5A) A path like a/b/c refers to the file or directory named c inside the directory name b inside the directory named a in the root directory /

open("a/b/c", O_RDWR)

a/b/c - path

O_RDWR - both read and write

It means that open the file in the path in a mode of read and write.

algorithm open

inputs: file name
type of open
file permissions

output: file descriptor

convert file name to inode (name) algorithm)
if (file does not exist or not permitted access)
return (error);
allocate file table entry for inode, initialize
count, offset;
allocate user file descriptor entry, set pointer
to file table entry;
if (type of open specifies turcate file)
free all file blocks;
unlock (inode);
return (user file descriptor);

5B) one direct block address points 1KB and we have 10 Direct block address. So it will take 10 kB. Address will take 4 bytes.

= 1 KB / 4 Bytes

= 1024/4

= 256 blocks

Single direct block take 256 * 1 KB = 256 KB

Double indirect block will take 256 * 256

$= 2^{16}$ KB

= 64 MB

Triple indirect block will take 256 * 64 MB

$= 2^{14}$ MB = 16 GB

Algorithm Bmap

input :- 1) inode

2) Byte offset

output :- 1) block number in file system

2) byte offset into block

3) byte of I/O in block

4) read ahead block number

Synopsis :- sector_t bmap (struct inode *inode, sector_t block);

BMap

```
{
    calculate logical block number in file from
    byte offset;
    Calculate start byte in block for I/O;
    calculate no. of bytes to copy to user;
    check if read-ahead applicable, mark inode;
    determine level of indirection;
    while (not at necessary level of indirection)
    {
        calculate index into inode or indirect block
        from logical block number in file;
        get disk block number from inode or indirect
        block;
        release buffer from previous disk read, if any;
        if (no more levels of indirection)
            return (block number);
        read indirect disk block;
        adjust logical block number in file [illegible]
        according to level of indirection;
    }
```

Description

Returns block number on the device holding the inode that is the disk block number for the block of file requested. That is, asked for block 4 of inode 1, the function will return the disk block relative to the disk start that holds that block of the file.

Structure Of a Regular file

The inode contains table of contents, that locates the file's data on disk. Each block on disk is addressable by a number. For more flexibility, kernal allocates file space one block at a time and allow the data in a file to be spread allover the system.