CS 241 – Week 1 Tutorial Solutions

Binary basics and C++ review

Winter 2017

1 Summary

1. C++ Review
2. Standard Template Library (STL) review

2 C++ Review

2.1 Code Style

Here are some problems about the code and possible ways we could improve it

1. Inconsistent Spacing. Indentation and use of spaces should be consistent throughout the program.
2. Repeated use of long type names. Mistake can be easily made when typing long, specific sequence of characters over and over again. Furthermore, these code are invulnerable when implementation changes are required. When there are complicated type, we can use typedef.
3. Redundant if statement. For example, in **foo**, the entire function could be replaced by simply returning the result of comparison.
4. Pass large data structures by value is inefficient, since the entire data structure are copied once the function is called. Pass by reference are often preferred.
5. Some code quick be efficiently solved and nicely presented by using the correct data structures. Make proper use of STL as much as possible. e.g. **baz** could be done by using **std::set**.
6. Magic numbers should be avoided entirely. All constant value should have a purpose, and a descriptive name.
7. Use **enum** constants to create a range of name constants. e.g. in **baz**, all string literals could be defined as enum.
8. Naming for functions and variables should be more descriptive.
9. Documentations are always the essential part of good style!

2.2 STL

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.

   Solution:

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

int main(){
  vector<int> seq;
  copy(istream_iterator<int>(cin), istream_iterator<int>(),
       back_inserter(seq));
  copy(seq.begin(), seq.end(), ostream_iterator<int>(cout, "\n"));
  copy(seq.rbegin(), seq.rend(), ostream_iterator<int>(cout, "\n"));
}
```

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.

   Solution:

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <iterator>
#include <string>
#include <numeric>
using namespace std;

struct Person {
  Person() : id(-1) {}

  string name;
  int id;
};

istream & operator>>(istream & in, Person & p) {
  return in >> p.name >> p.id;
}

ostream & operator<<(ostream & out, const Person & p) {
  return out << "[" << p.name << "," << p.id << "]";
}

struct Printer {
```

```
  ostream & out;
  int count;

  Printer(ostream & out) : out(out), count(0) {}

  void operator()(Person & p) {
    count = (count+1) % 5;
    out << p << (count == 0 ? "\n" : "␣");
  }
};

int main(){
  vector<Person> seq;
  copy(istream_iterator<Person>(cin), istream_iterator<Person>(),
    back_inserter(seq));
  for_each(seq.begin(), seq.end(), Printer(cout));
  cout << (seq.size() % 5 == 0 ? "" : "\n");
}
```

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.

   Solution:

```
#include <iostream>
#include <algorithm>
#include <map>
#include <string>

using namespace std;

typedef map<char, int> OccMap;

struct Fill{
  OccMap &occ;

  Fill(OccMap &occ) : occ(occ) {}

  void operator() (char c) {
    if (occ.count(c) == 0) occ[c] = 0;
    occ[c]++;
  }
};

struct Printer {
  ostream & out;

  Printer(ostream & out) : out(out) {}

  void operator()(pair<const char, int> &p) {
    out << p.first << "␣" << p.second << endl;
```

```
  }
};

int main(){
  string line;
  while (getline(cin, line)){
    OccMap occ;
    for_each(line.begin(), line.end(), Fill(occ));
    for_each(occ.begin(), occ.end(), Printer(cout));
  }
}
```