Complete below code of Customised Dynamic File System

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<iostream>

#define MAXINODE 100

#define READ 1
#define WRITE 2

#define MAXFILESIZE 2048

#define REGULAR 1
#define SPECIAL 2

#define START 0
#define CURRENT 1
#define END 2

typedef struct superblock
{
        int TotalInodes;
        int FreeInode;
}SUPERBLOCK, *PSUPERBLOCK;

typedef struct inode
{
        char FileName[50];
        int InodeNumber;
        int FileSize;
        int FileActualSize;
        int FileType;
        char *Buffer;
        int LinkCount;
        int ReferenceCount;
        int permission;
        struct inode *next;
}INODE,*PINODE,**PPINODE;

typedef struct filetable
{
        int readoffset;
        int writeoffset;
        int count;
        int mode;
        PINODE ptrinode;
}FILETABLE,*PFILETABLE;

typedef struct ufdt
{
        PFILETABLE ptrfiletable;
}UFDT;

UFDT UFDTArr[50];
SUPERBLOCK SUPERBLOCKobj;
PINODE head = NULL;
```

```
//////////////////////////////////////////////

void man(char *name)
{
        if(name == NULL) return;
        if(strcmp(name,"create") == 0)
        {
                printf("Description : Used to create new regular file\n");
                printf("Usage : create File_name Permission\n");
        }
        else if(strcmp(name,"read") == 0)
        {
                        // Add description
        }
        else if(strcmp(name,"write") == 0)
        {
                        // Add description
        }
        else if(strcmp(name,"ls") == 0)
        {
                        // Add description
        }
        else if(strcmp(name,"stat") == 0)
        {
                        // Add description
        }
        else if(strcmp(name,"fstat") == 0)
        {
                        // Add description
        }
        else if(strcmp(name,"truncate") == 0)
        {
                        // Add description
        }
        else if(strcmp(name,"open") == 0)
        {
                        // Add description
        }
        else if(strcmp(name,"close") == 0)
        {
                        // Add description
        }
        else if(strcmp(name,"closeall") == 0)
        {
                        // Add description
        }
        else if(strcmp(name,"lseek") == 0)
        {
                        // Add description
        }
        else if(strcmp(name,"rm") == 0)
        {
                        // Add description
        }
        else
        {
                printf("ERROR : No manual entry available.\n");
        }
}
```

```
void DisplayHelp()
{
	printf("ls : To List out all files\n");
	printf("clear : To clear console\n");
	printf("open : _____\n”);
	printf("close : _____\n");
	printf("closeall : _____\n");
	printf("read : _____\n");
	printf("write :_____\n");
	printf("exit : _____");
	printf("stat : _____\n");
	printf("fstat :_____\n");
	printf("truncate : _____\n");
	printf("rm : _____\n");
}

int GetFDFromName(char *name)
{
	int i = 0;

	while(i<50)
	{
		if(UFDTArr[i].ptrfiletable != NULL)
			if(strcmp((UFDTArr[i].ptrfiletable->ptrinode->FileName),name)==0)
				break;
		i++;
	}

	if(i == 50)		return -1;
	else			return i;
}

PINODE Get_Inode(char * name)
{
	PINODE temp = head;
	int i = 0;

	if(name == NULL)
		return NULL;

	while(temp!= NULL)
	{
		if(strcmp(name,temp->FileName) == 0)
			break;
		temp = temp->next;
	}
	return temp;
}

void CreateDILB()
{
	// Create singly linked list of MAXINODE (50) inodes.
	// And store address of first node into head pointer which is global.
	// LinkCount, ReferenceCount , FileType, FileSize members should be set to 0
	// Buffer member should be set to NULL;
	// In InodeNumber member store the value from 1 to MAXINODE(50) which indicates node number
}

void InitialiseSuperBlock()
{
	// Write one loop which iterates 50 times and initialise UFDTArr[i].ptrfiletable to NULL.
	// where i starts from 0 to 49.
	// TotalInodes & FreeInode members of SUPERBLOCKobj should be set to 50.
```

