nodes are of the same length)? Explain if this order is unique or there are multiple possible insertion orders that will create a perfectly balanced BST?

Answer:

c) Draw the final 2-3 tree after inserting the following letters in the given order: D, A, T, S, R, U, C

Answer:

d) Draw the Red-Black tree **after** inserting **each one of** the following letters in the given order: D, A, T, S, R, U, C (You should draw **seven** trees for this part and you should note how you drew red nodes/links (double circles/bold lines).

Answer:

Your representation of Red Nodes/Links:

Your representation of **Black** Nodes/Links:

Q4. (25 Points) Searching with Hash Tables

Given the following has function H(x):

x	A	B	C	D	E	F	G
H(x)	3	5	2	3	2	4	7

Assuming that the items are inserted to the hash table in the alphabetic order:
A, B, C, D, E, F, G

a) Draw a hash table of size 8 with **<u>separate chaining</u>** after the insertion of all items above. *Note:* Array indices should start at zero.

<u>Answer:</u>

b) Draw a hash table of size 8 with **<u>linear probing</u>** after the insertion of all items above. *Note:* Array indices should start at zero.

<u>Answer:</u>

Name/Surname: Id:

c) Given the following hash table contents stored using linear probing, draw the hash table after F is deleted from the table.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
A		E	D	F	C	B	G

Answer:

d) Given the following composite data structure,

```
public class Point {
      double x;
      double y;
      String name;
}
```

i) Comment on the suitability of the hash function G(P) for arbitrary collections of points.

```
G(P) = (int) (P.x*P.x + P.y*P.y);
```

ii) Give the definition of a better hash function G'(P). *Hint:* You can use the Java defined hash function method for Strings, hashCode().

Answer: