Nagar Yuwak Shikshan Sanstha's
Yeshwantrao Chavan College of Engineering
(An Autonomous Institution affiliatedtoRashtrasantTukadojiMaharajNagpur University)
Hingna Road, Wanadongri, Nagpur - 441 110
Ph.: 07104-237919, 234623, 329249,329250Fax:07104-232376,Website:www.ycce.edu
Department of Computer Science and Engineering (IOT)

YCCE

Vision

"To become the most preferred institution providing innovative, research and value based, professional education for the society at large".

Mission

YCCE is committed to

- Attract besttalent and create learning ambience
- Practice Innovative teaching-learning & research
- Integrate Industry-Institute Collaborations
- Nurture students towards holistic development and choicest career

Department

Vision of the Department

To be a well-known center for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary arena by developing problem-solving skills through emerging technologies.

Nagar Yuwak Shikshan Sanstha's
Yeshwantrao Chavan College of Engineering
(An Autonomous Institution affiliatedtoRashtrasantTukadojiMaharajNagpur University)
Hingna Road, Wanadongri, Nagpur - 441 110
Ph.: 07104-237919, 234623, 329249,329250Fax:07104-232376,Website:www.ycce.edu
Department of Computer Science and Engineering (IOT)

23CT1402	Lab: Operating Systems
Name of the Student: Parth Bhedurkar	Semester/ Section: 5 A
Roll No: 52	Enrollment Number: 23071495

Sr. No.	COs	POs											PSOs	
		Course Outcomes	1	2	3	4	5	6	7	8	9	10	11	PSO 1	PSO2
1	CO1	Demonstrate the ability to execute Linux process management, memory management, and shell commands to manage system resources efficiently.	3	3	3	-		-	-	-	-	-	3	3	
2	CO2	Develop programs utilizing system calls, thread programming, and page replacement algorithms to simulate and analyze operating system functionalities.	3	3	3	-	-	-	-	-	-	-	3	3	
3	CO3	Design and implement process scheduling, memory allocation, and deadlock detection algorithms to address real-world operating system challenges.	3	3	3	-		-	-	-	-	-	3	3	
		Avg		3	3	-		-	-	-	-	-	3	3	

Practical No. 6

Aim: Simulate the page replacment algorithm
first in first out (FIFO)

Theory:

A simulation of the **First-In, First-Out (FIFO)** page replacement algorithm involves tracking the sequence of page requests (the **reference string**) and how the pages are managed within a limited number of available **frames** in physical memory.

FIFO replaces the page that has been in memory for the **longest time**, regardless of how frequently or recently it has been used.

FIFO Page Replacement Simulation

Initial Setup

- Reference String: The sequence of pages requested by the CPU.

  $$7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1$$

- **Available Frames:** The number of slots in physical memory.

Code:

```
#include <iostream>
#include <vector>
#include <list>
#include <algorithm>

using namespace std;

void fifo_page_replacement(const vector<int>& reference_string, int num_frames) {
```

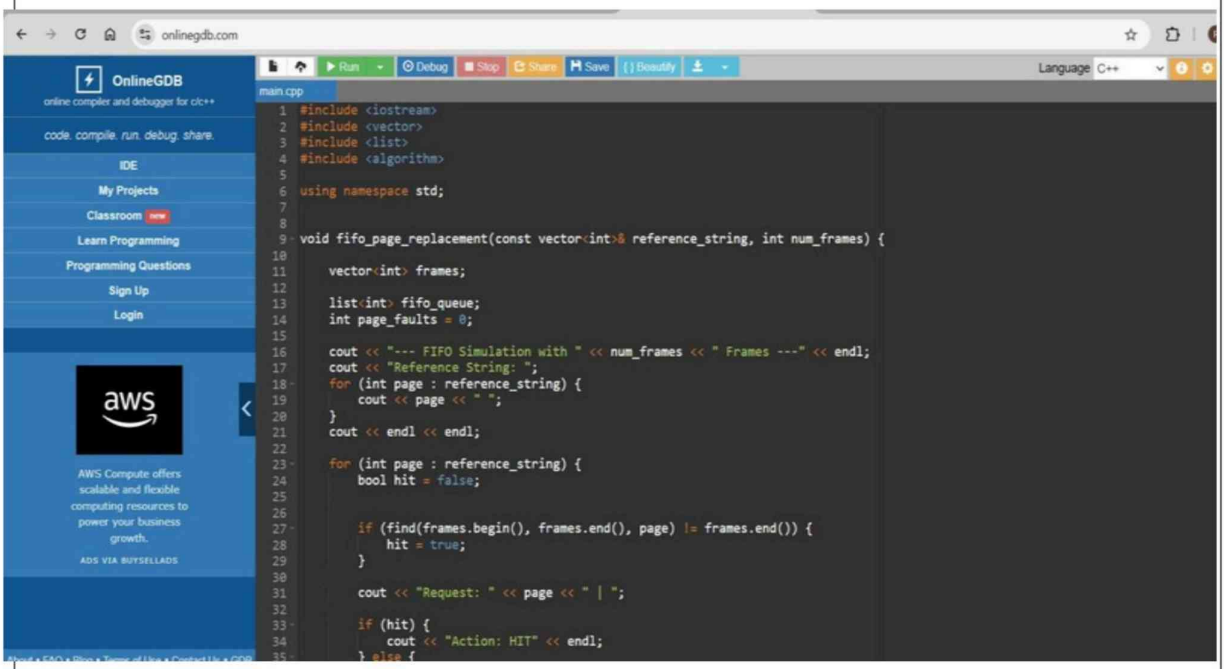
```
    vector<int> frames;


    list<int> fifo_queue;
    int page_faults = 0;


    cout << "--- FIFO Simulation with " << num_frames << " Frames ---" <<
endl;
    cout << "Reference String: ";
    for (int page : reference_string) {
        cout << page << " ";
    }
    cout << endl << endl;


    for (int page : reference_string) {
        bool hit = false;



        if (find(frames.begin(), frames.end(), page) != frames.end()) {
            hit = true;
        }
```

