```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define FREEBLOCK -2
#define ALLOCATE 1
#define NEWNODE (struct info*)malloc(sizeof(struct info))

struct info
{
    char fname[20];
    int start;
    int length;
    int blocks[50];
    struct info *next;
};

struct info *E = NULL;
struct info *last = NULL;

int *D = NULL;
int dsize = 0;
int used = 0;
int freeSpace = 0;

int main(void)
{
    // local variable declarations
    int choice = 0;
    // local function diclarations
    void initDisk(void);
    void allocate(void);
    void deallocate(void);
    void displayDirFile(void);

    // code

    printf("\n\nEnter Disk Size : ");
    scanf("%d",&dsize);

    initDisk();

    while(1)
    {
        printf("\n\n         MENU\n");
        printf("1:Allocate Space for File\n2:Deallocate Space for Deleted file
          \n3:show used and free space on disk\n4:Exit");
        printf("\nEnter Your Choice : ");
        scanf("%d",&choice);

        switch(choice)
        {
```

```c
            case 1:
                allocate();
                break;
            case 2:
                deallocate();
                break;
            case 3:
                displayDirFile();
                break;
            case 4:
                exit(0);
                break;
        }

    }
    return(0);
}

void initDisk(void)
{
    int i = 0;

    // creating disk
    D = (int *)malloc(sizeof(int) * dsize);

    for(i = 0; i < dsize; i++)
    {
        D[i] = FREEBLOCK;
    }

    used = 0;
    freeSpace = dsize;
}

void allocate(void)
{
    // local variable declarations
    char fname[20] = {'\0'};
    int length = 0;
    int blknum = 0;
    int i = 0;
    int blks[50] = { 0 };
    int cnt = 0;
    int start = 0;

    // local function declarations
    int search(void);
    void makeDirEntry(char[], int, int, int[]);

    // code

    printf("\n\nEnter File Name : ");
```

```c
    scanf("%s",fname);

    printf("Enter Length of File : ");
    scanf("%d",&length);

    if (length + 1 > freeSpace)
    {
        printf("\nError: No Disk Space Available\n");
        return;
    }

    blknum = search(); // get first block for new file
    start = blknum;
    D[blknum] = ALLOCATE; // allocate the first block

    for (cnt = 1,i = 0; cnt <= length; cnt++,i++)
    {
        blknum = search(); //  search new block
        blks[i] = blknum; // copy block number into index block
        D[blknum] = ALLOCATE; // allocate the block

    }

    makeDirEntry(fname,start,length,blks);

    used = used + (length + 1);
    freeSpace = freeSpace - (length + 1);
    printf("\n\nFile Allocated Successfully !!!\n");
    printf("\n\nDisk Status : ");
    for(i = 0; i < dsize;i++)
    {
        printf("%d ",D[i]);
    }
}

int search(void)
{
    // local variable declarations
    int i = 0;

    for (i = 0; i < dsize; i++)
    {
        if (D[i] == FREEBLOCK)
        {
            return(i);
        }
    }

    return(-1);
}

void makeDirEntry(char fname[],int start,int length,int blks[])
```

```
156 {
157     // local variable declarations
158     struct info *t = NULL;
159     int cnt = 0;
160     int i = 0;
161 
162     t = NEWNODE;
163 
164     if(t == NULL)
165     {
166         printf("malloc() failed !!!\n\n");
167         exit(0);
168     }
169 
170     strcpy(t->fname,fname);
171     t->start = start;
172     t->length = length;
173     t->next = NULL;
174 
175     for (cnt = 1,i = 0; cnt <= length; cnt++,i++)
176     {
177         t->blocks[i] = blks[i];
178     }
179 
180     if(E == NULL)
181     {
182         E = t;
183         last = t;
184     }
185     else
186     {
187         last->next = t;
188         last = last->next;
189     }
190 }
191 
192 void deallocate(void)
193 {
194     // local variable declarations
195     struct info *s = NULL;
196     char fname[20] = {'\0'};
197     int length = 0;
198     int i = 0;
199     int blknum = 0;
200     int cnt = 0;
201 
202     // local function declarations
203     void deleteEntry(struct info*);
204 
205     // code
206     printf("\nEnter File Name to delete : ");
207     scanf("%s",fname);
```

```c

    for(s = E; s != NULL; s = s->next)
    {
        if(strcmp(fname,s->fname) == 0)
        {
            blknum = s->start;
            length = s->length;

            D[blknum] = FREEBLOCK;

            for (cnt = 1,i = 0; cnt <= length; cnt++,i++)
            {
                blknum = s->blocks[i]; // get the next block for freeing
                D[blknum] = FREEBLOCK; // free the block
            }

            deleteEntry(s);

            printf("Disk Status : ");

            for (i = 0; i < dsize; i++)
            {
                printf("%d ", D[i]);
            }

            return;
        }
    }

    printf("\nInvalid File Name given to delete \n\n");
}

void deleteEntry(struct info *s)
{
    struct info *f = NULL;
    int length = s->length;

    if (s == E)
    {
        E = E->next;
        free(s);
        used = used - (length + 1);
        freeSpace = freeSpace + (length + 1);
        return;
    }

    f = E;

    while (f->next != s)
    {
        f = f->next;
    }
```

```
260
261     if (f->next == last) // you are going to delete last node
262     {
263         last = f;
264     }
265
266     f->next = s->next;
267     free(s);
268
269     used = used - (length + 1);
270     freeSpace = freeSpace + (length + 1);
271 }
272
273 void displayDirFile(void)
274 {
275     struct info *s = NULL;
276
277     printf("\nFName\tStart\tLength\n");
278
279     for(s = E; s != NULL;s = s->next)
280     {
281         printf("%s\t%d\t%d\n",s->fname,s->start,s->length);
282     }
283
284     printf("\n Used Block : %d\n",used);
285     printf("Free Bloack : %d",freeSpace);
286 }
287
288
```