WCOM125/ COMP125 Fundamentals of Computer Science
Workshop - Linked Lists

Learning outcomes

By the end of this session, you will have learnt about linked lists.

Questions

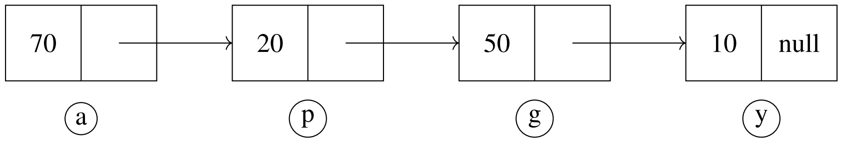
1 Node as the primitive for recursive data structure

All questions in this section use the following definition of `Node` class

```java
public class Node {
        private int data;
        private Node next;

        public int getData() {
                return data;
        }

        public Node getNext() {
                return next;
        }

        public void setData(int data) {
                this.data = data;
        }

        public void setNext(Node next) {
                this.next = next;
        }

        public Node(int data) {
                setData(data);
                setNext(null);
        }

        public Node(int data, Node node) {
                setData(data);
                setNext(node);
        }
}
```

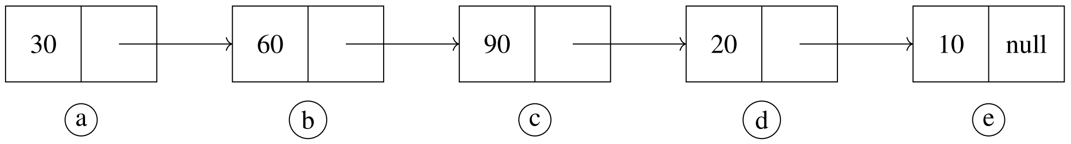
1. Draw a memory diagram representing objects in memory after the following code is executed.

```java
public class NodeStorage {
        public static void main(String[] args) {
                Node p = new Node(20, null);
                Node g = new Node(50, null);
                Node a = new Node(70, p);
                Node y = new Node(30, null);
                g.setNext(y);
                p.setNext(g);
                y.setData(10);
                p.getNext().getNext().setData(90);
        }
}
```



2. Consider the following code:

```java
Node e = new Node(10, null);
Node d = new Node(20, e);
Node c = new Node(90, d);
Node b = new Node(60, c);
Node a = new Node(30, b);
```



Write a piece of code that displays the data value of each node, starting at `Node a`. You must use a loop to do this.

3. Using the same definition for class `Node` as the previous question, what is the output produced by the following piece of code?

```java
public class Client {
        public static void main(String[] args) {
                Node n = new Node(1, null);
                for(int i=1; i < 4; i++) {
                        Node temp = new Node(2*i+1, n);
                        n = temp;
                }

                Node current = n;
                while(current != null) {
                        System.out.println(current.getData());
                        current = current.getNext();
                }
        }
}
```