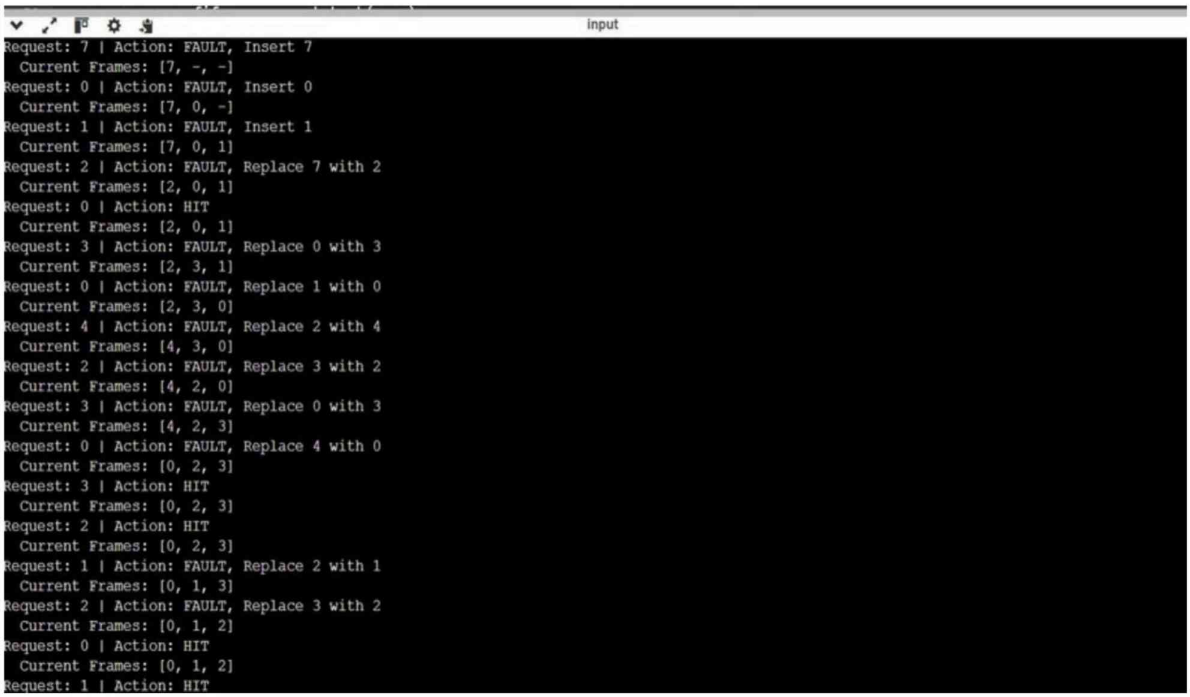
```
cout << "Request: " << page << " | ";


if (hit) {
   cout << "Action: HIT" << endl;
} else {

   page_faults++;



   if (frames.size() < num_frames) {

      frames.push_back(page);
      fifo_queue.push_back(page);
      cout << "Action: FAULT, Insert " << page << endl;
   } else {




      int page_to_evict = fifo_queue.front();
      fifo_queue.pop_front();
```

```
            auto it = find(frames.begin(), frames.end(), page_to_evict);
            if (it != frames.end()) {
               *it = page;
            }


            fifo_queue.push_back(page);

            cout << "Action: FAULT, Replace " << page_to_evict << " with "
<< page << endl;
         }
      }


      cout << " Current Frames: [";
      for (size_t i = 0; i < frames.size(); ++i) {
         cout << frames[i] << (i < frames.size() - 1 ? ", " : "");
      }
      for (int i = 0; i < num_frames - frames.size(); ++i) {
         cout << (frames.empty() && i == 0 ? "" : ", ") << "-";
      }
      cout << "]" << endl;
```

```
    }



    double hit_rate = (double)(reference_string.size() - page_faults) /
reference_string.size() * 100;

    cout << "\n ---------------------------------------- " << endl;

    cout << "Total Requests: " << reference_string.size() << endl;

    cout << "Total Page Faults: " << page_faults << endl;

    cout << "Hit Rate: " << hit_rate << "%" << endl;

}


int main() {

    vector<int> ref_str = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1};



    int num_frames = 3;

    fifo_page_replacement(ref_str, num_frames);

    return 0;

}
```

OUTPUT (SCREEN SHOT) IF ANY:





Conclusion: Simulate the page replacment algorithm is done successfully.