CS 240: Programming in C
Fall 2020
Homework 1

Prof. Turkstra

Due September 9, 2020 9:00 PM

1 Goals

The purpose of this homework is to provide you with a light introduction to C and program compilation.

2 Getting Started

1. Log in to one of the Purdue CS Linux systems. These include data.cs.purdue.edu and borgNN.cs.purdue.edu where NN=01,02, etc
2. Inside the cs240 directory, setup a hw1directory by running the following commands...

```
$ cd cs240
$ git clone ~cs240/repos/$USER/hw1
...
$ cd hw1
```

3. This creates your hw1 directory, and also copies what is called a **Makefile** into it. The Makefile for this homework will only commit (submit) your work for grading. In the future, we will provide you with a more advanced version that will compile and link the code automatically. For this lab, we want you to familiarize yourself with gcc.
4. Copy `hw1.c`, `hw1.h`, `hw1_main.c` and `hw1_test.o` from `~cs240/public/homework/hw1` into your hw1 directory. (Since the repository was provisioned more extra files, please copy the above files manually to the latest version - August 31.)
5. Complete hw1.c per the instructions and descriptions below. To compile your programs, you should do the following:

```
$ gcc -Wall -Werror -std=c99  -c hw1.c
$ gcc -Wall -Werror -std=c99  -c hw1_main.c
$ gcc -o hw1_main hw1_main.o hw1.o
```

And then run `hw1_main.c`:

```
$ ./hw1_main
```

We have also provided a hw1_test.o file for you. Once you have tested your code using hw1_main.c, you can try to link hw1.o against hw1_test.o and run hw1_test to find out what your tentative score will be.

3 The Big Idea

In mathematics, "arithmetic series" is a sequence of numbers such that the common difference between the consecutive terms is constant. For instance, the sequence $5, 7, 9, 11, 13, 15, 17, \ldots$ is an arithmetic series with common difference of 2.

If the initial term of an arithmetic series is a_1 and the common difference is d, then the nth term of the sequence a_n is given by: $a_n = a_1 + (n-1)d$.

In general, the nth term of the sequence a_n can be also denoted by $a_n = a_m + (n-m)d$ where $1 \leq m \leq n$.

For example, an arithmetic series with an initial starting value $a_1 = 10$, a common difference $d = 3$ and a limit $n = 5$ results in the following sequence: $10, 13, 16, 19, 22$.

Your task for this homework is simple. You are going to write two functions that:

- Check if a sequence is a valid arithmetic series.
- Generate a arithmetic series when provided with a starting value, a limit and a common difference.

Examine `hw1.h` carefully. It contains declarations for the global variables that you'll use. It also contains prototypes for the functions you'll write.

4 The Assignment

Functions you will write

You will write the following functions:

```
int compute_arithmetic_series(int, int, int);
```

This function should generate a arithmetic series starting at the first integer argument. The number of elements to generate is passed as the second integer argument. The third integer argument is a common difference. Each generated number should be placed in the global array `g_arithmetic_array[]`. The function should then return the sum of all numbers computed. For example, if the function is passed the values 23, 10 and 5, then it should return 455, and the first ten elements of the `g_arithmetic_array` should look like:

```
g_arithmetic_array[0] = 23;
g_arithmetic_array[1] = 28;
g_arithmetic_array[2] = 33;
g_arithmetic_array[3] = 38;
g_arithmetic_array[4] = 43;
g_arithmetic_array[5] = 48;
```

```
g_arithmetic_array[6] = 53;
g_arithmetic_array[7] = 58;
g_arithmetic_array[8] = 63;
g_arithmetic_array[9] = 68;
```

This function should return `ARITHMETIC_ERROR` if the number of elements passed is less than or equal to zero. You should also ensure the number of elements does not exceed the size of the array.

To implement this function, set the zeroth element of `g_arithmetic_array` to the first integer argument passed to the function. Then write a loop that counts from 1 up to the second integer argument passed to the function. Inside that loop, calculate the next element according to the common difference which is the third integer argument passed to the function.

```
int check_arithmetic_series(int);
```

This function should verify that a valid arithmetic series, up to element $limit - 1$ (where *limit* is the first and only argument passed) is present in `g_arithmetic_array`.

This function should return `ARITHMETIC_ERROR` if the argument passed is less than or equal to zero. `ARITHMETIC_ERROR` should also be returned if the argument passed would result in reading values beyond the end of the array.

This function should return `ARITHMETIC_CORRECT` if the arithmetic series is valid.

This function should otherwise return the index number corresponding to the first incorrect element.

This function may be implemented in a couple of ways. You can either loop through the existing array, `g_arithmetic_array`, and calculate each value to check on the fly, or allocate a local array of size `ARRAY_SIZE` and propagate it with the arithmetic series starting at value `g_arithmetic_array[0]`, then loop through and compare the two arrays. We would prefer the first, as it has smaller memory requirements. It is, however, up to you for this assignment. There are probably other creative ways as well. As long as it works, we'll be happy.

4.1 Input Files

There are no input files for this assignment!

4.2 Header Files

We provide a header file, `hw1.h`, for you. It contains prototypes for each of the functions that you will write as well as #definitions for the constants. You should not alter this file. We will replace it with the original when grading.

4.3 Error Codes

There is a #define for `ARITHMETIC_ERROR` and `ARITHMETIC_CORRECT` in `hw1.h`.

These Standard Rules Apply

- You may add any `#include`s you need to the top of your hw1.c file;
- You may not create any global variables other than those that are provided for you. Creation of additional global variables will impact your style grade;
- You should check for any failures and return an appropriate value;
- You should not assume that `ARRAY_SIZE` cannot change.
- Do not look at anyone else's source code. Do not work with any other students.

Code Standard

A linter is available to help assess whether your code properly adheres to the code standard. It is not perfect. The authoritative determination for your grade is made according to the written standard, not whether or not you pass the linter. For example, the linter cannot determine if your variable names are meaningful. The final grade will be determined by graders.

The linter is located at `~cs240/bin/linter`. To run it, you should add this path to your PATH environment variable:

```
$ export PATH=$PATH:~cs240/bin
```

If you would like this change to be permanent, you can add it to your `.bashrc` file.

The linter is run simply as follows:

```
$ linter file.c
```

If you encounter any issues or bugs, please let us know.

Submission

To submit your program for grading, type:

```
$ make submit
```

In your hw1 directory. You can do this as often as you wish. We encourage you to submit your code as often as possible. Only your final submission will be graded.

5 Grading

The operation of your functions will be graded out of 100 points. The point breakdown will be determined by the test program.

The test program will be run many times when grading. It is your job to do the same. The lowest score will be your final grade.

This homework will also have a style grade based on 20 points, with 1 point deducted for each code standard violation found.

Your code must compile successfully using `-Wall -Werror -std=c99` to receive any credit. Code that does not compile will be assigned an automatic score of 0.

If your program crashes (e.g., segmentation fault) at any point during the testing process, your score will also be an automatic 0.