# Sample Midterm: CMPT 127 Summer 2017

## Setup

- Check that you have a working copy of your gitlab repository. If not, clone your gitlab repository for this course. It should be `CMPT127-1174-username` where `username` is your SFU username.
- Before you do anything new in your repository first pull any changes to the local repository from the gitlab server by typing in `git pull` in your terminal window.
- **Create a directory called `sm` in your repository (which stands for *sample midterm*). All the files you write must be inside this directory.**
- For each file you add below you must make sure you `git add file`, then `git commit -m "commit message"` and finally do a `git push` to send the files to the gitlab server. Before you leave your terminal make sure you do a `git status` to make sure you have committed and pushed your midterm answers.
- You are allowed to look at your own lab programming assignments and the system man pages. Nothing else.

### (1) Task 1: Read and print

- Write a C program called `t1.c` with the following requirements.
- The program should take two command line arguments (using `argv`): the first argument (called `number`) is a floating point number and the second argument (called `count`) is an unsigned integer. For example if you compiled your program to a binary `t1` then it should be run as follows:

```
cc -Wall -o t1 t1.c -lm
./t1 5.3 10
```

- The first floating point number (called `number`) should be converted into a `double` type by using `atof()`. Then it should be rounded to an integer using the `round()` function. The program should then print out the integer conversion of the first argument and print the second argument using the following format:

```
./t1 5.3 10                  ./t1 1.9 15
n=5 count=10                 n=2 count=15
```

- There should be no extra spaces in the output and end with a newline.

### (2) Task 2: Print bars

- Write a C program called `t2.c` with the following requirements.
- The program should take two command line arguments (using `argv`): the first argument (`number`) is a floating point number and the second argument (`count`) is an unsigned integer. This part is identical to `t1.c`.
- Convert the first argument (`number`) into an unsigned integer. Let us call this integer `n`.
- On each line of the output, print out a sequence of `n` `#` characters followed by a newline. The number of lines to print is specified by the second argument `count` given to the program. e.g.

```
cc -Wall -o t2 t2.c -lm       ./t2 8.1 4
./t2 1.9 3                    ########
##                            ########                        ./t2 4 1
##                            ########                        ####
##                            ########
```

### (3) Task 3: Print a hailstone sequence

- Write a C program called `t3.c` with the following requirements.
- The program should take two command line arguments (using `argv`): the first argument (`number`) is a floating point number and the second argument (`count`) is an unsigned integer. This part is identical to `t1.c`.
- Convert the first argument (`number`) into an unsigned integer. Let us call this integer `n`.
- On each line of the output, print out a sequence of # characters. The first line contains `n` # characters.
- Each subsequent line should have a sequence of # characters equal to the output of the following formula:
  - If `n` is even, divide it by 2 to give `n = n/2`.
  - If `n` is odd, multiply it by 3 and add 1 to give `n = 3*n+1`
- A number is even if the number mod 2 (`%` is the mod operator) gives zero.
- Continue printing a sequence of `n` # characters using the new value of `n` computed using the above formula.
- The program stops printing when the number of lines is equal to the second argument `count` given to the program. e.g.

```
cc -Wall -o t3 t3.c -lm
./t3 5 5
#####
################
########
####
##
```

```
./t3 5 12
#####
################
########
####
##
#
####
##
#
####
##
#
```

```
./t3 11 6
###########
##################################
#################
####################################################
##########################
#############
```

**(4) Task 4: Print a hailstone sequence (part two)**

- Write a C program called `t4.c` with the following requirements.
- The program should take two command line arguments (using `argv`): the first argument (`number`) is a floating point number and the second argument (`count`) is an unsigned integer. The program should print out a sequence of # characters on each line followed by a newline based on the equation provided in Task 3. So far this task is identical to `t3.c`.
- The difference in this task is that you should stop printing the sequence of # characters when you observe the sequence 4, 2, 1. If you reach `count` lines before you observe 4, 2, 1 then you should print only `count` lines. e.g.

```
cc -Wall -o t4 t4.c -lm
./t4 5 2
#####
################
```

```
 ./t4 5 1000
#####
################
########
####
##
#
```

```
./t4 11 1000
###########
##################################
#################
####################################################
##########################
#############
########################################
####################
##########
#####
################
########
####
##
#
```

- Also try `./t4 27 1000`. It should print 112 lines of output. Verify that the last three lines have 4, 2 and 1 # characters respectively.