Problem 2

1. This code achieves 0(1) for time complexity because we are just removing from the first node which will always point to the first one that was entered

**public** **int** remove(){

**int** temp = first.data;

first = first.next;

size--;

**return** temp;

}

In all we are telling the address that the first one is in. that means we dont need to do any searching. The idea is similar for get vaule which if were given something greater than its index we know its a -1 value.

**if**(i>size-1){

**return** -1;

}

2. The code will receive 0(N) for its add function because

**if**(size == 0){

first = last;

}

**else**{

oldlast.next = last;

}

size++;

we are going through and making sure the the next value will always find its location therefore it will never go above N times.