

## CS310 hw2 Solution

Thanks to Prof. Betty O'Neil for parts of the solution.

1. Transfer qual solution to cs310/hw2

Test: cd to student dir, hw2 subdir

du should show (any nontrivial numbers at left):

```
11  ./src/cs310
12  ./src
10  ./classes/cs310
11  ./classes
```

where the src/cs310 has the .java files for the qual.

For hw2, enough to have src/cs310 there.

2.

a) 'a' -> 0, 'b' -> 1, ..., 'z' -> 25, and also 'A' -> 0,...'Z' -> 26  
in one map:

```
public static int mapOneLetter(char c)
{
    return Character.toLowerCase(c) - 'a';
}
```

b) "aa" -> 0, "ab" -> 1, ..., "az" -> 26, "ba" -> 26,

```
public static int mapTwoLetter(String s)
{
    return ((s.charAt(0) - 'a') * 26) + s.charAt(1) - 'a';
}
```

c) inverse of b

```
public static String inverseMapTwoLetter(int x)
{
    StringBuffer sb = new StringBuffer();
    sb.setCharAt(0, 'a' + x / 26);
    sb.setCharAt(1, 'a' + x % 26);
    return sb.toString();
}
```

### 3. Suggested solution:

```
/**
 * List-based implementation of the stack.
 * Author: Mark Allen Weiss
 */
public class ListStack implements Stack {
    public ListStack( ) {
        topOfStack = null;
    }
    public boolean isEmpty( ) {
        return topOfStack == null;
    }
    public void makeEmpty( ) {
        topOfStack = null;
    }
    public void push( Object x ) {
        topOfStack = new ListNode( x, topOfStack );
    }
    public void pop( ) {
        if( isEmpty( ) )
            throw new UnderflowException( "ListStack pop" );
        topOfStack = topOfStack.next;
    }
    public Object top( ) {
        if( isEmpty( ) )
            throw new UnderflowException( "ListStack top" );
        return topOfStack.element;
    }
    public Object topAndPop( ) {
        if( isEmpty( ) )
            throw new UnderflowException( "ListStack topAndPop" );

        Object topItem = topOfStack.element;
        topOfStack = topOfStack.next;
        return topItem;
    }
    private ListNode topOfStack;
}
```

4. Look up `String.hashCode()` in the JDK documentation and report on the hash function. From the formula, compute the hashCode of "", "A", "AB". Note that 'A' has int value  $0x41 = 65$ , its "ASCII code."

Similarly report on the `Integer.hashCode()`.

From the online docs, we have for `String.hashCode()`

Returns a hash code for this string.

The hash code for a String object is computed as

$$s[0]*31^{(n-1)} + s[1]*31^{(n-2)} + \dots + s[n-1]$$

using int arithmetic, where  $s[i]$  is the  $i$ th character of the string,  $n$  is the length of the string, and  $^$  indicates exponentiation. (The hash value of the empty string is zero.)

```
"".hashCode() = zero // by definition
```

```
"G".hashCode() = s[0]*31^(1-1) = s[0]*31^0 = s[0]*1 = 71
```

```
"GH".hashCode() = s[0]*31^(2-1) + s[1]*31^(2-2)
                  = 71*31^1 + 72
                  = 2201 + 72
                  = 2273
```

From the online docs, we have for `Integer.hashCode()`

a hash code value for this object, equal to the primitive int value represented by this Integer object.

5. Explain how the function of 1b. can be used in an app that counts the number of occurrences of each letter pair in a document. In particular, suppose the current letter pair is in variable "String curPair." How can we count it one call to the function plus one more (fast) operation? Alternatively, without the function from 1b, how would you do the same thing with a Map implemented by HashMap?

We could have setup an array "int count[26\*26]" which used the above function and the curPair as follows:

```
count[mapTwoLetters(curPair)]++;
```

If we didn't have the above function, and we used the HashMap, we'd do something like the following:

```
HashMap<String> counts = new HashMap<String>();
// iterate through key values: "aa", "ab", "ac", ... , "yz",
"zz"
// and put an 0 in each spot.
for (...)
    counts.put(key, 0);
...
...
// then we could just increment the count by saying
int count = counts.get(curPair);
counts.put(curPair, count+1);
```

6. Print a Collection in reverse order without using ListIterator. We can't iterate backwards, only forwards. However, we can easily dump any Collection to an array of Objects, and printing can be done via toString, an Object method, so that's the simplest way to go here.

```
public static <T> printReverse(Collection<T> c)
```



```

        if (!checkMatch(match, ch)) <--Bug in Weiss!!
            return false;
    }
    break;
default:
    System.out.println("Non-bracket: "+ ch + " in string");
    return false;
}
if (!pendingTokens.isEmpty())
    return false; // incomplete expr (missing test in Weiss)
return true; // passed tests
}

private boolean checkMatch(char openCh, char closeCh)
{
    if (openCh=='(' && closeCh != ')') ||
        openCh=='[' && closeCh != ']') ||
        openCh=='{' && closeCh != '}') {
        System.out.println("Found " + closeCh + " not matching "+
openCh);
        return false;
    } else return true;
}

```