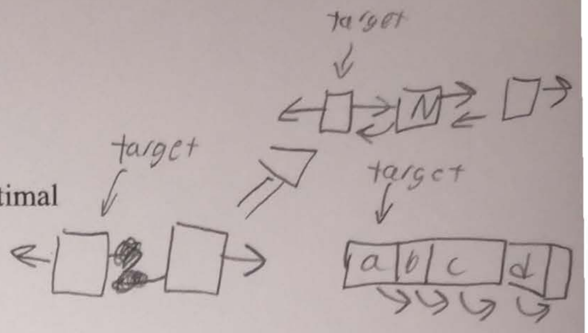
EECS 233 SI Session Three
Leader: Bertram Su
September 15, 2019



Objectives:

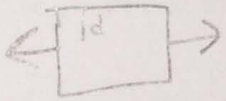
Upon completion of this SI session, participants will be able to:

1. Recognize situations where doubly linked lists would be optimal
2. Implement node classes to create a doubly linked list
3. Create methods for doubly linked lists

Foundation:

Operation	**Doubly Linked Lists**	**Arrays**
Access nth item	O(n)	O(1)
Search	O(n)	O(n)
Insertion (You have a field storing where to insert)	O(1)	O(n)
Deletion (You have a field storing where to delete)	O(1)	O(n)

Fill in the missing fields for a class Student so that it could act like a DLL node



```
public class Student {
      int id;
      String name;

      student next;
      student prev;
     //Assume getter, setter, and constructor methods are below
}
```

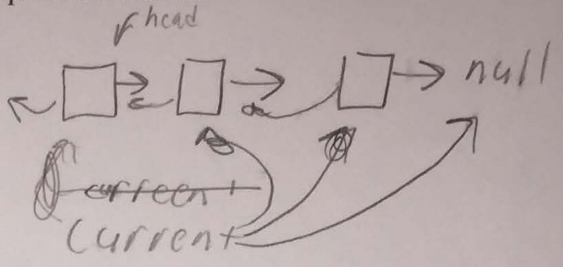
Exercises:

1. Create a class that uses Student objects called Recitation and acts like a doubly linked list class. What might a class that uses Student Nodes need?

```
public class Recitation{
     Student head;
     student tail;
     int size;



     //assume standard constructors and getters/setters are below
}
```

2. We want to be able to find a Student by id. Complete the method

```
public Student search(int id) {
```



```
}
```

3. Determine the simplest big O expression. Assume there are n items in the linked list. If he's trying to remove the node with the int id, will this work for a doubly linked list?

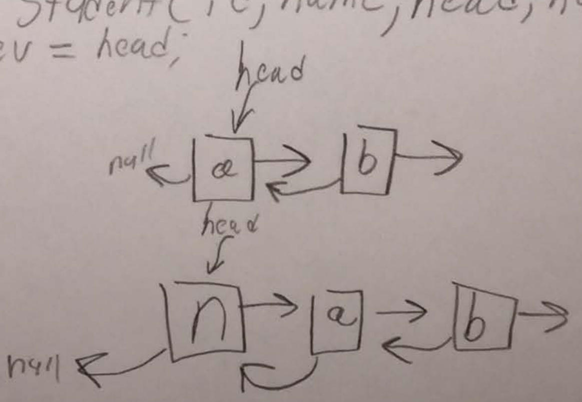
```
public boolean remove(int id) {
        Student ptr = head;
        Student targetNode = null;
        while (ptr != null) {
                if (ptr.getID() == id)
                        targetNode = ptr;
                ptr = ptr.getNext();
        }
        if (targetNode == null)
                return false;
        else {
                Student temp = targetNode;
                targetNode.setNext(null);
                temp.setNext(targetNode.getNext());
                return true;
        }
}
```

No, doesn't work

O(n) or O(1)

4. Using search(int id) to prevent id duplicates, create a method that will add an entry into Recitation.

```
public void add(String name, int id) { //Add it to the beginning and use the constructor
                                       // public Student(int id, String name, Student next, Student Prev)
    if(search(id) == null) {
        head = new Student(id, name, head, null);
        head.next.prev = head;
    }
}
```



5. Using search, create a method that will remove the student with an input ID for Recitation

```
public void remove(int id) {
    target = Search(id);
    if (target == head)
        head = head.next;
    else if (target == tail)
        tail = tail.prev;
    if (target.next != null)
        target.next.prev= target.prev
    if (target.prev != null)
        target.prev.next = target.next;
}   target.next = null;
    target.prev = null;
```

O(n)

Summary Checks:

6. Create a method that will print the name and id of all the Students in Recitation.

```
public void print() {
    Student ptr = head;
    while( ptr != nul)
        S.o.p (ptr.id + " _ " +ptr.name);
        ptr = ptr.next;
}
```

8. When would it be beneficial to use a doubly linked list?

inserting/deleting

7. Write what data structure you would use for the following situations

a) You constantly add and remove items from the beginning like a Pringles can

DLL

b) You need to access the items at an integer location given by your user.

array

EECS 233 SI Session Two
Leader: Bertram Su
September 12, 2019

Objectives:

Upon completion of this SI session, participants will be able to:

1. Use linked list nodes to store data
2. Implement interfaces and know their uses

Foundation:

1. How are classes and objects related?

Objects are instances of a class.

