Problem Set 0
ISE 407 – Computational Methods in Optimization
Due: September 5, 2019
Dr. Ralphs

This assignment is to get you familiar with compilation and testing of programs written in C/C++. Completing this assignment will require some familiarization with the programming environments and tools we'll use during the rest of the semester. The objective of this assignment is simply to get familiar with the tools and take a first try at doing some computational experiments. The overall assignment is to try to replicate the experiment discussed in Example 3 of Lecture 2, as follows.

1. Write a small C++ code that takes two command-line arguments: a vector size and a step size and performs the corresponding summation. In the example, the vector size is fixed at 1000, but make this an argument so you can easily vary it. Using command-line parameters makes it easier to automate your experimentation because it allows you to easily execute your code from a script under different experimental conditions. The C++ code should output the time it takes to do the memory allocation and the time it takes to do the summation separately.

2. Now write a script that performas a sequence of experiments with different step lengths to see if you can replicate the results of the experiment shown in Lecture 2. The script should perform your entire set of experiments and collate results with one single command. In general, such a wrapper script may or may not be written in the same language as the code being executed (in many cases, it will not). When the code to be experimented with is written in C/C++, two natural options for implementing the wrapper are `bash` (along with Unix command-line utilities) and `python`. The advantage of `bash` is that it is custom built for exactly this purpose. However, it does have a bit of a learning curve and Python may be more comfortable for you.

   A bash script might like something like this.

   ```
   #!/bin/bash

   for steps in 1 2 4 8 16 32 64 128 256 512 1024; do
     echo "Performing experiments with step size $steps"
     ./summation $steps 10000
   done
   ```

   To run, save the script in a file called something like `experiment.sh` and then make it executable with

   ```
   chmod u+x experiment.sh
   ```

   Finally, just execute it on the command like:

   ```
   ./experiment.sh
   ```

To parse the data, we can use something like the `awk` command, as follows (this is just an exmaple and must be adjusted for the form of your output):

```
./experiment.sh | awk '($1 == "LOOP") {print $3 " " $4;}' | tee output.txt
```

This command runs the previous script, pipes the output into `awk` and then finally pipes the output into the `tee` command, which prints it both to a file and the screen. Alternatively, we could do something like

```
./experiment.sh | cut -d " " -f 3-4 | tee output.txt
```

This data could then be dumped into a graphing/charting utility, such as `gnuplot`, for example.

Using `python` is a bit more powerful. Here, we can process that data and dump int into a graphing utility directly.

```
import subprocess
import sys
import matplotlib.pyplot as plt

sizes = [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
results = {}
sys.path.append("/home/ted/Courses/IE407/ProblemSet0/")
for i in sizes:
   print ("Running with step size %d" % i)
   c = subprocess.run(["summation.exe", "%d" % i, "10000"], stdout=subprocess.PIPE)
   results[i].append(float(c.stdout.decode('ascii').split()[6]))

for i in steps:
   plt.plot(steps, results)

plt.legend()
plt.show()
```

3. Write a simple Makefile with targets for building your simple program from source and for running the experiments.

4. Add your source code, your script, and your Makefile to your git repository in the folder `Homework0`.