

LAB09:

Write a C program to simulate the following contiguous memory allocation techniques

a) Worst-fit

b) Best-fit

c) First-fit

INPUT:

```

1  #include <stdio.h>
2  #include <limits.h>
3
4  void firstFit(int blockSize[], int m, int fileSize[], int n) {
5      int allocation[n];
6
7      for (int i = 0; i < n; i++) {
8          allocation[i] = -1;
9      }
10
11     for (int i = 0; i < n; i++) {
12         for (int j = 0; j < m; j++) {
13             if (blockSize[j] >= fileSize[i]) {
14                 allocation[i] = j;
15                 blockSize[j] -= fileSize[i];
16                 break;
17             }
18         }
19     }
20
21     printf("Memory Management Scheme - First Fit\n");
22     printf("File_no\tFile_size\tBlock_no\tBlock_size\tFragment\n");
23     for (int i = 0; i < n; i++) {
24         printf("%d\t%d\t\t", i+1, fileSize[i]);
25         if (allocation[i] != -1) {
26             printf("%d\t\t\t\t\t", allocation[i]+1, blockSize[allocation[i]], blockSize[allocation[i]]);
27         } else {
28             printf("Not Allocated\n");
29         }
30     }
31     printf("\n");
32 }
33
34 void bestFit(int blockSize[], int m, int fileSize[], int n) {
35     int allocation[n];
36
37     for (int i = 0; i < n; i++) {
38         allocation[i] = -1;
39     }
40
41     for (int i = 0; i < n; i++) {
42         int bestIdx = -1;
43         for (int j = 0; j < m; j++) {
44             if (blockSize[j] >= fileSize[i]) {
45                 if (bestIdx == -1 || blockSize[bestIdx] > blockSize[j]) {
46                     bestIdx = j;
47                 }
48             }
49         }
50     }

```

```

50
51     if (bestIdx != -1) {
52         allocation[i] = bestIdx;
53         blockSize[bestIdx] -= fileSize[i];
54     }
55 }
56
57 printf("Memory Management Scheme - Best Fit\n");
58 printf("File_no\tFile_size\tBlock_no\tBlock_size\tFragment\n");
59 for (int i = 0; i < n; i++) {
60     printf("%d\t%d\t\t", i+1, fileSize[i]);
61     if (allocation[i] != -1) {
62         printf("%d\t\t%d\t\t", allocation[i]+1, blockSize[allocation[i]]);
63     } else {
64         printf("Not Allocated\n");
65     }
66 }
67 printf("\n");
68 }
69
70 void worstFit(int blockSize[], int m, int fileSize[], int n) {
71     int allocation[n];
72
73     for (int i = 0; i < n; i++) {
74         allocation[i] = -1;
75     }
76
77     for (int i = 0; i < n; i++) {
78         int worstIdx = -1;
79         for (int j = 0; j < m; j++) {
80             if (blockSize[j] >= fileSize[i]) {
81                 if (worstIdx == -1 || blockSize[worstIdx] < blockSize[j]) {
82                     worstIdx = j;
83                 }
84             }
85         }
86
87         if (worstIdx != -1) {
88             allocation[i] = worstIdx;
89             blockSize[worstIdx] -= fileSize[i];
90         }
91     }
92
93     printf("Memory Management Scheme - Worst Fit\n");
94     printf("File_no\tFile_size\tBlock_no\tBlock_size\tFragment\n");
95     for (int i = 0; i < n; i++) {
96         printf("%d\t\t\t", i+1, fileSize[i]);
97         if (allocation[i] != -1) {
98             printf("%d\t\t\t", allocation[i]+1, blockSize[allocation[i]]);

```

```

98         printf("%d\t\t%d\t\t%d\n", allocation[i]+1, blockSize[allocation[i]], blockSize[allocation[i]]);
99     } else {
100         printf("Not Allocated\n");
101     }
102 }
103 printf("\n");
104 }
105
106 int main() {
107     int m, n;
108
109     printf("Enter the number of blocks: ");
110     scanf("%d", &m);
111     int blockSize[m];
112     printf("Enter the size of the blocks:\n");
113     for (int i = 0; i < m; i++) {
114         printf("Block %d: ", i+1);
115         scanf("%d", &blockSize[i]);
116     }
117
118     printf("Enter the number of files: ");
119     scanf("%d", &n);
120     int fileSize[n];
121     printf("Enter the size of the files:\n");
122     for (int i = 0; i < n; i++) {
123         printf("File %d: ", i+1);
124         scanf("%d", &fileSize[i]);
125     }
126
127     int blockSizeCopy[m];
128     for (int i = 0; i < m; i++) {
129         blockSizeCopy[i] = blockSize[i];
130     }
131
132     firstFit(blockSizeCopy, m, fileSize, n);
133
134     for (int i = 0; i < m; i++) {
135         blockSizeCopy[i] = blockSize[i];
136     }
137
138     bestFit(blockSizeCopy, m, fileSize, n);
139
140     for (int i = 0; i < m; i++) {
141         blockSizeCopy[i] = blockSize[i];
142     }
143
144     worstFit(blockSizeCopy, m, fileSize, n);
145     return 0;
146 }

```

OUTPUT:

Enter the number of blocks: 5

Enter the size of the blocks:

Block 1: 400

Block 2: 700

Block 3: 200

Block 4: 300

Block 5: 600

Enter the number of files: 4

Enter the size of the files:

File 1: 212

File 2: 517

File 3: 312

File 4: 526

Memory Management Scheme - First Fit

File_no	File_size	Block_no	Fragment
1	212	1	188
2	517	2	183
3	312	5	288
4	526	Not Allocated	

Memory Management Scheme - Best Fit

File_no	File_size	Block_no	Fragment
1	212	4	88
2	517	5	83
3	312	1	88
4	526	2	174

Memory Management Scheme - Worst Fit

File_no	File_size	Block_no	Fragment
1	212	2	176
2	517	5	83
3	312	2	176
4	526	Not Allocated	