ASSIGNMENT 1

A1_1] Consider the C function printf() on UNIX. Is printf() implemented by the OS, or by an application-level library? What system call does printf() make internally?

printf() function is implemented by an application level library in C, by the use of write() system call in C. A file needs to be opened for write operations. The write system call copies the data from the buffer to the opened file. If write() successfully writes data, it returns the number of bytes it wrote before it was interrupted.

A1_2] Why are mkdir, ln and rm implemented as separate user-level programs, while cd is implemented as a built-in command?

When the command $cd is executed, no new process is created to migrate to the other directory as in the case with other commands such as ls, ln, mkdir, etc. Instead the terminal itself executes this command. This is because, when a new process is created, child process inherits the directory in which the parent process was created. If the `$cd` command inherits the parent process' directory, then the objective of the command cd will never be achieved.

A1_5] What is the total number of processes at the end of the execution of the following program? Assume there is one process in the beginning that starts running at main. Also, assume that all system calls succeed.

```
main() {
fork();
fork();
fork();
}
```

Explain.

Total of 8 processes will be created. At beginning only 1 process P1 is present. After the first fork() call , its child process C1 is created. Second fork() call creates child process C2,C3 with parent processes P1,C1 respectively. Similarly the third fork() call creates child process C4,C5,C6,C7 with parent process P1,C1,C2,C3. Therefore we have total 8 process : P1,C1,C2…C7.

A1_6,7] Consider the following program:

```
main() {
      int fd;
      fd = open(“outfile.txt”, O_RDWR)
      fork();
      write(fd, “hello”, 5);
      exit(0);
}
```

Assume all system calls finish successfully on a uniprocessor system. Also, assume that a system call cannot be interrupted in the middle of its execution. What will be the contents of the outfile file, after all processes have successfully exited? Explain briefly.

The contents of outfile.txt will be : hellohello
“hello” will be written in the file twice as there is a fork() call in the program, which creates a child process. So the write call will run two times. Once for the child process and once for parent process.

A1_11] Write the pseudo-code for a program “cp" that takes two command-line arguments, say input-file and
output-file, copies the input-file to the output-file.

Syntax:
$ cp ifile ofile

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc, char *argv[])
{
   execl("/bin/cp", "cp", argv[1], argv[2],(char *)0);
   return 0;
}
```

To execute the program, we compile the code and pass the name of source and destination file in the command line as follows:

```
$ gcc A1_11.c
$ ./a.out source.txt dest.txt
```