CS 558: Computer Systems Lab (2020)

Assignment Set I

- **Assignments will be evaluated by the TAs.**
- **You should submit report (for answering non-code related questions, if any), complete source codes and executable files (whichever applicable).**
- **All codes must be properly documented and good code writing practice should be followed (carry marks).**
- **Copying is strictly prohibited. Any case of copying will automatically result in F grade for the whole course, irrespective of your performance in the other parts of the lab.**
- **Submission deadline: February 09, 2020.**
- **Marks distribution: 20, 35, 45.**

1. Personal Dictionary – *Basic concepts on shell scripting*: (20 Marks)

Suppose you want to build your personal dictionary. Whenever you get time or you find a new/interesting word, you want to include it in the dictionary. The newly added word should be included at its appropriate position, along with its meaning. Implement the dynamic dictionary using shell scripts considering the following points.

I. Assume that your personal dictionary is a text file (mydictionary.txt).
II. Your input should have two parts: the 'new word' and the 'meaning' of the word.
III. Make your script capable of checking whether a word exists in your dictionary or not.
IV. If the word already exists in the dictionary, it should be displayed along with the stored meaning.
V. If the word does not exist, the new word should be added to the file 'mydictionary.txt' at the appropriate position.
VI. The meaning of the new word should also be added with it separated by a colon.
VII. At any instance of time, if you want to view the entire dictionary or any particular word, your script should allow that.

Sample input and output:

Input Command:

```
./mydict_add.sh
```

Output:

```
Enter the new word: ample

Enter the meaning: enough or more than enough; plentiful.

The word has been successfully added to your dictionary!!


Enter the new word: apple

Enter the meaning: the round fruit of a tree of the rose family, which typically
has thin green or red skin and crisp flesh.
```

```
The word already exist in your dictionary.
```

Input Command:

```
./mydict_search.sh
```

Output:

```
Enter the word: ample

Ample: enough or more than enough; plentiful.


Enter the word: ambiguous

The word does not exist in your dictionary.
```

2. Implementation of Shell Commands - *Concepts on basic shell commands*: (35 Marks)

Write a C program to simulate a bash shell with your own shell (eg: name_shell) that can implement the following Linux commands with one option if mentioned with it.

(A) Basic terminal navigation commands:
   a. ls (option: -a)
   b. cd

(B) Displaying the file content on the terminal:
   a. cat (option: -n)

(C) File and directory manipulation commands:
   a. mkdir (option: -m)
   b. cp (option: -u)

(D) Sort and filter data commands:
   a. sort (option: -r)
   b. grep (option: -n)

Note that:

I. **name_shell** should look exactly like normal linux terminal, where you can enter the commands from standard input followed by its arguments.
II. **name-shell** prompt should look exactly like linux shell prompt showing its current working directory. (eg: user@system_pc:~/Assignment1$).
III. Any illegal command or unrecognized argument should show a meaningful error to the user without crashing **name_shell**.
IV. **name_shell** should also support **exit** command to smoothly exit the shell by printing a meaningful message without forcefully exiting the process.
V. **name_shell** should also support **up** and **down** arrows and should implement command history. (**Hint: Use readline function in C).
VI. For **grep** command implement a minimum of 4 arguments.

3. 'Snakes & Ladders' Game - *Process creation and Inter-process communication (Using PIPES)*: (45 Marks)

In this assignment, you require to implement ***Snakes and Ladders board game*** using C programs. The aim is to implement a Gameboard (G) and two or more players. The players participating in the game are required to communicate with the Gameboard using pipes. The game proceeds as follows:

(A) The main program Gameboard (G) having numbered, gridded squares maintains the current position of all the players. The Gameboard has several numbers of "ladders" and "snakes", each connecting two specific board squares. The user should take number of gridded squares "N", number of players and a filename (containing the "ladders" and "snakes" information) as input using the command line arguments.

(B) The Gameboard can have any number of "ladders" and "snakes". The "ladders" and "snakes" information in the file uses the following file entry format. The value of the ***Type field*** should be either "L" or "S" representing "ladders" or "snakes", respectively. In case of a ladder entry, the starting gridded square value is less than the ending gridded square. In case of snake entry the starting gridded square value is greater than the ending gridded square. Your program should perform all the necessary validations while parsing the information from the input file.

(C) Before creating the players, the main program Gameboard (G) should create two unidirectional pipes that are meant for communicating between the main program and a player.

(D) After creating the pipes, the main program creates the required number of players (as provided in the input) by spawning the required number of child processes from the main program.

(E) Initially, the main program (G) randomly chooses one among the players as the first player (the choice of first player is decided randomly. Hence, it may change every time the game starts). The main program G then initializes all the players' position to Zero. The main program G selects the order of the players in a round robin fashion starting from the first player. The G starts the game from the first player.

(F) In every iteration, G sends a request message to the player using the respective pipe. Then, the player generates a random number between 1 and 6. The generated number is sent as a response message back to G over the respective pipe. Subsequently, the G performs the following operations.

a. Update the player position considering the "ladders" and "snakes" in the gameboard.
b. Display the current position of all the players.
c. Check whether the current position of the player is greater than the number of gridded squares "**N**". If yes, the main program G should display the current player as winner of the game and end the game.

(G) If the last response is 6, then repeat the iteration with same player. Otherwise, G performs the iteration with the next player in the round robin order.

Note that:

(I) After the game completes, all the pipes should be closed properly and the child processes should terminated gracefully. Thus, you are required to print appropriate exit message before the game ends.

(II) During evaluation, the TA will provide different test cases (i.e., file with necessary information) as input. Your program should execute with the given custom inputs.
An example of command line input and an input file format for a gameboard with 50 squares is provided here for your reference:

Command Line Input format:

```
./Gameboard 50 3 filename.txt
```

L	**5**	**27**
S	**49**	**25**
L	**9**	**44**
L	**23**	**42**
S	**37**	**20**
S	**21**	**6**