```
1 #include <bits/stdc++.h>
 2 using namespace std;
 3
 4 struct Process
 5 |{
       int id;
 6
 7
       int size;
       int allocation;
 8
 9
       bool isGiven = false;
10 };
11
12 struct Memory
13 {
14
       int size;
15
       int free;
16
       int allocated;
17
       bool isTaken = false;
       int extfrag;
18
       int givenProcessId = -1;
19
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
       int j=0;
30
       for (int i = 0; i < n; i++)
31
32
           while(j<m)</pre>
33
           {
34
                if (mem[j].size >= p[i].size)
35
                {
36
                    mem[j].isTaken = true;
37
                    p[i].isGiven = true;
38
                    mem[j].givenProcessId = p[i].id;
39
                    mem[j].free -= p[i].size;
40
                    mem[j].size -= p[i].size;
41
                    p[i].allocation = j + 1;
42
                    mem[j].allocated = p[i].id;
43
                    break;
44
                }
                j = (j+1)%m;
45
46
           }
47
       }
48 }
49
50 void calcfrag(Process p[], int n, Memory mem[], int m)
51 {
       int flag = 0;
52
53
       for (int i = 0; i < m; i++)
54
           if (mem[i].givenProcessId ==-1)
55
56
           {
57
                flag = 1;
                break;
58
59
           }
60
       }
61
       if (flag == 0)
```

localhost:40561 1/3

```
24/03/2022, 20:55
  62
        {
  63
            external fragmentation = 0;
  64
        }
        else
  65
  66
        {
  67
            for (int i = 0; i < m; i++)
  68
                if (mem[i].isTaken != true || mem[i].givenProcessId==-1)
  69
  70
                    external fragmentation += mem[i].size;
  71
  72
  73
            }
        }
  74
  75
        for (int i = 0; i < m; i++)
  76
  77
  78
            if (mem[i].isTaken != false)
  79
  80
                internal fragmentation += mem[i].free;
  81
            }
        }
  82
  83 }
  84
  85 void printTable(Process P[], int n, Memory mem[], int m, int memorySize[])
  86 {
  87
        for(int i=0;i<m;i++)</pre>
  88
  89
            if(mem[i].free==memorySize[i])
  90
  91
            {
  92
                   mem[i].free=0;
  93
            }
  94
        }
  95
  96
        cout << "\nTable-->(-1 Denotes Unallocated process)\n";
  97
        int i;
  98
        puts("+----+");
  99
        puts("| BNO | Block Size | Process All. | Internal Fragm. |");
 100
        puts("+----+"):
 101
 102
 103
        for (i = 0; i < m; i++)
 104
 105
            printf("| %2d | %2d
                                                               %3d
                                                                   |\n", i+1,
     memorySize[i], mem[i].givenProcessId, mem[i].free);
            puts("+----+");
 106
 107
 108
        cout << "External Fragmentation: " << external_fragmentation << endl;</pre>
 109
        cout << "Internal Fragmentation: " << internal_fragmentation << endl;</pre>
 110
 111 |}
 112
 113 | int main()
 114 {
        cout << "\nEnter the number of memory blocks: ";</pre>
 115
 116
        cin >> m;
 117
        Memory mem[m];
 118
        int memorySize[m];
 119
        for (int i = 0; i < m; i++)
 120
            cout << "\n";
 121
```

localhost:40561 2/3

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

localhost:40561 3/3