CSE115 / CSE503
Introduction to Computer Science I

Dr. Carl Alphonce
343 Davis Hall
alphonce@buffalo.edu

Office hours:
Thursday 12:00 PM – 2:00 PM
Friday 8:30 AM – 10:30 AM
OR request appointment via e-mail
Turn off and put away electronics:

- cell phones
- pagers
- laptops
- tablets
- etc.
Today
linear search

Coming up
binary search
REVIEW

TWO’S COMPLEMENT
2’s complement is a fixed-width representation

To find representation of \(-x\) given the representation of \(x\):

1. find the one’s complement of \(x\)
   
   \textit{do this by flipping all the bits in the representation (1 becomes 0, 0 becomes 1)}

2. find the two’s complement of the result
   
   \textit{do this by adding one to the one’s complement, ignoring any overflow carry}
- 4-bit wide two's complement
- Non-negative
- Negative
- Arrow shows the effect of negation
All four signed integral types use the two’s complement representation scheme.
The width of the representations differ, and therefore the possible range of values:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>BITS</th>
<th>BYTES</th>
<th>MIN</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8</td>
<td>1</td>
<td>-128</td>
<td>+127</td>
</tr>
<tr>
<td>short</td>
<td>16</td>
<td>2</td>
<td>-32,768</td>
<td>+32,767</td>
</tr>
<tr>
<td>int</td>
<td>32</td>
<td>4</td>
<td>-2,147,483,648</td>
<td>+2,147,483,647</td>
</tr>
<tr>
<td>long</td>
<td>64</td>
<td>8</td>
<td>-9,223,372,036,854,775,808</td>
<td>+9,223,372,036,854,775,807</td>
</tr>
</tbody>
</table>

That’s 9 quintillion, 233 quadrillion, 372 trillion, 36 billion, 854 million, 775 thousand, 8 hundred and 8.
In general, for an n-bit wide representation, the range of possible values is

\[-2^{(n-1)} \rightarrow +2^{(n-1)} - 1\]
channel 1
Which bit pattern represents -3 in a four-bit wide two’s complement?

A. 0011
B. 1011
C. 1100
D. 1101
E. 1111
Which bit pattern represents -3 in a four-bit wide two’s complement?

A. 0011  
B. 1011  
C. 1100  
D. 1101  
E. 1111

Convince your neighbor your answer is correct!
Which bit pattern represents -3 in a four-bit wide two’s complement?

A. 0011
B. 1011
C. 1100
D. 1101
E. 1111

(D. 1101)
channel 1
What base 10 value does 01010 represent when interpreted as a 5-bit wide 2’s complement number?

A. 5
B. -5
C. 10
D. -10
E. 1010
F. -1010
What base 10 value does 01010 represent when interpreted as a 5-bit wide 2’s complement number?

A. 5
B. -5
C. 10
D. -10
E. 1010
F. -1010

Convince your neighbor your answer is correct!
What base 10 value does 01010 represent when interpreted as a 5-bit wide 2’s complement number?

A. 5
B. -5
C. 10
D. -10
E. 1010
F. -1010
EQUALITY TESTING
Assume $x$ and $y$ are reference variables. What is the difference between:

$x == y$

and

$x.equals(y)$
java.awt.Color objects

Constructor:

    public Color(int red, int green, int blue)

Two color objects are .equals if their red, green and blue components match.
channel 1
Given:
Color r = new Color(0,0,0);
Color s = new Color(1,1,1);
Color t = new Color(1,1,1);
Color u = s;

Which of (A), (B) and (C) is/are false?

A. s != t && t.equals(s)
B. s == u && u.equals(s)
C. r != s && r.equals(s)
D. all of (A), (B) and (C) are false
E. none of (A), (B) or (C) are false
Given:
Color r = new Color(0,0,0);
Color s = new Color(1,1,1);
Color t = new Color(1,1,1);
Color u = s;

Which of (A), (B) and (C) is/are false?

A. s != t && t.equals(s)
B. s == u && u.equals(s)
C. r != s && r.equals(s)
D. all of (A), (B) and (C) are false
E. none of (A), (B) or (C) are false
Given:
Color r = new Color(0,0,0);
Color s = new Color(1,1,1);
Color t = new Color(1,1,1);
Color u = s;

Which of (A), (B) and (C) is/are false?

