Homework #4

Question 1 (5 pt.) Shell

Extend data type `struct Command` with the following fields:

```
char *stdin_redirect;
char *stdout_redirect;
int background;
```

Fields `stdin_redirect` and `stdout_redirect` are strings indicating file names to which the standard input of the first sub-command and the standard output of the last sub-command will be redirected, respectively. Remember operators `<` and `>` are used in the shell to redirect the standard input and output, respectively. If any of these redirections is not provided by the user, the corresponding field in the structure will take a value of `NULL`.

Field `background` is a flag indicating whether the process should run in the background. If the user adds the `&` operator at the end of the line, `background` should be set to 1. Implement the following functions, or modify the previous versions of them from previous assignments:

a) `void ReadRedirectsAndBackground(struct Command *command);`

This function populates fields `stdin_redirect`, `stdout_redirect`, and `background` for the `command` passed by reference in the first and only argument. The function assumes that all other fields of the `command` structure have already been populated, as a result to a previous invocation to `ReadCommand`. The function should internally scan the arguments from the last sub-command in reverse order, extracting trailing `&`, `> file`, or `< file` patterns in a loop.

b) `void PrintCommand(struct Command *command);`

Extend this function to dump the value of `stdin_redirect`, `stdout_redirect`, and `background` for the command passed by reference in its first and only argument. Make sure that you consider the special cases where the redirection fields are set to `NULL`.

c) Write a main program that reads a command line from the standard input, creates the command object, and dumps all its fields, by invoking the previous functions. Upload your code in file `q1.c`. Here is an example of a possible execution:

```
$ ./q1
Enter a string: a b > out < in &
Command 0:
argv[0] = 'a'
argv[1] = 'b'

Redirect stdin: in
Redirect stdout: out
Background: yes
```

Question 2 (5 pt.)

Consider a computing system with the following workload, assuming that no job performs I/O operations:

- Job A arrives at 0ms, runs for 30ms.
- Job B arrives at 0ms, runs for 30ms.
- Job C arrives at 30ms, runs for 10ms.
- Job D arrives at 50ms, runs for 10ms.

Write a timing diagram representing the behavior of the system as presented in class until all jobs complete, using the scheduling algorithms below. In each case, calculate the average turnaround and response time, showing all intermediate calculations.

a) First-In First-Out (FIFO)

b) Shortest Time-to-Completion First (STCF)

c) Multi-Level Feedback Queue (MLFQ) with 4 queues, 10ms time slices, and 50ms priority reset period. For this timing diagram, use the representation given in class, and remember to:

   - Represent queues as rows starting with highest (Q1) to lowest (Q4) priority.
   - Add the current CPU time of each job as a subscript of the job name (e.g., A_{20} means that A ran for 20ms so far).
   - Circle the name of the currently running job.
   - Write a vertical line representing the beginning of each time slice, or the time when any other meaningful event occurred. Specify the kind of event at the bottom of these vertical bars, including job arrival, completion, or queue priority reset).

Attach a PDF file named `hw4.pdf` with your diagrams and calculations.