Lab #4 – Process Control and File I/O

Part I

Read the following programs and write down the expected results. Then run the programs to check the results. Make sure you understand why. To pass the lab test, you may be asked for the results of similar programs. For this exercise, it is assumed that all *fork()* function calls are successful, and the part of code for checking whether a *fork()* call is successful or not is omitted.

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main() {

        fork();
        fork();
        fork();
        printf("%d\n", getpid());
}
```

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main() {

        if (fork() == 0)
                fork();
        else {
                fork();
                fork();
                printf("%d\n", getpid());
        }
}
```

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main() {
```

```
        if (fork() == 0)
                fork();
        else {
                fork();
                fork();
        }
        printf("%d\n", getpid());
}
```

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main() {

        if (fork() == 0)
                fork();
        else {
                fork();
                fork();
                exit(0);
        }
        printf("%d\n", getpid());
}
```

Part II

Use *fork()* to create a child process.

- The parent process checks the first command line argument. If it is not an integer number between 1 and 10, the parent process will print out an error message; otherwise, it will write the number into a file called *data.dat*.
- The child process reads the number from the file and prints out its factorial number. You need to make sure that the child process will read the file only after the parent has finished writing into it.

You must use system call I/O. No library I/O call is allowed.