A. s != t && t.equals(s)
B. s == u && u.equals(s)
C. r != s && r.equals(s) ✔️
D. all of (A), (B) and (C) are false
E. none of (A), (B) or (C) are false
Given:
Color r = new Color(0,0,0);
Color s = new Color(1,1,1);
Color t = new Color(1,1,1);
Color u = s;

Which of (A), (B) and (C) is/are false?

A. s != t && t.equals(s)
B. s == u && u.equals(s)
C. r != s && r.equals(s)
D. all of (A), (B) and (C) are false
E. none of (A), (B) or (C) are false
LINEAR SEARCH

Is it there?
Computers are good at storing large amounts of data

Finding a particular value in a large collection is a typical operation

LINEAR SEARCH
how to search for a value in an unordered collection
General problem: determine whether a value exists in a given collection

Assumption: to make things easier we assume that the collection contains Strings

Approach: define a method which accepts as arguments a Collection<String> and a value of type String, and returns a boolean indicating whether the value is in the collection
Step 1 – stub out the method

```java
public boolean isMemberOf(String s, Collection<String> c){
    return false;
}
```

* Yes, ‘contains’ is a method already defined for this purpose. We are building an implementation of isMemberOf to understand how a method like contains works.
Step 2 – set up loop

Can use any of while/for/for-each.

```java
public boolean isMemberOf(String s, Collection<String> c) {
    for (String x : c) {
    }
    return false;
}
```
steps in defining method

Step 3 – set up test in body of loop

```java
public boolean isMemberOf(String s, Collection<String> c) {
    for (String x : c) {
        if (s.equals(x)) {
            return true;
        }
    }
    return false;
}
```
Step 3 – set up test in body of loop

```java
public boolean isMemberOf(String s, Collection<String> c) {
    for (String x : c) {
        if (s.equals(x)) {
            return true;
        }
    }
    return false;
}
```
public boolean isMemberOf(String s, Collection<String> c) {
    for (String x : c) {
        if (s.equals(x)) {
            return true;
        }
    }
    return false;
}
Step 4a – but we need to be careful, as s could be null!

```java
public boolean isMemberOf(String s, Collection<String> c) {
    for (String x : c) {
        if (s == null) {
            if (s == x) {
                return true;
            }
        } else {
            if (s.equals(x)) {
                return true;
            }
        }
    }
    return false;
}
```
Step 4b – another way to write the method

```java
public boolean isMemberOf(String s, Collection<String> c){
    for (String x : c) {
        if (s==null && s == x) {
            return true;
        }
        if (s!=null && s.equals(x)) {
            return true;
        }
    }
    return false;
}
```
Do check whether \( s \) == null just once.
FINDING THE MAX
General problem: determine the maximum value in a collection, such as an ArrayList

To make things easier we assume two things:
- that the collection contains non-null Integers
- that the collection is not empty

Approach: define a method which accepts as argument an ArrayList<Integer> and returns the largest value is in the collection
Collections can hold only references to objects. Values of primitive types cannot be put into collections directly. Each primitive type has an associated wrapper class:

<table>
<thead>
<tr>
<th>primitive type</th>
<th>wrapper class</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>Byte</td>
</tr>
<tr>
<td>short</td>
<td>Short</td>
</tr>
<tr>
<td>int</td>
<td>Integer</td>
</tr>
<tr>
<td>long</td>
<td>Long</td>
</tr>
<tr>
<td>char</td>
<td>Character</td>
</tr>
<tr>
<td>float</td>
<td>Float</td>
</tr>
<tr>
<td>double</td>
<td>Double</td>
</tr>
<tr>
<td>boolean</td>
<td>Boolean</td>
</tr>
</tbody>
</table>
Each instance of a wrapper class holds a primitive value. Examples:

```java
int i = 3;
Integer wi = new Integer(i);  // create wrapper
int y = wi.intValue();  // get int value

double d = 3.5;
Double wd = new Double(d);  // create wrapper
double z = wd.doubleValue();  // get double value
```
ASIDE: autoboxing

The compiler will insert code to wrap(box)/unwrap(unbox) primitive values as needed

Examples:

```java
int i = 3;  // primitive
Integer wi = i;  // autoboxing (wrap int)
int y = wi;  // unwrap Integer

Integer z = y + wi;  // unwrap and wrap

Integer z = new Integer(y + wi.intValue());
```
Step 1 – stub out the method

```java
public Integer maximum(ArrayList<Integer> c){
    return 0;
}
```

**Exercise**

Write this method on your own.