2. What are interfaces used for?

planned functionality

Exercises:

3. Create an interface for a vehicle

```
public interface vehicle{
    void drive();
    void turn();
    int speedometer();

}
```

4. We want to create list interface with generics but without collections

```
public interface simpleList <E> {

    E get();
    void set(E e);
    void add(E e);
    void remove(int i);
    void remove(E e);

}
```

5. Create a class for a Node that holds an integer.

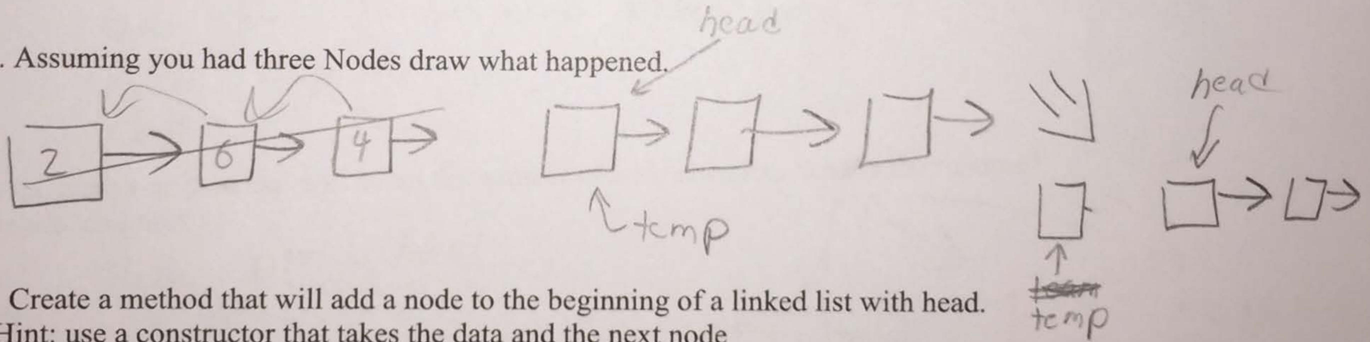
```
public class Node{
        int data;                       Node (int i, Node n){
        Node next;                          this.data = i;
                                            this.next=n;

                                        }

        //Assume getter, setter, and constructor methods are below
}
```

6. Write the code to remove the head of a linked list and determine the simplest big O expression. Assume head is an accessible node.

```
public void removeHead() {
    if (isempty()==false){
        Node temp = head;
        head = head.next;
        temp.next = null;
    }
}
```

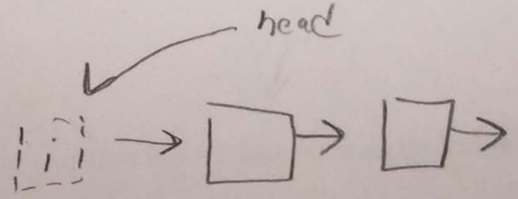
O(1)

7. Assuming you had three Nodes draw what happened.



8. Create a method that will add a node to the beginning of a linked list with head.
*Hint: use a constructor that takes the data and the next node

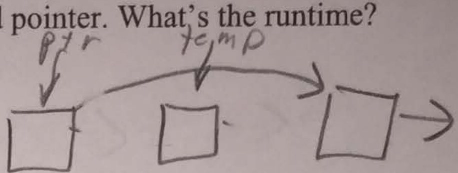
```
public boolean addBeginning(int i) {
    head = new Node (i,head);
```



```
}
```

9. Create a method that will remove the node after a Node called pointer. What's the runtime?

```
public void remove() {
    temp = pointer.next;
    pointer.next = temp.next;
    temp.next = null;
}
```



____

Summary Checks:

10. What are the two most important fields for a node to contain?

1. E data
2. Node next

11. Come up with a situation where we would want to use an interface.

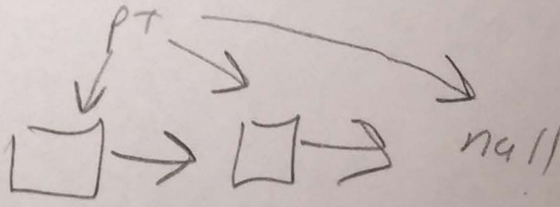
Specify functionality
thermostat

12. When would we use a linked list over an array to store data?

We don't know how much data
Add beginning
remove beg

13. Create a method that will count the number of nodes in a list. What's the runtime?

```
public int count() {
    Node ptr = head;
    int count = 0;
    while (ptr != null) {
        count++;
        ptr = ptr.next
    }
    return count
}
```



O(n)

14. Draw what occurred in question 9 if we had three nodes and pointer was the middle Node.