## Advance Systems Programming
## In-class Activity 1

In this activity, you are going to write a C program (in pseudocode) that simulates a shell. A shell controls a terminal and executes commands entered on the command line, which can be as simple as “execute a single program” or “execute multiple programs based on a specific set of rules”. It is important to note that the shell can execute any program if all necessary command line arguments of that program are provided along with the name of the program (executable).

Assume that the shell has the following structure:

```
Display the command line symbol ($)
While a command to read on the command line do
    Process the command line
    Display the command line symbol ($)
End while
```

You are going to use the system calls we have learned (fork, exec, wait/waitpid, pipe, close, dup/dup2) to implement the following 6 cases of “Process the command line”, where prog, prog1, and prog2 represent the names of some executables. You can assume that directories that host the executables are included in the PATH environment variable. Also, you can assume that the input has already been parsed and divided into tokens.

Case 1: Output redirection

$ prog args > file

Execute prog in a process and redirect the output on the standard output to the file

Case 2:

Case 2: Input redirection

$ prog args < file

Execute prog in a process and redirect standard input to the file as if the standard input is coming from the file

Case 3: Pipe

$ prog1 args1 | prog2 args2

Execute prog1 and prog2 in parallel (in two different processes) and redirect standard output of the process running prog1 to the standard input of the process running prog2.

Case 4: Sequential

$ prog1 args1 ; prog2 args2

First execute prog1 in a process and once it terminates execute prog2 in a process.

Case 5: Conditional AND

$ prog1 args1 && prog2 args2

First execute prog1 in a process and if the return status is 0 then execute prog2 in a process.

Case 6: Conditional OR

$ prog1 args1 || prog2 args2

First execute prog1 in a process and if the return status is not zero then execute prog2 in a process.

To get full credit, you should use the mentioned system calls appropriately. Note that you are not allowed to use the system function!