(1) Explain FIFO file API's. write a Program to implement the same.

Ans:- FIFO files are sometimes called named Pipes. Pipes can be used only between related Processes when a common ancestor has created the Pipe.
creating a FIFO is similar to creating a file. Indeed the pathname for a FIFO exists in the file system.
The prototype of mkfifo is

```
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
int mkfifo(const char* Path_name, mode_t mode);
```

Sucess it return 0 and failure it returns -1.

Program:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <string.h>
#include <errno.h>
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
if(argc!=2 && argc!=3)
  {
```

KNSIT

```
        cout<<"usage: "<<argv[0]<<" <file> |<arg>|";
        return 0;
    }
    int fd;
    char buf[256];
    (void) mkfifo(argv[1], S_IFIFO |S_IRWXU |S_IRWXG|
    S_IRWXO);
      if(argc==2)
    {
      fd = open(argv[1], O_RDONLY |O_NONBLOCK);
      while (read(fd, buf, sizeof(buf)) ==-1 &&
errno = EAGAIN
          sleep(1);
          while(read(fd, buf, sizeof(buf))>0)
          cout<< buf <<endl;
      }
      else
        {
        fd=open(argv[1], O_WRONLY);
        write(fd, argv[2], strlen(argv[2]));
      }
      close(fd);
    }
```

(2) Explain symbolic link file API's. write a program to implement the same.

Ans:- A symbolic link is an indirect pointer to a file, unlike the hard links which pointed directly to the inode of the file.

Symbolic links are developed to get around the limitations

of hard links.
- Symbolic links can files across file systems.
- Symbolic links can link directory files.
- Symbolic links always reference the latest version of the files to which they link.

The prototype of link is

```
int symlink(const char * org_link, const char * sym_link);
int readlink(const char * sym_link, char* buf, int size);
int lstat(const char* sym_link, struct stat * statv);
```

Program:

```
#include <unistd.h>
#include <sys/types.h>
#include <string.h>
#include <stdio.h>
int main(int argc, char * argv[])
{
char* buf[256], fname[256];
 if(argc == 4)
     return symlink(argv[2], argv[3]); /*create a symbolic
                                         link*/
else
    return link(argv[1], argv[2]); /*creates a hard link*/
}
```