```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;
    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)
        {
            case 1:
```

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

```c
    scanf("%d",&length);

    // serach free block if found return block number else -1
    blknum = search(length);

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

    printf("\nBlock Allocated\n");

    assignBlock(blknum,length);

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

    makeDirEntry(fname,blknum,length);

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

int search(int length)
{
    // local variable declarations
    int i = 0;
    int cnt = 0;
    int flag = 0;
    int blknum = 0;

    for (i = 0; i < dsize; i++)
    {
        if (D[i] == FREEBLOCK)
        {
            flag = 1;
            blknum = i;
            for (cnt = 1; cnt <= length; cnt++)
            {
                if (D[i] == FREEBLOCK)
                {
                    i++;
                }
                else
                {
                    flag = 0;
                    break;
                }
```

```
156             }
157             if (flag == 1)
158                 return blknum;
159         }
160     }
161     return -1;
162 }
163
164 void assignBlock(int blknum,int length)
165 {
166     int cnt = 0;
167
168     for(cnt = 1; cnt <= length; cnt++)
169     {
170         D[blknum] = ALLOCATE;
171         blknum++;
172     }
173 }
174
175 void makeDirEntry(char fname[],int start,int length)
176 {
177     struct info *t = NULL;
178
179     t = NEWNODE;
180
181     if(t == NULL)
182     {
183         printf("malloc() failed !!!\n\n");
184         exit(0);
185     }
186
187     strcpy(t->fname,fname);
188     t->start = start;
189     t->length = length;
190     t->next = NULL;
191
192     if(E == NULL)
193     {
194         E = t;
195         last = t;
196     }
197     else
198     {
199         last->next = t;
200         last = last->next;
201     }
202 }
203
204 void deallocate(void)
205 {
206     // local variable declarations
207     struct info *s = NULL;
```

```
208     char fname[20] = {'\0'};
209     int length = 0;
210     int i = 0;
211     int blknum = 0;
212 
213     // local function declarations
214     void freeBlock(int,int);
215     void deleteEntry(struct info*);
216 
217     // code
218     printf("\nEnter File Name to delete : ");
219     scanf("%s",fname);
220 
221     for(s = E; s != NULL; s = s->next)
222     {
223         if(strcmp(fname,s->fname) == 0)
224         {
225             blknum = s->start;
226             length = s->length;
227 
228             freeBlock(blknum,length);
229 
230             printf("Disk Status : ");
231 
232             for(i = 0; i < dsize;i++)
233             {
234                 printf("%d ",D[i]);
235             }
236 
237             deleteEntry(s);
238 
239             return;
240         }
241     }
242 
243     printf("\nInvalid File Name given to delete \n\n");
244 }
245 
246 void freeBlock(int blknum,int length)
247 {
248     int cnt = 0;
249 
250     for(cnt = 1; cnt <= length;cnt++)
251     {
252         D[blknum] = FREEBLOCK;
253         blknum++;
254     }
255 }
256 
257 void deleteEntry(struct info *s)
258 {
259     struct info *f = NULL;
```

```
260     int length = s->length;
261
262     if (s == E)
263     {
264         E = E->next;
265         free(s);
266         used = used - length;
267         freeSpace = freeSpace + length;
268         return;
269     }
270
271     f = E;
272
273     while (f->next != s)
274     {
275         f = f->next;
276     }
277
278     if (f->next == last) // you are going to delete last node
279     {
280         last = f;
281     }
282
283     f->next = s->next;
284     free(s);
285
286     used = used - length;
287     freeSpace = freeSpace + length;
288 }
289
290 void displayDirFile(void)
291 {
292     struct info *s = NULL;
293
294     printf("\nFName\tStart\tLength\n");
295
296     for(s = E; s != NULL;s = s->next)
297     {
298         printf("%s\t%d\t%d\n",s->fname,s->start,s->length);
299     }
300
301     printf("\n Used Block : %d\n",used);
302     printf("Free Bloack : %d",freeSpace);
303 }
304
305
```