Joanna Klukowska
joannakl@cs.nyu.edu

Project 2: Working with Object Code.

Due date: Apr. 3, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but **all work for all assignments must be entirely your own**. Any sharing or copying of assignments will be considered cheating.

Things to remember for all of the programs:

- The assembly code generated from the code that you write may not be exactly the same as the decompiled versions of the posted files. It is sufficient that the functions perform the same tasks for all inputs.
- The program `project2` tests your functions on a limited set of inputs. The fact that your implementation passes the test does not guarantee that the code that you wrote is correct.
- If you are allocating memory using `malloc`, you need to deallocate it using `free`. Proper memory management is always a requirement in this course.

Project Description

In this lab, we give you 5 object files, problem1.o, problem2.o, ..., problem5.o, and withhold their corresponding C sources. Each object file implements a particular function (e.g. problem1.o defines function problem1). The later objects files contain an additional function. We ask you to understand the x86-64 assembly code for each of these functions and figure out what that function tries to do. Your task is to write the corresponding C function that accomplishes the same thing for each of the functions.

The problems are numbered in order of difficulty. You should attempt to solve them in that order. But you can jump around and work on the problems in any order you wish (this is definitely a good stategy if you are stuck on a particular problem).

Reading the Assembly

The object files whose assembly code you seek to understand are in the `objs/` subdirectory. Suppose you set out to figure out what function `problem1` (implemented in objs/problem1.o) does. Here are two approaches to do this. You should use them both to help uncover the mystery.

Using `objdump`

Disassemble the object files. Read the assembly and try to understand what the function tries to achieve. To disassemble `problem1.o`, run

```
$
$ objdump -rd problem1.o
```

Using `gdb`

You can build an executable program from the object files in the `objs` subdirectory. To do so, run

```
$ make clean
$ make project2_given
```

This produces the program `project2_given`. You can run this program in `gdb`. This will/may help you to determine how the other functions are called and what arguments are passed to them.

Writing Your C Functions

The repository contains all files you need to work on this project. There are the problem files `problem1.c` ... `problem5.c` and the header file `project2.h`. You need to develop your code within those files.

Joanna Klukowska
joannakl@cs.nyu.edu

To compile each function use the provided `Makefile` as follows:

```
$ make clean
$ make problem1.o
```

This should generate your object code for `problem1.c`. For other problems, simply replace the number 1 with the number of the problem.

Test Your Solution

After you finish each function you can test its correctness (or at least partially test its correctness) as follows:

```
$ make clean
$ make project2
$ ./project2
problem1 passed the initial test
problem2 passed the initial test
problem3 passed the initial test
problem4 passed the initial test
problem5 passed the initial test
$
```

This suggest that all functions run correctly on these limited initial tests.

Note: Passing the test does not guarantee that you will get a perfect grade (i.e. your implementation is not necessarily correct). During grading, we will manually examine your source code to determine its correctness.
(If you look carefully at the code for `project2` *you could figure out how each test works and how to write you functions to satisfy the particular test that* `project2` *program uses (by simply returning what the test expects to be returned). DO NOT DO THAT! We will use a different test program to validate your functions.)*

Questions

Post any questions you have regarding this assignment to Piazza under the ”homeworks” topic.

Committing and Pushing Your Work

You should commit your code regularly to the repository on GitHub (this is your backup and a proof to us that you are working on it). At the least, you should commit your work after every single problem. You are **required** to make at least 5 commits (including the push to the remote repository) before the due date for the project.

Please, run `make clean` before committing the changes so that the object files are not uploaded together with your sources.