CS 31
Discussion 2H

Some pointers

1. Comment codes!
2. Test cases:
    a. Valid
    b. Invalid

Pass by Value vs Pass by Reference

Call by value vs call by reference: syntax

```
void addOne(int num);
int main(){
   int num = 4;
   addOne(num);
   cout << num << endl;
}

void addOne(int num){
   num++;
}
```

```
void addOne(int& num);
int main(){
   int num = 4;
   addOne(num);
   cout << num << endl;
}

void addOne(int& num){
   num++;
}
```

Call by value: example

❖ Output?

```
#include <iostream>
using namespace std;

void duplicate (int a, int b, int c) {
    a *= 2;
    b *= 2;
    c *= 2;
}

int main() {
    int x = 1, y = 3, z = 7;
    duplicate(x, y, z);
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "z = " << z << endl;
}
```

```
x = 1
y = 3
z = 7
```

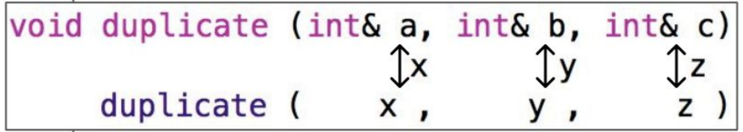
Call by reference: example

❖ Output?

```cpp
#include <iostream>
using namespace std;

void duplicate (int& a, int& b, int& c) {
    a *= 2;
    b *= 2;
    c *= 2;
}

int main() {
    int x = 1, y = 3, z = 7;
    duplicate(x, y, z);
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "z = " << z << endl;
}
```



```
void duplicate (int& a, int& b, int& c)
                     ↕x      ↕y      ↕z
    duplicate (     x ,     y ,     z )
```

```
x = 2
y = 6
z = 14
```

Time for practicing the Worksheet

1) What does the following code snippet output?

```
void mystery(int& a, int b) {
    int count = 0;
    while (count < 2) {
        a = a + b/2;
        b = a + 5;
        cout << "a: " << a << " b: " << b << endl;
        count++;
    }
}

int main() {
    int a = 5, b = 10;
    cout << "a: " << a << " b: " << b << endl;
    mystery(a, b);
    cout << "a: " << a << " b: " << b << endl;
}
```

1) What does the following code snippet output?

```
void mystery(int& a, int b) {
        int count = 0;
        while (count < 2) {
                a = a + b/2;
                b = a + 5;
                cout << "a: " << a << " b: " << b << endl;
                count++;
        }
}

int main() {
        int a = 5, b = 10;
        cout << "a: " << a << " b: " << b << endl;
        mystery(a, b);
        cout << "a: " << a << " b: " << b << endl;
}
```

```
a: 5 b: 10
a: 10 b: 15
a: 17 b: 22
a: 17 b: 10
```

2) What does the following code snippet output?

```
void mystery(char code) {
  switch(code) {
    case 'a':
    case 'b':
    case 'c':
      cout << "spooky";
     break;
   case 'd':
     cout << "feeling";
     break;
   case '1':
     cout << " ";
     break;
   case '2':
     cout << "?";
   default:
     cout << endl;
     break;
  }
}
```

```
int main() {
  string message = "d1a2c1d#";
  int i = 0;
  do {
    mystery(message[i]);
    i++;
  } while(i < message.length());
}
```

2) What does the following code snippet output?

```
void mystery(char code) {
  switch(code) {
    case 'a':
    case 'b':
    case 'c':
      cout << "spooky";
      break;
    case 'd':
      cout << "feeling";
      break;
    case '1':
      cout << " ";
      break;
    case '2':
      cout << "?";
    default:
      cout << endl;
      break;
  }
}
```

```
int main() {
  string message = "d1a2c1d#";
  int i = 0;
  do {
    mystery(message[i]);
    i++;
  } while(i < message.length());
}
```

```
feeling spooky?
spooky feeling
```

1) Create a function *changeString* that accepts two parameters:

- string1: a reference to a string value that does not contain spaces and
- string2: a string consisting of letters that will be used as delimiters.

Now, for every character in string2 that appears within string1, replace the letter within string1 with a space.

Note: You may assume that every letter within string2 will be unique.

For example:
changeString("HelatelmyIcookie", "l") -> "He ate my cookie"
changeString("ShouldeHIstartemylab?", "He") -> "Should I start mylab?"

```
void changeString(string& word, string delimiters) {
        for (int i = 0; i < word.length(); i++) {
                for (int j = 0; j < delimiters.length(); j++) {
                        if (word[i] == delimiters[j])
                                word[i] = ' ';
                }
        }
}
```

3) Write a function *findRun* that takes in a string of lowercase and uppercase alphabetical characters and returns the character with the longest "run." In other words, return the character that occurs the most times in succession. You may assume that the string is not empty. If two characters have equally long runs, return the first one.

For example:
findRun("abbccccdda") returns 'c'
findRun("aaaabcbbbcbcbcbcb") returns 'a'

```
char findRun(string s) {
  int maxRun = 0;
  char maxChar = ' ';
  int currRun = 0;
  char currChar = ' ';
 for (char c : s) {
   if (c == currChar)
     currRun++;
   else {
     currChar = c;
     currRun = 1;
   }

   if (currRun > maxRun) {
     maxRun = currRun;
     maxChar = currChar;
   }
 }
 return maxChar;
}
```

5) Write a function *findLastLength* that takes in a string that consists of uppercase alphabetical characters, lowercase alphabetical characters, and empty space ' ' characters. It returns the length of the last word, unless the last word does not exist, in which case it returns 0.

For example:

findLastLength("Misfits should have won against SKT") returns 3

findLastLength(" ") returns 0

```
int findLastLength(string s) {
  bool foundWord = false;
  int lastLen = 0;
  for (int x = s.length() - 1; x >= 0; x--) {
    bool onAlpha = isalpha(s[x]);

    if (foundWord && !onAlpha)
      break;
    else if (onAlpha) {
      foundWord = true;
      lastLen++;
    }
  }
  return lastLen;
}
```

Writing PseudoCode

http://web.cs.ucla.edu/classes/fall18/cs31/pseudocode.html

Good resources: http://netlab.cs.ucla.edu/~schoi/cs31/notes/cs31s11dis4sol.pdf

Thank You!