Assignment: Write a C program as per the following specification

Allocate memory of 1MB dynamically with 4K allignment i.e., starting address of this block is multiple of 4K and map that memory as follows.

Super block	Datablock Bitmap	Inodeblock Bitmap	Inode Table	Data blocks
1 block	1 block	1 block	22 blocks	n blocks

|<---------------------------------------Block Group-->|

1Block = 1KB ----> 1MB will have 1024 data blocks.

Super block :

-->Map the following structure to the block named as super block .This structure should be alligned to 1024 byte boundary.
The super block structure should be

```
struct  super_block {
        u_int   total_inodes;
        u_int   total_datablocks;
        u_int   free_inodes;
        u_int   free_datablocks;
};
```

Data block bit map :

-->1024 bits are valid ->representing 1024 blocks of the block group and the rest of the bits are invalid.
-->A bit is set when the block is allocated to a particular inode.
-->A bit is cleared when the block is deallocated or removed from the inode.

Inode bit map :

The inode structure should be

```
struct inode {
        u_int   i_no : 10;       // inode number
        u_int   size : 12;       // size of file
        u_int   blk_used : 3;    // no of data blocks allocated
        u_short blk[5];          //allocated data blocks indexes
        u_char fname[8];         //file name limited to 8 characters
};
```

-->1024 bits are valid representing an inode and the rest of the bits are invalid.
-->A bit position is an index into the inode table, which contains 1024 inode structures.
-->A bit is set when the corresponding inode is allocated.

-->A bit is cleared, when the corresponding inode is deallocated.

<u>Inode table :</u>

-->Table of 1024 inode structures.
-->Each inode structure has a corresponding bit representation in the inode bit map i.e., there is a one-to-one mapping between bit position in the inode bitmap and index into the inode table.

<u>Data Blocks:</u>

-->Free data blocks are allocated when needed .
-->First 25 blocks of the block group are occupied by super block, inode block bitmap, data block bitmap, inode table and the remaining 999 blocks are available for allocation for files.
-->There is one-to-one mapping between bit position in the data block bitmap and the data block.

The program is a menu driven execution with following menu:

<u>Main Menu:</u>

1.Create a file.
    1.1 Enter file name.
2.Remove a file.
    2.1 Enter file name.
3.Write into a file.
    3.1 Enter file name.
    3.2 Enter text.
4.cat
    4.1 Enter file name.
5.ls

1.Create a file:
->Get a free bit (cleared bit) from inode bitmap.
->Set that bit in the inode bitmap.
->Get the bit position.
->Index this bit position in the inode table to get the corresponding inode structure.
->Populate the name and inode number in the corresponding inode structure.
->Update the super block structure appropriately.

2.Remove a file:
->Get the inode corresponding to the entered file name from the inode table.
->Clear the data blocks, if allocated, from the inode structure and appropriately update in the data block bit map.
->Clear the bit in the inode bitmap correspondingly to this inode.
->Update the super block structure appropriately.

3.Write in to a file:
->Get the inode corresponding to the entered file from the inode table.
->Read data from stdin.
->Allocate data blocks appropriately to fill the data.

        ->Get a free bit (cleared bit) from the data block bitmap.
        ->Set that bit in the data block bitmap.
        ->Get the bit position.
        ->Index this bit position in the block group to get the data block.
        ->Update the appropriate inode structure with the data block number.
    ->Copy the data got from stdin to the data block.

4.Reading from the file (cat command):

->Get the inode of the entered file from the inode table.
->Get the data block number from the above inode structure
->Read the content of that data block and dump it on the terminal.

5.ls:

->Check every valid bit of inode bitmap.
->If the bit is set, then display the corresponding file name present in the inode structure by indexing the bit position into the inode table.
->For every valid bit set in the inode bitmap, get the corresponding inode structure from the inode table and print the file name present in the inode structure.