LAB 3

1. Execute the following program and observe how fork system call works. Explain how the program works and how the output was obtained.
2. Execute all the programs multiple times. Do you get the exact same output (in the exact same order) every time? Explain why it is possible to get different outputs (in different order).

a. Program 1

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
void main()
{
        printf("I am a process with pid %d\n", (int) getpid());
}
```

b. Program 2

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
void main()
{
        int pid;
        printf("I am a process with pid %d\n", (int) getpid());
        pid = fork();
        printf("Fork returned %d\n", pid);
        printf("I am a process with pid %d\n", (int) getpid());

}
```

c. Program 3

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
void main()
{
        int pid;
        printf("I am a process with pid %d\n", (int) getpid());
        pid = fork();
        if(pid<0)
        {
                perror("Fork failed\n");
        }
        if (pid ==0)
        {
                printf("I'm the child process with pid %d\n", (int) getpid());
        }
        else if (pid>0)
        {
                printf("I'm the parent process with pid %d\n", (int) getpid());

        }

}
```

d. Program 4

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
void main()
{
        int pid;
        printf("I am a process with pid %d\n", (int) getpid());
        pid = fork();
        if(pid<0)
        {
                perror("Fork failed\n");
        }
        if (pid ==0)
        {
                printf("I'm the child process with pid %d\n", (int) getpid());
                exit(0);
        }
//Now you can confirm that the next is parent
                printf("I'm the parent process with pid %d\n", (int) getpid());


}
```

e. Program 5

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
void main()
{
        int childpid;
        int count1=0,count2=0;
        printf("Before it forks!!\n");
        childpid = fork();
        if (childpid ==0)
        {
                printf("This is a child process with pid %d\n", (int) getpid());
                while(count1 <10)
                {
                        printf("Child process: %d\n",count1);
                        sleep(1);
                        count1++;
                }
        }
        else
        {
                printf("This is a parent process with pid %d\n",(int) getpid());
                while(count2 <20)
                {
                        printf("parent process: %d\n",count2);
                        sleep(1);
                        count2++;
                }
        }
}
```

3. Compile the following program. Explain the need of *execlp* statement (you can read the man page of *execlp*). After running the program, what output did you get? Explain how the program works and how the output was obtained.

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

main()
{
      int pid;

      //
      if ((pid = fork()) < 0)
            exit(1);

      if (pid == 0){
            printf("Value of pid: %d\n", pid);
            execlp("/bin/ls", "ls", NULL);
            exit(1);
      }
```

4. Create the processes in the following hierarchy. Print the process id, parent process id, and a character ('A' in P1, 'B' in P2 and so on) for each of the processes.