Programming Assignment 1

Due Date: Wed. 09/26

Please read these instructions carefully. You will be graded based upon meeting all requirements stated in this assignment.

Background

The producer - consumer problem is a classic programming exercise in computer science. It has both high practical benefit as well as a straight-forward implementation. This problem is composed of two parts. First, a "producer" creates some sort of workload, storing instructions or intermediate work in a queue. Second, a "consumer" reads this queue, performs the desired work and produces appropriate output.

Problem Statement: a serial producer - consumer

Using a collection of invertible functions (provided), implement a serial version of producer consumer. This program will serve as a reference and also provide a basis for future multi-threaded implementations.

- You will be provided with the following four functions in a separate .o file, which you will need to link with your .c program to produce an executable a.out file.

  uint16_t transformA(uint16_t input_val)
  uint16_t transformB(uint16_t input_val)
  uint16_t transformC(uint16_t input_val)
  uint16_t transformD(uint16_t input_val)

- Your main program should declare an array of the following type in which to store your work queue:

```
struct work_entry
{
    char cmd;
    uint16_t key;
};
```

- For this version of your program, use a work queue size of 5. For your future design consideration, this value will change in later program versions.
- The main portion of you program should first run the producer function.
- Each time the producer runs, if any entries were made into the work queue then the main program should run the consumer function.
- After the consumer function runs, the main program should resume the producer unless a *cmd* of X has been read from the input file.

producer

The producer section of your program will:

- read all input from standard input
- for each *cmd* / *key* pair read from the input file, validate the input as follows:
  - valid values for *cmd* are { A, B, C, D, X }. Input lines containing other values for *cmd* should be discarded.

  - valid values for *key* are [0 - 1000]. Extra leading zeroes may or may not appear in the *key* field. Input lines containing other values for *key* should be discarded.

- for each valid *cmd* read, the producer should call the corresponding transformX() function (where X $\in$ { A, B, C, D }) to create an encoded key.
- the producer should then create a work buffer entry containing the current value of *cmd* along with it's encoded key.
- the producer should stop upon filling the work queue or upon reading a *cmd* of X.

consumer

The consumer section of your program will:

- process the work queue until empty
- for each *cmd* $\in$ { A, B, C, D } read from the work queue, call the corresponding transformX() function (where X $\in$ { A, B, C, D }) to decode the key.
- the consumer should then print (on standard output):
  - the queue position (0 - 4) of the current work item being processed;
  - the *cmd* for the current work item
  - the encoded key (from the queue)
  - the decoded key (computed in the consumer)

Input Data Format

Each line of the data file contains the following:

- char *cmd*
- white space
- uint16_t *key*
- newline

There may or may not exist additional blank lines or other entries after a line containing an 'X' command. Those lines should be ignored.

Instrumentation

We will use multiple methods to instrument the run-time of your initial serial program.

- Measure and report the run time of your producer and consumer modules using the Unix time(2) and clock(2) system calls.
  - `http://www.cplusplus.com/reference/ctime/time/?kw=time`
  - `http://www.cplusplus.com/reference/ctime/clock/?kw=clock`
- You will need to measure each run of the producer and each run of the consumer, accumulating the run times into a single total for each module.
- Also, when executing your program, use the Unix time(1) utility to report elapsed clock, user and system times.

Program Output

A suitable example format for your output would resemble:

```
Q:0     A       0       7
Q:1     B       2       29
Q:2     C       4       123
Q:3     D       8       503
Q:4     A       16      23
Q:0     B       32      63
Q:1     C       64      63
Q:2     D       100     411
Q:3     A       200     207
Q:4     B       256     287
Q:0     C       512     639
Q:1     D       1000    535
```

Report Requirements

Run your program against all of the test files provided, providing the following for each test file:

- The first 50 and last 50 lines of your program output.
- The 'real' and 'user' time reported by time(1).
- The total runtime of your producer and consumer modules as reported by time(2) and clock(2).

Testing & Submission Instructions

You are responsible for completely testing your program. Your program must compile and run correctly on all evaluation data.

- Submit your program files from an Owens login node using the OSC submit system: `https://www.osc.edu/resources/getting_started/howto/howto_submit_homework_to_repository_at_osc`.
- Test input files will be made available on Owens in the /project file system.
- Submit your report in .pdf format via Carmen.