MACQUARIE University

Faculty of Science and Engineering

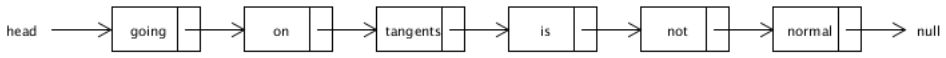
COMP125 Fundamentals of Computer Science
Workshop Week 11

Learning outcomes

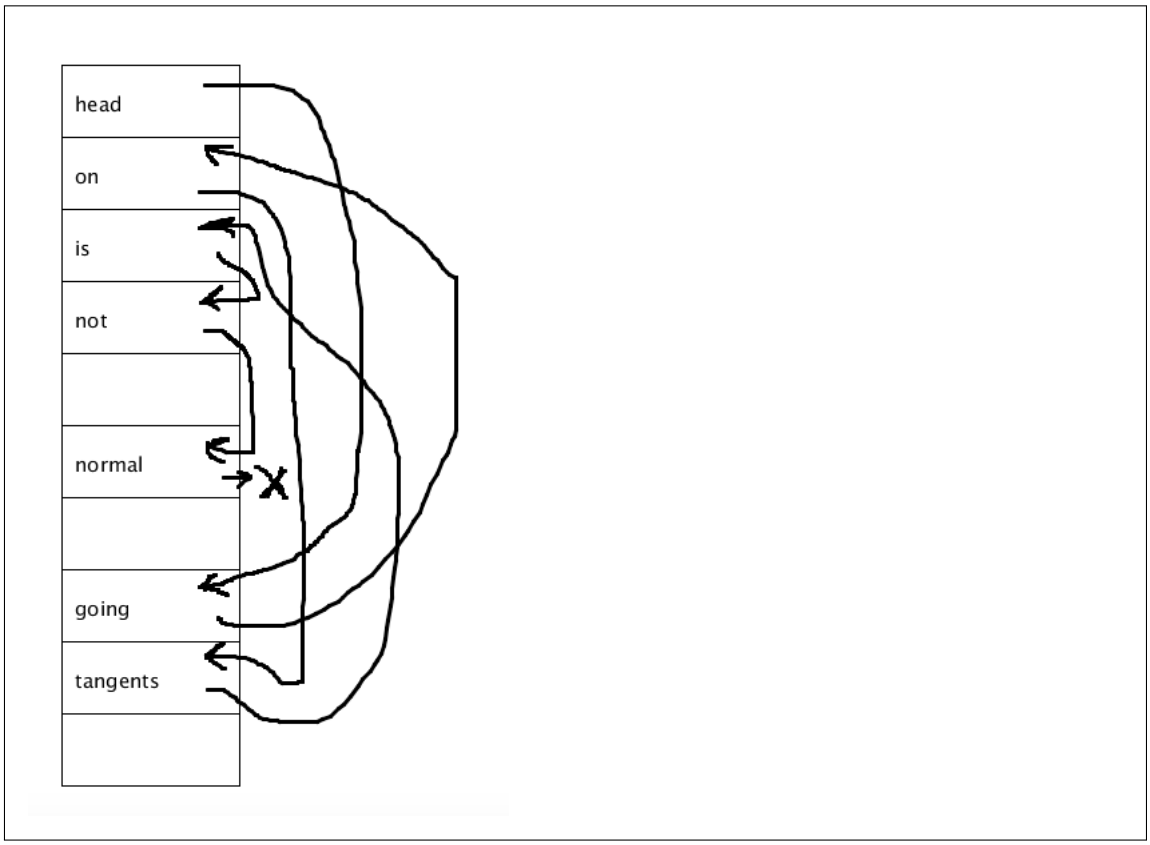
By the end of this session, you will have learnt about linked lists.

Questions

1. Represent, both as a logical diagram, and a memory diagram, the state of a linked list containing the following items (in the order of appearance in the list),
    a. "going"
    b. "on"
    c. "tangents"
    d. "is"
    e. "not"
    f. "normal"



Solution:



2. A `LinkedList` is a resizable set of objects such that each item links to the next item. If you don't parameterise an `LinkedList`, it can hold a variety of objects. That is, each item of the `LinkedList` can be of a different class.

   A parameter-less `LinkedList` is created as -

```
1    LinkedList list = new LinkedList();
```

   where `list` is the `LinkedList` object.

   You can parameterize an `LinkedList` so that it stores objects of a specific class. A parameterized `LinkedList` is created as -

```
1    LinkedList<ClassType> list = new LinkedList<ClassType>();
```

   where `list` is the `LinkedList` object.

For example,

```
    LinkedList<String> list = new LinkedList<String>();
```

can only hold `String` objects.

A subset of methods (the important ones) applicable to an `LinkedList` object is given below -

- `int size()`: returns the number of items in the list
- `Object get(int index)`: returns the `Object` at the specified index, if any; and `null` otherwise.
- `add(Object obj)`: adds the specified `Object` to the end of the list and returns `true`, if it can; and `false` otherwise.
- `add(int idx, Object obj)`: adds the specified `Object` at given index. Shifts all items at index `idx` onwards to the right.
- `contains(Object obj)`: returns `true` if the specified exists, and `false` otherwise.
- `indexOf(Object obj)`: returns the index of the specified `Object` if it exists, and -1 otherwise.
- `remove(Object obj)`: removes the specified `Object` to the list and returns `true`, if it can; and `false` otherwise.
- `set(int index, Object obj)`: updates the item at given index ot the object passed. Returns the item that the new object has replaced.

Write a piece of code that performs the following operations in the given order -

a. Create a linked list of integers
b. Add items (in that order) to the end of the list: 30, 70, 40, 80, 20, 100, 60
c. Update the list so that each item becomes twice of its current value
d. Calculate and store the highest value in the list in a variable `max`

Solution:

```
LinkedList<Integer> list = new LinkedList<Integer>();
list.add(30);
list.add(70);
list.add(40);
list.add(80);
list.add(20);
list.add(100);
list.add(60);
for(int i=0; i < list.size(); i++)
        list.set(i, list.get(i) * 2);
int max = list.get(0);
for(Integer item: list)
        if(item > list)
                max = item;
```

3. (Assessed task) 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.

```
        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;
}
```

4. (Assessed task) 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;
        }
```

5. (Assessed Task) 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) {
        if(n < 2)
                return false;
        for(int i=2; i*i < n; i++)
                if(n%i == 0)
                        return false;
        return true;
}
```

6. (Voluntary assessed task) 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:

```
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;
}
```

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

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

Solution:

```
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;
}
```