Lab #5 – Process Control

Part I

Write a C program where child processes are forked in this way:

- the original process creates two child processes
- each process created from the original process creates two child processes

Each process is given a unique process number in range 1..7 in this way:

- the original process is numbered 1
- the two processes created by process 1 are numbered 2 and 3
- the two processes created by process 2 are numbered 4 and 5
- the two processes created by process 3 are numbered 6 and 7

Your program receives 7 arguments from command line. These arguments are expected to be natural numbers. Each process will go to sleep for some time after having created child processes if needed. The number of seconds a process numbered i is put to sleep is the ith argument.

The sleep time can be considered as to simulate the different computation time each process takes to carry out its subtask.

Before each process terminates, print out its process number and process id.

Sample output:

```
>>>>> a.out 1 2 3 4 5 6 7
process 1 terminated. pid=4860
process 2 terminated. pid=4861
process 3 terminated. pid=4862
process 4 terminated. pid=4863
process 5 terminated. pid=4864
process 6 terminated. pid=4865
process 7 terminated. pid=4866
>>>>> a.out 7 6 5 4 3 2 1
process 7 terminated. pid=4876
process 6 terminated. pid=4875
process 5 terminated. pid=4874
process 4 terminated. pid=4873
process 3 terminated. pid=4872
process 2 terminated. pid=4871
process 1 terminated. pid=4870
>>>>> a.out 2 8 3 10 9 1 5
```

process 6 terminated. pid=4940
process 1 terminated. pid=4935
process 3 terminated. pid=4937
process 7 terminated. pid=4941
process 2 terminated. pid=4936
process 5 terminated. pid=4939
process 4 terminated. pid=4938

Part II (optional)

If your previous solution is not written this way, try it: Use two nested for-loop statements to complete the assignment. Processes 2 and 3 are created in the outer for-loop, and processes 4, 5, 6, 7 are created in the inner for-loop.