Example 1. This example gives sample output of running the `a1p3` program using `% a1p3 -1 < a1p3.data` where the input file uses three programs:

- Program `/usr/bin/timeout`: the program takes as argument a time DURATION, and a COMMAND. `timeout` starts executing the COMMAND, and kills it if it still running after DURATION (see `'man timeout'`).
- The `myclock` Bash script (cf. Part 2 of the assignment).
- Program `do_work.c` (cf. Part 3 of the assignment). Download, compile, and put the executable `do_work` in your work directory. The program takes as input a relatively large input file (to consume CPU time). To obtain one such a file, use `man bash > bash.man` to generate the Bash manual pages (≈ 370 KBytes).

The input file `a1p3.data`: The file has the following lines

```
# Comment lines and empty lines should be ignored

timeout 3 ./myclock outA
timeout 4 myclock outB
xyz
do_work bash.man
timeout 3 do_work bash.man 50
```

Note: You can make `myclock` executable by using the shell command: `'chmod +x myclock'`.

Sample output (edited for clarity):

- As the program processes the input lines, it shows its work:

```
print_cmd(): [timeout 3 ./myclock outA]
print_cmd(): [timeout 4 myclock outB]
print_cmd(): [xyz]
print_cmd(): [do_work bash.man]
print_cmd(): [timeout 3 do_work bash.man 50]
```

- After finishing the main loop, the program displays stored information about the forked children processes.

```
Process table:
0 [3307613: 'timeout 3 ./myclock outA']
1 [3307614: 'timeout 4 myclock outB']
2 [3307615: 'xyz']
3 [3307616: 'do_work bash.man']
4 [3307617: 'timeout 3 do_work bash.man 50']
```

- If `execlp` fails, the child process prints an error to this effect:

```
child (3307615): unable to execute 'xyz'
```

- If a child process terminates, and the program has been waiting for it, the program writes a line to this effect (include the child's exit status):

```
process (3307616:'do_work bash.man') exited (status= 0)
process (3307613:'timeout 3 ./myclock outA') exited (status= 31744)
process (3307617:'timeout 3 do_work bash.man 50') exited (status= 31744)
process (3307614:'timeout 4 myclock outB') exited (status= 31744)
```

- Before termination, the program writes the recorded times:

```
real:    4.00 sec.
user:     ... sec.
sys:      ... sec.
child user:    1.59 sec.
child sys:     2.22 sec.
```

Important: Don't forget to cleanup the running processes by using the command
`pkill -u $USER pattern`
where `pattern` can be a string that appears in the processes you would like to terminate, e.g., `pkill -u $USER clock`.
