CS2106 Introduction to Operating Systems
Lab 2

For this lab you will have to read ahead in Lecture 3. In particular you will have to read slides 30, 32, 34, 38 and 44.

Question 1 [2 marks]

Write a C program that takes a numeric argument n from the command line, and performs the following actions:

- It forks n child processes..
- Each child process starts an instance of the /usr/bin/xterm program.
- The parent process then waits for all the child processes to terminate. When run, your program should open n terminal windows. When you close the newly opened terminal windows, the parent program should terminate as well, giving control to the command prompt in the original window.

Question 2 [4 marks]

The pstree command prints the entire process tree of a Linux system. It takes the form: pstree -p -n -l

- where the -p argument will add process identifiers to the process names; -n will sort processes by their *pid* instead of by name, and -l will allow long lines of output (so it's good to increase the width of your terminal window to run this command. In what follows, you will be required to use this command to inspect the process subtree of the program that you create. Let us consider the following program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
  int main(int argc, char ** argv) {
  int n = atoi(argv[1]) ;
  int pid = getpid() ;
  int childpid = 0 ;
  printf("Superdaddy: %d\n", pid) ;
  while ( n > 0 ) {
      childpid = fork() ;
    if (childpid == 0) {
        printf("child %d has parent %d\n", getpid(), getppid()) ;
    } else if (childpid > 0) {
        printf("parent %d created child %d\n", getpid(), childpid) ;
    } else perror(strerror(errno)) ;
    n -- ;
    }
  sleep(30);
  return 0 ;

}
```

The program above takes a numeric argument and starts child processes in a specific pattern. Child processes, in turn, will start their own children, and so on, in a number that is proportional to the square of the numeric argument given on the command line. Each process will pause for 30 seconds before exiting, so there is a possibility to take a snapshot of the process tree using the pstree command. For that, you run the program above with the argument of your choice, and then you need to go (quickly) to a different terminal window and run the command pstree -p -n -l before the processes you just created had a chance to terminate. For instance, the following is the process subtree for a numeric argument of 3 (we're not giving the process tree of the entire system here, since it's too large -- you'll have to look for the subtree inside the big tree).

```
ex2(7669)─┬─ex2(7670)─┬─ex2(7672)───ex2(7674)
          │           └─ex2(7676)
          ├─ex2(7671)───ex2(7673)
          └─ex2(7675)
```

And here is the subtree corresponding to an argument equal to 4.

```
ex2(7682)─┬─ex2(7683)─┬─ex2(7684)─┬─ex2(7687)───ex2(7689)
                 │            │           └─ex2(7693)

                 │            ├─ex2(7685)───ex2(7692)

                 │            └─ex2(7686)
                 ├─ex2(7688)─┬─ex2(7690)───ex2(7697)

          │           └─ex2(7691)
          ├─ex2(7694)───ex2(7695)
          └─ex2(7696)
```

Moreover, here's the tree for an argument equal to 5.

```
ex2(7701)─┬─ex2(7702)─┬─ex2(7703)─┬─ex2(7704)─┬─ex2(7705)───ex2(7708)
                 │            │           │           └─ex2(7706)

                 │            │           ├─ex2(7707)───ex2(7710)

                 │            │           └─ex2(7709)

                 │            ├─ex2(7711)─┬─ex2(7712)───ex2(7731)

                 │            │           └─ex2(7713)

                 │            ├─ex2(7723)───ex2(7726)

                 │            └─ex2(7728)
                 ├─ex2(7714)─┬─ex2(7717)─┬─ex2(7718)───ex2(7725)
```

```
                    |           └─ex2(7720)

       |            ├─ex2(7722)───ex2(7732)

       |            └─ex2(7724)
       ├─ex2(7715)─┬─ex2(7727)───ex2(7729)

|           └─ex2(7730)
├─ex2(7716)───ex2(7719)
└─ex2(7721)
```

Please compile the program given above, and verify that the process tree has indeed the shape given above. Now, here is your task for this exercise.

Modify the program given above so that it would yield complete binary process subtrees, as shown in the figures below:

For a numeric argument of 3:

```
 ex2(7758)─┬─ex2(7759)─┬─ex2(7761)─┬─ex2(7763)
```

And for a numeric argument of 4:

```
ex2(7778)─┬─ex2(7779)─┬─ex2(7780)─┬─ex2(7782)─┬─ex2(7784)
|
|
|
└─ex2(7760)─┬─ex2(7762)─┬─ex2(7765)
|           └─ex2(7766)
└─ex2(7768)─┬─ex2(7769)
            └─ex2(7771)
|
|
|
|
|
|
|
└─ex2(7783)─┬─ex2(7785)─┬─ex2(7787)─┬─ex2(7789)
|           └─ex2(7767)
└─ex2(7764)─┬─ex2(7770)
            └─ex2(7772)
|           |           └─ex2(7788)
|           └─ex2(7791)─┬─ex2(7793)
|                       └─ex2(7795)
└─ex2(7781)─┬─ex2(7786)─┬─ex2(7790)
|           └─ex2(7798)
└─ex2(7800)─┬─ex2(7802)
            └─ex2(7804)
|
|
|
└─ex2(7799)─┬─ex2(7801)─┬─ex2(7807)
|           └─ex2(7792)
└─ex2(7794)─┬─ex2(7796)
            └─ex2(7797)
|           └─ex2(7808)
└─ex2(7803)─┬─ex2(7805)
            └─ex2(7806)
```

Submission deadline: *February 21, 23:59*, via IVLE workbin. Please submit into the CORRECT FOLDER.

Lab TAs will not be responsible for passing your script over if you upload into the wrong bin. YOU WILL RECEIVE 0 IF YOU UPLOAD INTO THE WRONG FOLDER.