

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Process
5 {
6     int id;
7     int size;
8     int allocation;
9     bool isGiven = false;
10 };
11
12 struct Memory
13 {
14     int size;
15     int free;
16     int allocated;
17     bool isTaken = false;
18     int extfrag;
19     int givenProcessId = -1;
20 };
21
22 int m;
23 int n;
24 int external_fragmentation = 0;
25 int internal_fragmentation = 0;
26
27 void firstFit(Process p[], int n, Memory mem[], int m)
28 {
29     // pick each process and find suitable blocks
30     // according to its size and assign to it
31     for (int i = 0; i < n; i++)
32     {
33         for (int j = 0; j < m; j++)
34         {
35             if (mem[j].size >= p[i].size)
36             {
37                 mem[j].isTaken = true;
38                 p[i].isGiven = true;
39                 mem[j].givenProcessId = p[i].id;
40                 mem[j].free -= p[i].size;
41                 mem[j].size -= p[i].size;
42                 p[i].allocation = j + 1;
43                 mem[j].allocated = p[i].id;
44                 break;
45             }
46         }
47     }
48 }
49
50
51 void calcfrag(Process p[], int n, Memory mem[], int m)
52 {
53     int flag = 0;
54     for (int i = 0; i < n; i++)
55     {
56         if (p[i].isGiven != true)
57         {
58             flag = 1;
59             break;
60         }
61     }
```

```

62     if (flag == 0)
63     {
64         external_fragmentation = 0;
65     }
66     else
67     {
68         for (int i = 0; i < m; i++)
69         {
70             if (mem[i].isTaken != true)
71             {
72                 external_fragmentation += mem[i].size;
73             }
74         }
75     }
76
77     for (int i = 0; i < m; i++)
78     {
79         if (mem[i].isTaken != false)
80         {
81             internal_fragmentation += mem[i].free;
82         }
83     }
84 }
85
86 void printTable(Process P[], int n, Memory mem[], int m, int memorySize[])
87 {
88
89     for(int i=0;i<m;i++)
90     {
91         if(mem[i].free==memorySize[i])
92         {
93             mem[i].free=0;
94         }
95     }
96
97     cout << "\nTable-->(-1 Denotes Unallocated process)\n";
98     int i;
99
100    puts("+-----+-----+-----+-----+");
101    puts("| BNO | Block Size | Process All. | Internal Fragg. |");
102    puts("+-----+-----+-----+-----+");
103
104    for (i = 0; i < m; i++)
105    {
106        printf("| %2d | %2d | %2d | %3d |\n", i,
memorySize[i], mem[i].givenProcessId, mem[i].free);
107        puts("+-----+-----+-----+-----+");
108    }
109
110    cout << "External Fragmentation: " << external_fragmentation << endl;
111    cout << "Internal Fragmentation: " << internal_fragmentation << endl;
112 }
113
114 int main()
115 {
116     cout << "\nEnter the number of memory blocks: ";
117     cin >> m;
118     Memory mem[m];
119     int memorySize[m];
120     for (int i = 0; i < m; i++)
121     {

```

```
122     cout << "\n";
123     cout << "Enter the size of the memory block " << i + 1 << ": ";
124     cin >> mem[i].size;
125     mem[i].free = mem[i].size;
126     mem[i].allocated = -1;
127     mem[i].extfrag = 0;
128     memorySize[i] = mem[i].size;
129 }
130
131 cout << "\nEnter the number of processes: ";
132 cin >> n;
133 Process p[n];
134 for (int i = 0; i < n; i++)
135 {
136     p[i].id = i + 1;
137     cout << "\n";
138     cout << "\nEnter the size of the process" << p[i].id << ": ";
139     cin >> p[i].size;
140 }
141
142 firstFit(p, n, mem, m);
143
144 calcfrag(p, n, mem, m);
145
146 printTable(p, n, mem, m, memorySize);
147 }
```