

Introduction to Computer Science II

CS112-2008S-30

Midterm Review

David Galles

Department of Computer Science
University of San Francisco

30-0: Midterm 2

- Midterm 2 is in 1 week
 - Code tracing
 - Writing small methods / functions

30-1: Problems ...

```
public static void main(String arg[])
{
    Integer i1 = new Integer(1);
    Integer i2 = new Integer(2);
    Integer i3 = new Integer(1);
    Integer i4 = i3;

    System.out.println(i1 == i2);
    System.out.println(i1.equals(i2));
    System.out.println(i1 == i3);
    System.out.println(i1.equals(i3));
    System.out.println(i1 == i4);
    System.out.println(i1.equals(i4));
    System.out.println(i2 == i3);
    System.out.println(i2.equals(i3));
    System.out.println(i2 == i4);
    System.out.println(i2.equals(i4));
    System.out.println(i3 == i4);
    System.out.println(i3.equals(i4));
}
```

30-2: Problems ...

```
public static void main(String arg[])
{
    Integer i1 = new Integer(1);
    Integer i2 = new Integer(2);
    Integer i3 = new Integer(1);
    Integer i4 = i3;

    System.out.println(i1 == i2);           false
    System.out.println(i1.equals(i2));      false
    System.out.println(i1 == i3);          false
    System.out.println(i1.equals(i3));      true
    System.out.println(i1 == i4);           false
    System.out.println(i1.equals(i4));      true
    System.out.println(i2 == i3);           false
    System.out.println(i2.equals(i3));      false
    System.out.println(i2 == i4);           false
    System.out.println(i2.equals(i4));      false
    System.out.println(i3 == i4);           true
    System.out.println(i3.equals(i4));      true
}
```

30-3: Problems ...

```
public static void mangle(LList L)
{
    while (L.next != null)
    {
        L.next = L.next.next;
        L = L.next;
    }
}

public static void main(String arg[])
{
    LList L = new LList(1,null)
    L = new LList(2,L);
    L = new LList(3,L);
    L = new LList(4,L);
    L = new LList(5,L); mangle(L);
    for (LList tmp = L; tmp != null; tmp = tmp.next) {
        System.out.print(tmp.data);
    }
}

public class LList
{
    public LList next;
    public int data;

    public LList(int d,
                  LList n)
    {
        data = d;
        next = n;
    }
}
```

30-4: Problems ...

```
public static void mangle(LList L)
{
    while (L.next != null)
    {
        L.next = L.next.next;
        L = L.next;
    }
}
```

```
public static void main(String arg[])
{
    LList L = new LList(1,null)
    L = new LList(2,L);
    L = new LList(3,L);
    L = new LList(4,L);
    L = new LList(5,L);  mangle(L);
    for (LList tmp = L; tmp != null; tmp = tmp.next) {
        System.out.print(tmp.data);
    }
}  Output: 531
```

```
public class LList
{
    public LList next;
    public int data;

    public LList(int d,
                  LList n)
    {
        data = d;
        next = n;
    }
}
```

30-5: Problems ...

- Write a method that returns a new linked list, with each element duplicated
- `LList duplicate(LList L);`

Called with `[2, 4, 1]`, returns `[2, 2, 4, 4, 1, 1]`

```
public class LList {
    public int data;
    public LList next;
    public LList(int d, LList n)
    {
        data = d;
        next = n;
    }
}
```

30-6: Problems ...

```
public static LList duplicate(LList L)
{
    LList returnVal = null;
    if (L == null)
        return null;
    returnVal = new LList(L.data, null);
    returnVal = new LList(L.data, returnVal);
    LList tmp = returnVal.next;
    L = L.next;
    while (L != null)
    {
        tmp.next = new LList(L.data, null);
        tmp.next = new LList(L.data, tmp.next);
        tmp = tmp.next.next;
        L = L.next;
    }
    return returnVal;
}
```

