Breakout / Lab 03

More Simple Programs

Last Updated: February 1, 2017

Problem / Exercise

Write C++ programs to perform the following tasks. In many of the program descriptions below, example input and output is provided. **NOTE:** You don't need arrays to solve any of these problems. You should NOT use arrays to solve any of these problems. You should also not use string manipulation beyond extraction from an `std::stringstream`, if needed.

- `exclusive.cpp`: Let the user input an odd number of positive integers, space separated, on a single line (as seen below). Assume that each integer except for one, the *exclusive* integer, appears an even number of times in the input. Your program should output the exclusive integer. Remember, you do not need nor should you use an array to solve this problem. You should also not use string manipulation beyond extraction from an `std::stringstream`, if needed.

```
Enter integers: 2 1 55 3 2 1 4 4 2 2 55
     Exclusive: 3
```

- `primesum.cpp`: Let the user input a single integer n where $1 \leq n \leq 100000$. Your program should print the sum of the first n prime numbers. Remember, you do not need nor should you use an array to solve this problem. In the first example below, the first 5 prime numbers are 2, 3, 5, 7, and 11.

```
Enter integer: 5
    Prime sum: 28

Enter integer: 100000
    Prime sum: 62260698721
```

- `armstrong.cpp`: Write a program that prints out all Armstrong numbers, each on their own line. An Armstrong number is a three digit integer such that the sum of the cubes of its digits is equal to the number itself. For example, 371 is an Armstrong number since $3^3 + 7^3 + 1^3 = 371$. Remember, you do not need nor should you use an array to solve this problem.

- `bitsum.cpp`: Let the user input a single integer n where $1 \leq n \leq 18446744073709551615$. Your program should print the number of 1 bits in the binary representation of n. Remember, you do not need nor should you use an array to solve this problem. In the example below, the binary representation of 115 is 1110011. Remember, you should not use string manipulation beyond extraction from an `std::stringstream`, if needed.

```
Enter integer: 115
      Bit sum: 5
```

- `endtime.cpp`: Write a program to read two integers with the following significance. The first integer value represents a time of day on a 24 hour clock, so that 1245 represents quarter to one mid-day, for example. The second integer represents a time duration in a similar way, so that 345 represents three hours and 45 minutes. This duration is to be added to the first time, and the result printed out in the same notation, in this case 1630 which is the time 3 hours and 45 minutes after 12.45.

```
Enter in a start time: 1415
  Enter in a duration: 50
         End time is: 1505
```

1 Group Brainstorm

You are NOT allowed to use the computers during this time.

Breakup into groups based on your seating and brainstorm about how to solve the problem or exercise. Make sure everyone understands the problem, and sketch out potential ways to move towards a solution. Perhaps something that was discussed during lecture might be useful?

2 Submit Individual Brainstorm

You may use a computer from this point forward.

Login to eLC and submit a version of your group's brainstorm, written in your own words. You may add additional information if you want. You need to write enough in order to convince the grader that you understand the problem or exercise and that you have a plan for moving forward towards a solution. Please include the last names of the other people in your group in your submission. The brainstorm submission should be available on eLC in your assignment dropbox. You should submit your individual brainstorms before the end of your breakout period. **NOTE:** Submissions that do not include an individual brainstorm will not be graded.

3 Some Nonfunctional Requirements

Your submission needs to satisfy the following nonfunctional requirements:

- **Directory Setup:** Make sure that all of your files are in a directory called `LastName-FirstName-lab03`, where `LastName` and `FirstName` are replaced with your actual last name and first name, respectively.
- **Documentation:** All classes, structs, and functions must be documented using Javadoc (or Doxygen) style comments. Use inline documentation, as needed, to explain ambiguous or tricky parts of your code.
- `Makefile`**:** You need to include a `Makefile`. Your `Makefile` needs to compile and link separately. That is, make sure that your `Makefile` is setup so that your `.cpp` files each compile to individual `.o` files. This is very important. The expectation is that the grader should be able to type `make clean` and `make` to clean and compile/link your submission, respectively.
- **Standards & Flags:** Make sure that when you compile, you pass the following options to `g++` in addition to the `-c` option:

  ```
  -Wall -std=c++14 -g -O0 -pedantic-errors
  ```

  Other compiler/linker options may be needed in addition to the ones mentioned above.
- `README`**:** Make sure to include a `README` file that includes the following information presented in a reasonably formatted way: i) your Name and 810/811; ii) instructions on how to compile and run your program; and iii) reflection section. Make sure that each line in your `README` file does not exceed 80 characters. Do not assume line-wrapping. Please manually insert a line break if a line exceeds 80 characters.
- **Compiler Warnings:** Since you should be compiling with both the `-Wall` and `-pedantic-errors` options, your code is expected to compile without `g++` issuing any warnings.

4 Submission

Before your next breakout lab session, you need to submit your code. You will still be submitting your project via nike. Make sure your work is on `nike.cs.uga.edu` in a directory called `LastName-FirstName-lab03`. From within the parent directory, execute the following command:

```
$ submit LastName-FirstName-lab03 cs1730a
```

It is also a good idea to email a copy to yourself. To do this, simply execute the following command, replacing the email address with your email address:

```
$ tar zcvf LastName-FirstName-lab03.tar.gz LastName-FirstName-lab03
$ mutt -s "lab03" -a LastName-FirstName-lab03.tar.gz -- your@email.com < /dev/null
```