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

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

struct info
{
    char fname[20];
    int start;
    int end;
    int length;
    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)
```

```
52          {
53              case 1:
54                  allocate();
55                  break;
56              case 2:
57                  deallocate();
58                  break;
59              case 3:
60                  displayDirFile();
61                  break;
62              case 4:
63                  exit(0);
64                  break;
65          }
66
67      }
68      return(0);
69  }
70
71  void initDisk(void)
72  {
73      int i = 0;
74
75      // creating disk
76      D = (int *)malloc(sizeof(int) * dsize);
77
78      for(i = 0; i < dsize; i++)
79      {
80          D[i] = FREEBLOCK;
81      }
82
83      used = 0;
84      freeSpace = dsize;
85  }
86
87  void allocate(void)
88  {
89      // local variable declarations
90      char fname[20] = { '\0' };
91      int length = 0;
92      int blknum = 0;
93      int start = 0;
94      int end = 0;
95      int cnt = 0;
96      int tbn = 0;
97      int i = 0;
98
99      // local function declarations
100     int search(void);
101     void makeDirEntry(char[], int, int, int);
102     // code
103     printf("\n\nEnter The File Name : ");
```

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

    printf("\n\nEnter Length of a file : ");
    scanf("%d", &length);

    if (length > freeSpace)
    {
        printf("\n\nError : No Space Avaliable to disk !!!\n");
        return;
    }

    blknum = search();
    start = blknum;
    D[blknum] = ENDBLOCK;

    for (cnt = 2; cnt <= length; cnt++)
    {
        tbn = search(); // getting new block
        D[blknum] = tbn; // writing next block number into current block number
        blknum = tbn; // moving blknum to next block
        D[blknum] = ENDBLOCK; // Making next block as endblock
    }

    end = blknum;

    // making entry to directory
    makeDirEntry(fname, start, end, length);

    used = used + length;
    freeSpace = freeSpace - length;

    printf("\n\nDISK STATUS : ");

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

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

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

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

```
208     // code
209     printf("\nEnter File Name to delete : ");
210     scanf("%s", fname);
211
212     for (s = E; s != NULL; s = s->next)
213     {
214         if (strcmp(fname, s->fname) == 0)
215         {
216             blknum = s->start;
217             length = s->length;
218
219             // freeing block
220
221             for (cnt = 1; cnt <= length; cnt++)
222             {
223                 tbn = D[blknum]; // take next block number
224                 D[blknum] = FREEBLOCK; // free the blknum block
225                 blknum = tbn; // shift to next block
226             }
227
228             deleteEntry(s);
229
230             printf("Disk Status : ");
231
232             for (i = 0; i < dsize; i++)
233             {
234                 printf("%d ", D[i]);
235             }
236
237             return;
238         }
239     }
240
241     printf("\nInvalid File Name given to delete \n\n");
242 }
243
244
245 void deleteEntry(struct info *s)
246 {
247     // local variable declarations
248     struct info *f = NULL;
249
250     // code
251     int length = s->length;
252
253     if (s == E)
254     {
255         E = E->next;
256         free(s);
257         used = used - length;
258         freeSpace = freeSpace + length;
259         return;
```

```
260     }
261
262     f = E;
263
264     while (f->next != s)
265     {
266         f = f->next;
267     }
268
269     if (f->next == last) // you are going to delete last node
270     {
271         last = f;
272     }
273
274     f->next = s->next;
275     free(s);
276
277     used = used - length;
278     freeSpace = freeSpace + length;
279 }
280
281 void displayDirFile(void)
282 {
283     // local variable declarations
284     struct info *s = NULL;
285
286     // code
287     printf("\nFName\tStart\tEnd\tLength\n");
288
289     for(s = E; s != NULL;s = s->next)
290     {
291         printf("%s\t%d\t%d\t%d\n",s->fname,s->start,s->end,s->length);
292     }
293
294     printf("\n Used Block : %d\n",used);
295     printf("Free Bloack : %d",freeSpace);
296 }
297
298
```