30-7: Problems ...

```
public static LList duplicate(LList L)
{
    if (L == null)
        return null;
    LList returnVal = duplicate(L.next);
    returnVal = new LList(L.data, returnVal);
    returnVal = new LList(L.data, returnVal);
    return returnVal;
}
```

30-8: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the array n :

```
public static int countOccurrences(int data[], int elem, int n)
{
    int numOccurrences = 0;
    for (int i = 0; i < n; i++)
    {
        if (data[i] == elem)
            numOccurrences++;
    }
    return numOccurrences;
}
```

30-9: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the array n :

```
public static int countOccurrences(int data[], int elem, int n)
{
    int numOccurrences = 0;
    for (int i = 0; i < n; i++)
    {
        if (data[i] == elem)
            numOccurrences++;
    }
    return numOccurrences;
}
```

- $O(n)$

30-10: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the array n :

```
public static boolean isFirst(int data[], int elem, int n)
{
    return data[0] == elem;
}
```

30-11: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the array n :

```
public static boolean isFirst(int data[], int elem, int n)
{
    return data[0] == elem;
}
```

- $O(1)$

30-12: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the array n :

```
public static boolean isLast(int data[], int elem, int n)
{
    if (n > 0)
        return data[n-1] == elem;
    else
        return false;
}
```

30-13: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the array n :

```
public static boolean isLast(int data[], int elem, int n)
{
    if (n > 0)
        return data[n-1] == elem;
    else
        return false;
}
```

- $O(1)$

30-14: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the List n :

```
public static boolean isLast(LinkedList L, int elem)
{
    if (L == null)
        return false;
    while (L.next != null)
        L = L.next;
    return L.data == elem;
}
```

- Bonus Question: Does this function modify the list passed in? That is, after a call to `isLast`, is the value of the list passed in changed?

30-15: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the List n : $O(n)$

```
public static boolean isLast(LinkedList L, int elem)
{
    if (L == null)
        return false;
    while (L.next != null)
        L = L.next;
    return L.data == elem;
}
```

- Bonus Question: Does this function modify the list passed in? That is, after a call to `isLast`, is the value of the list passed in changed? No! Why not?

30-16: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the array n :

```
public static int countDuplicates(int data[], int elem, int n)
{
    int numDuplicates = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if (i != j && data[i] == data[j])
                numDuplicates++;
    return numDuplicates;
}
```

- Bonus question: What does this method return for $[1, 3, 1, 2, 2]$?

30-17: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the array n :
 $O(n^2)$

```
public static int countDuplicates(int data[], int elem, int n)
{
    int numDuplicates = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if (i != j && data[i] == data[j])
                numDuplicates++;
    return numDuplicates;
}
```

- Bonus question: What does this method return for [1, 3, 1, 2, 2]?
- 4(!) How could we modify it to return 2 for this

30-18: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the array n :

```
public static int countDuplicates(int data[], int elem, int n)
{
    int numDuplicates = 0;
    for (int i = 0; i < n; i++)
        for (int j = i+1; j < n; j++)
            if (data[i] == data[j])
                numDuplicates++;
    return numDuplicates;
}
```

30-19: Problems ...

- What is the $O()$ running time of the following function, in terms of the length of the array n :

```
public static int countDuplicates(int data[], int elem, int n)
{
    int numDuplicates = 0;
    for (int i = 0; i < n; i++)
        for (int j = i+1; j < n; j++)
            if (data[i] == data[j])
                numDuplicates++;
    return numDuplicates;
}
```

- $O(n^2)$

30-20: Problems ...

- Write a tail-recurse function that creates a reversed version of a linked list. The original linked list should not be affected

```
public class LList {
    public int data;
    public LList next;
    public LList(int d, LList n)
    {
        data = d;
        next = n;
    }
}
```

30-21: Problems ...

- Write a tail-recurse function that creates a reversed version of a linked list. The original linked list should not be affected

```
public static LList reverse(LList input, LList output)
{
    if (input == null)
        return output;
    return reverse(input.next, new LList(input.data, output));
}
```

```
public static LList reverse(LList input)
{
    return reverse(input, null);
}
```