CSE3080-3: Data Structures (Spring 2020)
Homework 2: Tower of Hanoi

Handed out: April 22, Due: May 3, 11:59PM (KST)

1. Problem Description

In this homework, you are going to implement a program which solves the "Tower of Hanoi" puzzle. We have three towers, tower A, tower B and tower C. Initially n disks are placed in tower A. Our task is to move all the disks to tower C. The puzzle has the following rules.

- The disks are numbered from disk #1 to disk #n. The disk with a smaller number is larger in size. (Disk #1 is the largest disk.)
- You cannot place a larger disk on top of a smaller disk.
- When removing a disk from a tower, you can only pick up the disk at the top.
- When adding a disk to a tower, you can only place the disk at the top.
- You can only move one disk at a time.

※ You can refer to lecture #07 to further understand the problem.

The purpose of this homework is to design an algorithm using proper data structures in order to solve a problem. Also, we would like to understand how the running time increases as we increase the input size.

2. Your task and requirements (Read Carefully!)

(1) You should write a single C program, named **hw2**.

(2) The program takes one command-line argument, which is the number of disks. In other words, the user should run the program like this:

```
./hw2 10
```

The first argument is 10, which means we are going to move 5 disks. In order to get command-line arguments, your main function should now take arguments **argc** and **argv**.

(3) If the number of command-line arguments is not 1 (zero or more than 1), then you should print a usage string on the display and exit the program. The usage string looks like this:

```
usage: ./hw2 number-of-disks
```

If the user correctly gives one command-line argument, you can safely assume that the user will correctly enter a natural number. You don't need to implement error routines for errors such as user entering a character as number of disks.

(4) Your program should print lines to the output file named "hw2_result.txt". First, you should write a line for each move. The printed lines should be like the following:

```
...
MOVE DISK #7 FROM A TO B (moves: 89)
MOVE DISK #8 FROM A TO C (moves: 90)
MOVE DISK #7 FROM B TO C (moves: 91)
...
```

Follow the format shown here.

(5) At the end of the program when all moves are done, you should print the following line to the output file.

```
Total number of moves for 25 disks: 33554431
```

(6) You should also print the following line which indicates the running time in seconds.

```
Running time: 4.202713 seconds
```

You can use the **clock()** function provided by the library in order to measure running time. In order to use the function, you should include <time.h>.

(7) Similar to hw1, you should write a Makefile. The TA will build your code by running "make". It should create the binary file, "hw2".

※ Test-run the given binary code and write your program so that it produces the same result as the given binary code.

3. Submission

You should submit your source codes and the Makefile. (You do not need to submit compiled binary files.) Combine your files into a zip file named cse3080_hw2_**20190001**.zip. The red numbers should be changed to **your student ID**. Submit your files on the cyber campus.

4. Evaluation Criteria

(1) Your program should compile well to produce the binary file.

(2) If the user does not give correct number of arguments, your program should print the usage message on the display and terminate.

(3) When the user correctly executes the program with a command-line argument, your program should produce the result file "hw2_result.txt".

(4) The result file should correctly list out the disk moves, along with the total number of moves and running time. The contents of your output file should be the same as the output file created from running the given binary file, except the running time. (The running time could be different each time you run the program.)

5. Notes

- You must write your own code. You may discuss ideas with other students but must not copy their work. Similarly, you can find ideas on the Internet, but must not copy the code from the sources obtained from the Internet.

- Our department uses a software that detects duplicate codes. Since the software uses an assembly code level detection, just changing the variable names will not bypass the software. Make sure you write your own code from the scratch. Then, you will have no problem.

- Explained earlier, but your program should compile and run on the **cspro** machine. Make sure you check before you submit your work. The TA will download your code, run "make" and execute your binary files to check if they produce correct outputs.

6. Late Policy

10% of the score is deducted for each day, up to **three days**. Submissions are accepted up to three days after the deadline.