CS 241 – Week 1 Tutorial

C++ review

Spring 2017

1 Summary

1. Code style review
2. Standard Template Library (STL) review

2 C++ Review

You can use several different languages in CS241. For the most part we'll avoid talking about specific languages whenever possible in this course, but since most people elect to do the assignments in C++, here's a bit of C++ review.

2.1 Code Style

There are a number of flaws in the following code snippet. How can this piece of code be improved? Think about both stylistic improvements and performance improvements.

```
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include <string>
#include <algorithm>
using namespace std;

bool foo(vector<string> v) {
    if (v.size() > 16) {
            return true;
    } else {
            return false;
        }
}

int bar(map<string, map<vector<string>, int> >  m, string w) {
        return m[w].size();
}

bool baz(string fruit) {
```

```
        return fruit == "apple" || fruit == "pear" || fruit == "mango" ||
                fruit == "coconut" || fruit == "kiwi" || fruit == "pepper";
}


string temp;
bool qux(pair<vector<string>, int> p) {
        int count = 0;
        for (int i = 0; i < p.second; ++i) {
                if (p.first[i] == temp) count++;
        }
        if (count > p.second/2) {
                return true;
        } else {
                return false;
        }
}

int main() {
        vector<string> fruits;
    map<string, map<vector<string>, int> > fruitMap;
        while (true) {
                string fruit;
            cin >> fruit;

            fruitMap[fruit][fruits] = fruits.size();

                int mode;
                if (fruit == "apple") {
                    mode = 0;
                } else if (fruit == "banana") {
                    mode = 1;
                } else if (fruit == "tangelo") {
                    mode = 2;
                } else {
                    throw 143;
                }

                fruits.push_back(fruit);
        if (foo(fruits)) {
                    cout << "Many␣fruits" << endl;
                }

        int val = bar(fruitMap, fruit);

                bool flag1 = false, flag2 = false;
                if (val > 12345) {
                    flag1 = true;
                } else if (fruits.size() > 8 && fruits.size() < 12) {
                    flag1 = true;
                } else if (mode == 1) {
```

```
                        flag2 = true;
                }

                if (flag2 || flag1 && baz(fruit)) {
                        break;
        }
        }


    for (map<string, map<vector<string>, int> >::iterator it = fruitMap.begin()
          it != fruitMap.end(); ++it) {
        temp = (*it).first;
                cout << count_if((*it).second.begin(), (*it).second.end(), qux);
        }
}
```

2.2 Standard Template Library (STL)

The STL is a collection of generic containers and algorithms. Getting used to using these can make your code shorter, faster and easier to understand.

- containers: vector, list, map, set, and pair.
- algorithms: find, count, copy, for_each, transform, accumulate (and more!)

Try using the STL as much as possible to complete the following exercises:

1. Write a short C++ program which reads in a sequence of numbers separated by whitespace and prints the sequence twice: once forwards, then once backwards.
2. Modify your solution to the previous problem to read in a sequence of name and ID pairs, where the name and ID are separated by whitespace, and each pair is separated from the next pair by whitespace. Print 5 pairs per line, where each pair is formatted as `[name,ID]`. Avoid using global variables.
3. Write a short C++ program which reads from standard input line by line, and prints the number of times each character appears in that line. You may omit the count of newline character. For simplicity, print each character followed by its frequency on its own line. Make as much use of the STL as you can.