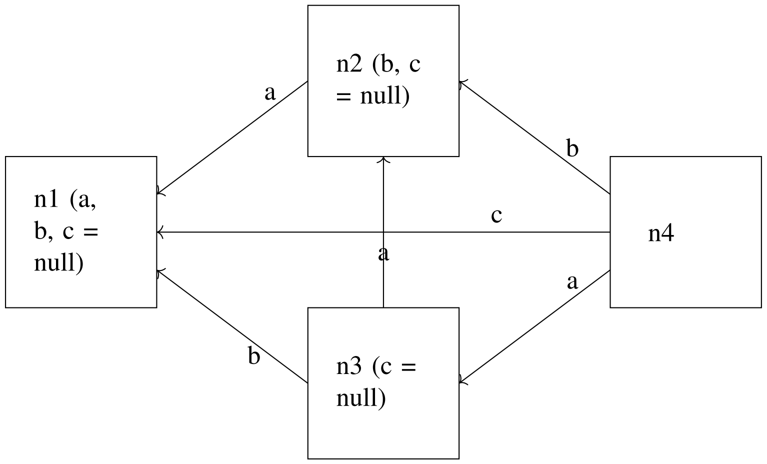
SOLUTION: Output is -

```
7
5
3
1
```

4. Consider the following class:

```java
class Node2 {
        private int data;
        private Node2 a, b, c;
        public Node2(int d, Node _a, Node _b, Node _c) {
                data = d;
                a = _a;
                b = _b;
                c = _c;
        }
}

public class Client {
        public static void main(String[] args) {
                Node2 n1 = new Node(20, null, null, null);
                Node2 n2 = new Node(50, n1, null, null);
                Node2 n3 = new Node(10, n2, n1, null);
                Node2 n4 = new Node(70, n3, n2, n1);
        }
}
```

Draw a graph illustrating the nodes and the links between them. Provide a direction and label for each link.



2 Java's built-in LinkedList class

Java has a built-in implementation of linked lists in class `LinkedList`. It behaves ***almost*** identically to `ArrayList` class. Thus, for the user, there is hardly any difference.

5. Write a method `countPositives` that when passed an `LinkedList` of `Double` objects, returns the number of positive items in the `LinkedList`. The method should return 0 if the list is `null` or empty.

```java
        public static int countPositives(LinkedList <Double> list)
```

Solution:

```
public static int countPositives(LinkedList <Double> list) {
                if(list == null || list.size() == 0)
                        return 0;
                int result = 0;
                for(Double item: list)
                        if(item > 0)
                                result++;
                return result;
        }
```

6. Write a method `countMatches` that when passed an `LinkedList` of `String` objects and a String `target`, returns the number of items in the list that contains `target`. The method should return 0 if the list is `null` or empty. For example, if `list = ["thereby", "they", "proved", "the", "other", "guy", "was", "the", "father"]` and `target = "the"`, the method should return 6, as there are six Strings containing `"the"`.

```
        public static int countMatches(LinkedList<String> list, String
            target)
```

Solution:

```
        public static int countMatches(LinkedList<String> list,
            String target) {
                if(list == null || list.size() == 0)
                        return 0;
                int result = 0;
                for(String item: list)
                        if(item.contains(target))
                                result++;
                return result;
        }
```

7. Add a method `count` that when passed an `LinkedList<Integer> list`, returns the number of prime numbers in `list`.

```
        public static int countPrimes(LinkedList<Integer> list)
```

Solution:

```
public static int countPrimes(LinkedList<Integer> list) {
        if(list == null)
                return 0;
        int result = 0;
        for(Integer item: list)
                if(isPrime(item))
                        result++;
        return result;
}

public static boolean isPrime(int n) {
```

```java
        if(n < 2)
                return false;
        for(int i=2; i*i <= n; i++)
                if(n%i == 0)
                        return false;
        return true;
}
```

8. **(D-level)** Write a method that when passed a `LinkedList` of integers, returns a number constructed with the first digit of each item of the list. The method should return 0 if the list is `null` or empty. For example, if the list is [15, 673, 8914], the method returns the number 168.

Solution:

```java
public static int getFirstDigitNumber(LinkedList <Character> list) {
        if(list == null)
                return null;
        int result = 0;
        for(Integer item: list)
                result = result*10 + firstDigit(item);
        return result;
}

public static int firstDigit(int n) {
        if(n == 0)
                return 0;

        if(n < 1)
                n*=-1;

        while(n > 10) {
                n/=10;
        }
        return n;
}
```

9. Write a method `getPerfectSquares` that when passed a `LinkedList<Integer> list`, returns a list containing perfect squares (squares of integers) in that list.

```java
public static LinkedList<Integer> getPerfectSquares(LinkedList<Integer>
    list)
```

Solution:

```java
public static LinkedList<Integer> getPerfectSquares(LinkedList<
    Integer> list) { if(list == null)
                return null;

        LinkedList<Integer> result = new LinkedList<Integer>();
        for(Integer item: list) {
                double root = Math.sqrt(item * 1.0);
                if((int)root * root == item)
                        result.add(item);
        }
        return result;
}
```