Assignment 6

```
#include <bits/stdc++.h>
using namespace std;

int pageFaults = 0;

void printFrames(const vector<int>& frames) {
    cout << "Frames: ";
    for (int f : frames) cout << f << " ";
    cout << endl;
}

double hitRatio(int hits, int total) {
    return (double)hits / total;
}

double missRatio(int misses, int total) {
    return (double)misses / total;
}

int fcfs(const vector<int>& pages, int frame_size) {
    vector<int> frames;
    int hits = 0;

    for (int page : pages) {
        if (find(frames.begin(), frames.end(), page) ==
frames.end()) {
            if (frames.size() < frame_size)
                frames.push_back(page);
            else {
                frames.erase(frames.begin());
                frames.push_back(page);
```

```
            }
            pageFaults++;
        } else {
            hits++;
        }
        printFrames(frames);
    }
    cout << "Page Faults: " << pageFaults << endl;
    cout << "Hit Ratio: " << hitRatio(hits, pages.size()) << endl;
    cout << "Miss Ratio: " << missRatio(pageFaults, pages.size()) <<
endl;
    return pageFaults;
}

int lru(const vector<int>& pages, int frame_size) {
    vector<int> frames;
    unordered_map<int, int> last_used;
    pageFaults = 0;
    int hits = 0;

    for (int i = 0; i < pages.size(); ++i) {
        int page = pages[i];
        if (find(frames.begin(), frames.end(), page) ==
frames.end()) {
            if (frames.size() < frame_size)
                frames.push_back(page);
            else {
                int lru_index = 0;
                for (int j = 1; j < frames.size(); ++j) {
                    if (last_used[frames[j]] <
last_used[frames[lru_index]])
                        lru_index = j;
                }
                frames[lru_index] = page;
```

```
            }
            pageFaults++;
        } else {
            hits++;
        }
        last_used[page] = i;
        printFrames(frames);
    }
    cout << "Page Faults: " << pageFaults << endl;
    cout << "Hit Ratio: " << hitRatio(hits, pages.size()) << endl;
    cout << "Miss Ratio: " << missRatio(pageFaults, pages.size()) <<
endl;
    return pageFaults;
}

int optimal(const vector<int>& pages, int frame_size) {
    vector<int> frames;
    pageFaults = 0;
    int hits = 0;

    for (int i = 0; i < pages.size(); ++i) {
        int page = pages[i];
        if (find(frames.begin(), frames.end(), page) ==
frames.end()) {
            if (frames.size() < frame_size)
                frames.push_back(page);
            else {
                int furthest_index = -1, replace_index = -1;
                for (int j = 0; j < frames.size(); ++j) {
                    int index = find(pages.begin() + i + 1,
pages.end(), frames[j]) - pages.begin();
                    if (index == pages.size()) {
                        replace_index = j;
                        break;
```

```
                    }
                    if (index > furthest_index) {
                        furthest_index = index;
                        replace_index = j;
                    }
                }
                frames[replace_index] = page;
            }
            pageFaults++;
        } else {
            hits++;
        }
        printFrames(frames);
    }
    cout << "Page Faults: " << pageFaults << endl;
    cout << "Hit Ratio: " << hitRatio(hits, pages.size()) << endl;
    cout << "Miss Ratio: " << missRatio(pageFaults, pages.size()) <<
endl;
    return pageFaults;
}

int main() {
    vector<int> pages;
    int frame_size, choice;

    cout << "Enter the number of frames (minimum 3): ";
    cin >> frame_size;
    if (frame_size < 3) {
        cout << "Frame size must be at least 3." << endl;
        return 1;
    }

    cout << "Enter page reference sequence (end with -1): ";
```

```
    int page;
    while (cin >> page && page != -1) {
        pages.push_back(page);
    }

    cout << "Select Page Replacement Algorithm:\n1. FCFS\n2. LRU\n3.
Optimal\nEnter your choice: ";
    cin >> choice;

    switch (choice) {
        case 1:
            cout << "FCFS Page Replacement:\n";
            fcfs(pages, frame_size);
            break;
        case 2:
            cout << "LRU Page Replacement:\n";
            lru(pages, frame_size);
            break;
        case 3:
            cout << "Optimal Page Replacement:\n";
            optimal(pages, frame_size);
            break;
        default:
            cout << "Invalid choice!" << endl;
            break;
    }

    return 0;
}
```

Output :

```
Enter the number of frames (minimum 3): 3
```

```
Enter page reference sequence (end with -1): 1 3 0 3 5 6 3
-1
Select Page Replacement Algorithm:
1. FCFS
2. LRU
3. Optimal
Enter your choice: 1
FCFS Page Replacement:
Frames: 1
Frames: 1 3
Frames: 1 3 0
Frames: 1 3 0
Frames: 3 0 5
Frames: 0 5 6
Frames: 5 6 3
Page Faults: 6
Hit Ratio: 0.142857
Miss Ratio: 0.857143

LRU Page Replacement:
Frames: 1
Frames: 1 3
Frames: 1 3 0
Frames: 1 3 0
Frames: 5 3 0
Frames: 5 3 6
Frames: 5 3 6
Page Faults: 5
Hit Ratio: 0.285714
```

```
Optimal Page Replacement:
Frames: 1
Frames: 1 3
Frames: 1 3 0
Frames: 1 3 0
Frames: 5 3 0
Frames: 6 3 0
Frames: 6 3 0
Page Faults: 5
Hit Ratio: 0.285714
Miss Ratio: 0.714286
```