```
#include"include.h"
/* show the output based on client requirement
 * print the type of file and size */
int myfunc(const char *pathname,const struct stat *statptr,int type)
{
        //printf("name:%s\n",name);
char *ptr;
#if 1
        static char j=0;

        //if(((f_OP ==1) && strstr(statptr->d_name,f_OP_str) == NULL) ||
        if(((f_OP ==1) && strstr(name,f_OP_str) == NULL) ||
          ((s_OP==1)  && ( statptr->st_size < s_OP_size )))  /* return thr file if file not satisfied the user requiremet */
        {  return 0;}

        if((t_OP ==1)&& (((t_OP_CH =='d')&& !S_ISDIR(statptr->st_mode)) ||
                                        ((t_OP_CH =='f')&& !S_ISREG(statptr->st_mode))  ))
        {  return 0;}
#if 0
        if( (j !=0) && i==1 )
                {
                        printf("\t"); /* this is for allignment */
                  }
#endif
        printf("%-40s ",pathname);

        switch(type)
        {
```

```
case FTW_F:
        switch(statptr->st_mode & S_IFMT){
                case S_IFREG:printf("REGULAR FILE ");
                        break;
                case S_IFBLK:printf("BLK FILE ");
                        break;
                case S_IFCHR:printf("CHR FILE ");
                        break;
                case S_IFIFO:printf("FIFO FILE ");
                        break;
                case S_IFLNK:printf("LINK FILE ");
                        break;
                case S_IFSOCK:printf("SOCK FILE ");
                        break;
                case S_IFDIR:printf("DIR FILE ");
                        break;
                            }
          break;
case FTW_D:
        printf("DIR ");
        break;
case FTW_DNR:
        printf("CANT READ DIR ");
        break;
case FTW_NS:
        printf("STAT ERROR ");
        break;
default:
        printf("unknow error ");
```

```
        }
        if((S_OP ==1))
                printf(" (%ld)",statptr->st_size);
        printf("\n");



#endif
  return 0;
}
```