```
}

int CreateFile(char *name,int permission)
{
        // Check input parameters

        // Decrement free node count from superblock

        // Find out empty node from Inode linked list

        // Find out empty entry from UFDTArr

        // Allocate memory for FileTable

        // Initialise file table entries

        // Allocate members of Inode

        // Allocate memory for storing data of file (1024) bytes and store its address in Buffer pointer

        // Return the index of UFDTArr in which address of FileTable is stored.
}

void ls_file()
{
        // Travel linked list of inodes and display all its members.
        // Use head pointer
}

int fstat_file(int fd)
{
        // Display all information of file from file descriptor.
        // From FD access FileTable
        // From File table access inode
        // Display information from the inode
}

int stat_file(char *name)
{
        // Display all information of file from file name.
        // From file name search specific inode from linked list
}

int main()
{
        char *ptr = NULL;
        int ret = 0, fd = 0, count = 0;
        char command[4][80], str[80], arr[1024];

        InitialiseSuperBlock();
        CreateDILB();

        while(1)
        {
                fflush(stdin);
                strcpy(str,"");

                printf("\nMarvellous VFS : > ");
                fgets(str,80,stdin);

                count = sscanf(str,"%s %s %s %s",command[0],command[1],command[2],command[3]);

                if(count == 1)
                {
```

```
            if(strcmp(command[0],"ls") == 0)
            {
                    ls_file();
            }
            else if(strcmp(command[0],"closeall") == 0)
            {

            }
            else if(strcmp(command[0],"clear") == 0)
            {
                    system("cls");
                    continue;
            }
            else if(strcmp(command[0],"help") == 0)
            {
                    DisplayHelp();
                    continue;
            }
            else if(strcmp(command[0],"exit") == 0)
            {
                    printf("Terminating the Marvellous Virtual File System\n");
                    break;
            }
            else
            {
                    printf("\nERROR : Command not found !!!\n");
                    continue;
            }
    }
    else if(count == 2)
    {
            if(strcmp(command[0],"stat") == 0)
            {
                    ret = stat_file(command[1]);
                    if(ret == -1)
                            printf("ERROR : Incorrect parameters\n");
                    if(ret == -2)
                            printf("ERROR : There is no such file\n");
                    continue;
            }
            else if(strcmp(command[0],"fstat") == 0)
            {
                    ret = fstat_file(atoi(command[1]));
                    if(ret == -1)
                            printf("ERROR : Incorrect parameters\n");
                    if(ret == -2)
                            printf("ERROR : There is no such file\n");
                    continue;
            }
            else if(strcmp(command[0],"close") == 0)
            {

            }
            else if(strcmp(command[0],"rm") == 0)
            {

            }
            else if(strcmp(command[0],"man") == 0)
            {
                    man(command[1]);
            }
            else if(strcmp(command[0],"write") == 0)
            {
```

```
                }
                else if(strcmp(command[0],"truncate") == 0)
                {

                }
                else
                {
                        printf("\nERROR : Command not found !!!\n");
                                continue;
                }
        }
        else if(count == 3)
        {
                if(strcmp(command[0],"create") == 0)
                {
                        ret = CreateFile(command[1],atoi(command[2]));
                        if(ret >= 0)
                                printf("File is successfully created with file descriptor : %d\n",ret);
                        if(ret == -1)
                                printf("ERROR : Incorrect parameters\n");
                        if(ret == -2)
                                printf("ERROR : There is no inodes\n");
                        if(ret == -3)
                                printf("ERROR : File already exists\n");
                        if(ret == -4)
                                printf("ERROR : Memory allocation failure\n");
                        continue;
                }
                else if(strcmp(command[0],"open") == 0)
                {

                }
                else if(strcmp(command[0],"read") == 0)
                {

                }
                else
                {
                        printf("\nERROR : Command not found !!!\n");
                        continue;
                }
        }
        else if(count == 4)
        {
                if(strcmp(command[0],"lseek") == 0)
                {

                }
                else
                {
                        printf("\nERROR : Command not found !!!\n");
                        continue;
                }
        }
        else
        {
                printf("\nERROR : Command not found !!!\n");
                continue;
        }
    }
    return 0;
}
```