```c
/*
    Name: Rohit Kumar
    Student Id: 110088741
    Assignment: 1
    Course: COMP 8567
    Section: 3
*/

#define _XOPEN_SOURCE 500

//included all header files
#include <ftw.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>

/*
    declare variables
        source: char[] with size 100
        destination : char[] with size 100
        extension list : two dimenstional array
*/
char source[100], destination[100], ext_list[5][5];


/*
    initializing the variables
*/
int size = 0, folder_num = 0;


/*
    below function will validate which file extenstion needs to be ignored
provided by user
*/
int validate_extension(const char *ext_path)
{
    int i;
    char *path = strrchr(ext_path, '.');
    if (path == NULL)
        return 0;

    for (i = 0; i < size; i++)
    {
        //compare the file extension
        if (strcmp(path + 1, ext_list[i]) == 0)
            return 1;
    }
    return 0;
}
```

```c
/*
    below function will copy the files from source to destination
    also create destination if does not exist
*/
int copy_source_dest(const char *path, const struct stat *sb, int
file_type, struct FTW *ftwbuf)
{
    char destination_path[100];
    int create;

    //this function formats and constructs a new string by combining
multiple strings
    sprintf(destination_path, "%s%s", destination, path + strlen(source));

    //check if fpath is a directory
    if (file_type == FTW_D)
    {
        //creating directory from source to destination
        if (folder_num == 0)
        {
            folder_num++;
        }
        else
        {
            // creating destination directory if doesn't exist
            create = mkdir(destination_path, 0777);
            if (create == -1 && errno != EEXIST)
                printf("\nError occurred in mkdir");
        }
    }

    //check if fpath is a regular file
    if (file_type == FTW_F)
    {
        //creating file from source to destination
        if (size == 0)
        {
            //copy all files and creating hard link for both file path
            create = link(path, destination_path);
            if (create == -1)
                printf("\nError occurred while creating hard link");
        }
        else
        {
            // ignore files if extension matches
            if (!validate_extension(path))
            {
                //copy all files and creating hard link for both file path
                create = link(path, destination_path);
                if (create == -1)
                    printf("\nError occurred while creating hard link");
            }
        }
    }
    return 0;
}
```

```
//method to copy directory and files using nftw system call
int copy_folders(const char *source, const char *destination)
{
    return nftw(source, copy_source_dest, 10, FTW_PHYS);
}

//method to move directory and files using nftw system call
int move_folders(const char *source, const char *destination)
{
    int flag;

    // copy all folders
    flag = copy_folders(source, destination);
    if (flag == -1)
        return flag;

    // remove all source folders using system call
    flag = nftw(source, remove, 10, FTW_DEPTH | FTW_PHYS);
    if (flag == -1)
        printf("\nfile and directories removed");

    return flag;
}

/*
    this is main method and program execution will starts from here
*/
int main(int argc, char *argv[])
{

    /*
        below logic will check if all the parameters provided by user or
not
    */
    if (argc < 4)
    {
        printf("-----------I N V O K E---F U N C T I O N---A S---B E L O W-
---------\n");
        printf("./ncpmvdir [source_dir] [destination_dir] [-cp|-mv]
[extension list]\n");
        printf("-------------------------------------------------------------
---------\n");
        exit(0);
    }


    /*
        below logic will copy source and destination path provided by user
    */
    strcpy(source, argv[1]);
    strcpy(destination, argv[2]);


    /*
        below logic will check if source location is correct or not
        in case it is not aavailable then will print error message
    */
```

```
    struct stat status;
    if (!(stat(source, &status) == 0 && S_ISDIR(status.st_mode)))
    {
        printf("!!Error: source file is not available!!\n", argv[0]);
        exit(0);
    }


    /*
        below logic will create destination folder provided by user
        in case it is not available
    */
    const char * dest_path = destination;
    if (!(stat(dest_path, &status) == 0 && S_ISDIR(status.st_mode)))
    {
        // mkdir system call for making directory with read write
permissions
        mkdir(dest_path, 0777);
    }


    /*
        below logic will copy all the extenstion provided by user
        Note: maxmimum extenstion allowed is six,
              in case user will enter more than six extension program will
ignore after six
    */
    if (argc > 4)
    {
        //only 6 extensions are allowed
        for (int i = 4; i < argc && i - 4 < 6; i++)
        {
            strcpy(ext_list[i - 4], argv[i]);
            size++;
        }
    }


    /*
        perform operation: copy or move
        below logic will either copy or move from source to destination
        based on the operation passed by user
    */
    if (strcmp(argv[3], "-cp") == 0)
        copy_folders(source, destination);
    else if (strcmp(argv[3], "-mv") == 0)
        move_folders(source, destination);
    else
        printf("!!!Error: Invalid operation!!!\n", argv[0]);
}
```