

17. Construct a C program to simulate the Least Recently Used paging technique of memory management.

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4
5  int pageFaults(int pages[], int n, int capacity)
6  {
7
8      unordered_set<int> s;
9
10
11      unordered_map<int, int> indexes;
12
13
14      int page_faults = 0;
15      for (int i=0; i<n; i++)
16      {
17
18          if (s.size() < capacity)
19          {
20              if (s.find(pages[i])==s.end())
21              {
22                  s.insert(pages[i]);
23
24
25                  page_faults++;
26              }
27
28              indexes[pages[i]] = i;
29          }
30
31          else
32          {
33
34              if (s.find(pages[i]) == s.end())
35              {
36
37                  int lru = INT_MAX, val;
38                  for (auto it=s.begin(); it!=s.end(); it++)
39                  {
40                      if (indexes[*it] < lru)
```

```

33 {
34
35     if (s.find(pages[i]) == s.end())
36     {
37
38         int lru = INT_MAX, val;
39         for (auto it=s.begin(); it!=s.end(); it++)
40         {
41             if (indexes[*it] < lru)
42             {
43                 lru = indexes[*it];
44                 val = *it;
45             }
46         }
47
48
49         s.erase(val);
50
51
52         s.insert(pages[i]);
53
54
55         page_faults++;
56     }
57
58
59     indexes[pages[i]] = i;
60 }
61 }
62
63 return page_faults;
64 }
65
66 int main()
67 {
68     int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};
69     int n = sizeof(pages)/sizeof(pages[0]);
70     int capacity = 4;
71     cout << pageFaults(pages, n, capacity);
72     return 0;
73 }

```

Output:

C:\Users\kalya\OneDrive\Desktop\7.ipc sm.exe

6

Process exited after 0.09555 seconds with return value 0

Press any key to continue . . .