Lab-12.

① Write a c/c++ program to emulate the Unix ln command.

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<string.h>
int main(int argc, char * argv[])
{
if(argc <3 || argc >4 || (argc ==4 && strcmp(argv[1],
      "-s")))
{
printf("Usage : ./a.out [-s] <org-file><new_link>\n");
return 1;
}
if(argc ==4)
{
if((symlink(argv[2],argv[3])) == -1)
printf("Cannot Create symbolic link\n");
else
printf("Symbolic link Created\n");
}else
 {if((link(argv[1],argv[2])) == -1)
 printf("Cannot Create hard link\n");
 else
  printf("Hard link Created \n");
 }
 return0;
 }
```

O/P

```
gcc lncmd.c
./a.out lncmd.c z
 Hard link Created.
```

```
./a.out -s lncmd.c zz
 Symbolic link Created
```

⑤ Write a c/c++ POSIX Complaint program that prints the POSIX defined Configuration options supported on any given system using features test macros.

```
#define _POSIX_SOURCE
#define _POSIX_C_SOURCE 199309L
#include<stdio.h>
#include<unistd.h>
int main()
{
#ifdef _POSIX_JOB_CONTROL
printf("System supports job Control \n");
#else
 printf("System does not support job Control \n");
#endif
 #ifdef _POSIX_SAVED_IDS
printf("System supports saved set-UID and saved
        set-GID\n");
#else
 printf("System does not support saved set-UID and
       saved set-GID \n");
 #endif
 #ifdef _POSIX_CHOWN_RESTRICTED
 printf("chown_restricted option is %d\n", _POSIX_CHOWN_RESTRICTED
#else
printf("System does not support chown_restricted option\n");
#endif
#ifdef _POSIX_NO_TRUNC
printf("Pathname trunc option is %d\n",_POSIX_NO_TRUNC);
#else
printf("System does not support system-wide pathname
        trunc option \n");
#endif
 #ifdef _POSIX_VDISABLE
```

```
printf("Disable character for terminal files is %d\n", _POSIX_
                                                    VDISABLE);
#else
 printf("System does not support _POSIX_VDISABLE \n");
#endif
 return 0;
}
```

O/P

```
gcc posix.c
./a.out posix.c
System supports job control
System supports saved set-UID and saved set-GID
chown_restricted option is 0
Pathname trunc option is 1
Disable character for terminal files is 0
```

O/P 8w 16/1/23

③ Write a c/c++ program which demonstrates interprocess communication b/w a reader process and a writer process. Use mkfifo, open, read, write and close APIs in

```
#include<sys/types.c>
#include<unistd.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<string.h>
#include<errno.h>
#include<stdio.h>
int main(int argc, char* argv[])
{
int fd;
char buf[256];
if(argc != 2 && argc != 3)
{
printf("USAGE %s <file> [<arg>] \n", argv[0]);
```