ICS Lab1-DataLab

`DEADLINE: 2020-10-9 23:59:59`

Download

Download `datalab-handout.tar` from `elearning`. For honor-class students, download both `datalab-handout.tar` and `datalab-handout-honor.tar`.

Then execute this in terminal:

```
# make sure you are in the directory where you download the .tar file
tar xvf datalab-handout.tar
```

Prerequisite

1. Make

We will use the command `make` in this lab.

Try this in terminal:

```
cd datalab-handout/
make clean
```

If some error occurs like this:

```
make: Command not found
```

Then you have to install some packages related to `make`. Execute this in terminal:

```
sudo apt-get update
sudo apt-get install libc6 libc6-dev libc6-dev-i386
```

Then execute this in terminal:

```
make clean
make
```

If there occurs any error here, please contact the TAs.

2. Permission of the given executable file

Execute this in terminal:

```
# make sure you are in datalab-handout/ directory
./dlc -v
```

If some error occurs like this:

```
./dlc: Permission denied.
```

Then execute this in terminal:

```
chmod +x ./dlc
./dlc -v
```

Introduction

The purpose of this assignment is to become more familiar with bit-level representations of integers and floating point numbers. You'll do this by solving a series of programming "puzzles." Many of these puzzles are quite artificial, but you'll find yourself thinking much more about bits in working your way through them.

Lab Guide

- You should complete the functions in `bits.c` (and `bits-honor.c` for honor-class students).
- Requirements for each function is listed in `bits.c`, including:
  - Set of valid operators
  - The number of operators
  - The range of constant numbers
  - Variable types
  - Whether control statements like `if` are allowed
- You are not allowed to modify any files other than `bits.c` without TA's permission.
- Commands for Testing:
  - `./dlc bits.c` shows if there exists any mistakes. Before your submission, you should confirm that there is no output when you execute this command.
  - `./dlc -e bits.c` shows the operators you use in each function.
  - ```
    make clean
    make
    ./btest
    ```

    This is the test for correctness of each function.
    ```

- You can make use of `./ishow` and `./fshow`

Submission

Content

You should submit `bits.c` and your report.

The report should contain the following:

- The snapshot of executing command `./dlc -e bits.c` and `./btest`
- Clear, tidy description of your implementation of the functions
- References (also list the references in `bits.c`)
- Anything you want to express after completing this lab (optional)
- Suggestions or complaint to TAs (optional)

Format

The format of your report can be `.pdf`, but should not be `.doc`.

After finishing your `bits.c` and your report, you create a directory named `ID` and copy your `bits.c` and report to this directory. The directory should contain `bits.c` (and `bits-honor.c` for honor-class students) and your report (may be a single file or a directory).

Then execute the following command in terminal:

```
# Take myself as an example.
# I'm now in directory 18307130024/. Make sure you are in the right place.
cd ..
tar cvf 18307130024.tar 18307130024/
```

Please submit this `.tar` file to `elearning`.

Deadline and overdue punishment

DEADLINE: `2020-10-9 23:59:59`

One day of overdue submission means 20% loss in the total score in this lab. TAs treat the time period of less than a day as a day.

Scoring

The scoring for this lab is 71% for **correctness** and 29% for your **report** and **coding style**.

NEVER use the search engine at once!

NEVER copy others' code!

CHEATING MEANS ZERO

TA's feeling

- `C` is different from `C++`. All local parameters within an action scope are required to be declared at first.
- Start from a simple example may help.
- In integer tests, `if else`, `==` and `!=` are not allowed. But you can use them to clearly express your thoughts and then replace them with legal operators.

- Bits are really interesting!

Reference

- ics-fudan-2018
- Stanford Bithack