# COP4530 Recitation Fall 2012
# Week 2

* If you have the Monday recitation (Labor Day) and struggle with the concepts covered in this recitation handout, please attend office hours or schedule an appointment.

## Objective

1. g++ Optimization flags
2. Assignment 1 Discussion

## g++ Optimization flags

### A. Introduction

g++ offers various optimization options when compiling. It is important to note, optimization does not always mean faster. Optimizations can involve improvements in runtime, space (executable size), or memory usage.

### B. Flags

What does –O mean?

What does –O2 mean?

What does –O3 mean?

What does –Os mean?

What does –O0 mean?

### C. Benefits at a cost

“Without any optimization option, the compiler's goal is to reduce the cost of compilation and to make debugging produce the expected results. Statements are independent: if you stop the program with a breakpoint between statements, you can then assign a new value to any variable or change the program counter to any other statement in the function and get exactly the results you would expect from the source code.

Turning on optimization flags makes the compiler attempt to improve the performance and/or code size at the expense of compilation time and possibly the ability to debug the program."[1]

## Programming Assignment 1 Discussion

### A. Toy example of similar concepts

We will discuss some concepts in Assignment 1 through a toy example. Suppose we are tasked to develop a system to find the percent of cars at three dealerships with a particular color. The expected usage is:

```
> searchCars <color>
```

Furthermore, we have to output the name of the dealer and the query result to the screen:

```
> searchCars blue
Toyota: 14.2857%
Honda: 15.3846%
Ford: 15.3846%
```

### B. Interface

We are provided an interface for a class CarDealer we need to implement.

```
#############################################################
      class CarDealer
      {
          public:
              // Set inventory to contents of filename and set name
              CarDealer(string filename, string dealerName);
              int search(string color); // return # of color at dealer
              float percentMatching(int count); // return % at a dealer
              string dealerName(); // return name

          private:
              vector<string> cars; // color inventory
              string name; // dealer name
      };
#############################################################
```

[1] http://gcc.gnu.org/onlinedocs/gcc-4.3.2/gcc/Optimize-Options.html

## C. Implementation

Can you fill in the missing code (….)?

```
#############################################################
      // filename is a text file with a list of colors on a single line
      // separated by a single-space (Ex/ black blue orange ...)
      CarDealer::CarDealer(string filename, string dealerName)
      {
          name = dealerName;

          // set the ifstream pointer to a new ifstream object
          // using filename
          ifstream* infile = ……..

          string car;
          // read the colors from infile and add the read color
          // into the cars vector

          // deallocate memory
          delete infile;
      }

      int CarDealer::search(string color)
      {
          vector<string>::iterator itr;
          int count = 0;

          // use itr to loop through the vector cars (vector of colors)
          // and increment color each time a match is found
          for (……..;……..;……..) {
              if (……..)
                  ++count;
          }
          return count;
      }

      float CarDealer::percentMatching(int count)
      {
          // the percent matching is the value of count divided
          // by the total number of cars at a dealer times 100
          ……..
      }

      string CarDealer::dealerName()
      {
          return name;
      }
      ############################################################
```

## D. Driver program

Can you fill in the missing code (….)?

```
############################################################
      #include<iostream>
      #include<fstream>
      #include<string>
      #include<vector>
      using namespace std;
      #include “car_dealer.h”

      int main(int argc, char* argv[])
      {
          if (argc != 2) {
              cout << “Usage: search <color>” << endl;
              exit(1);
          }

          string color = argv[1],
                 input;
          vector<CarDealer> dealerships;
          vector<CarDealer>::iterator itr;

          // add three CarDealer objects to dealerships
          // you can make up the dealer name
          ……
          ……
          ……

          // search each dealership and display the result
          // use itr to loop through dealerships
          for (……;……;……) {
              cout << …… << “: “
                   << …… << “%” << endl;
          }

          return 0;
      }
############################################################
```

### D. Improvements

How can we improve on this code?

## References:


1. Vector: http://www.cplusplus.com/reference/stl/vector/
2. Dynamic memory: http://www.cplusplus.com/doc/tutorial/dynamic/