#Shruti Baravkar

#121B1B067

Assignment No. 7

Write a to implement paging replacement algorithms :

a) FCFS

b) Least Recently Used (LRU)

c) Optimal algorithm

Code :

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

// Function to implement First Come First Serve (FCFS) paging algorithm
void FCFS(const vector<int>& pages, int capacity) {
   int pageTable[capacity];
   fill_n(pageTable, capacity, -1);
   int pageFaults = 0, hits = 0;
   int pointer = 0;

   cout << "FCFS Output:" << endl;
   for (int i = 0; i < pages.size(); ++i) {
      bool found = false;
      for (int j = 0; j < capacity; ++j) {
         if (pageTable[j] == pages[i]) {
            found = true;
            ++hits;
            break;
         }
      }
```

```
        if (!found) {
            pageTable[pointer] = pages[i];
            pointer = (pointer + 1) % capacity;
            ++pageFaults;
        }
        cout << "Process " << pages[i] << ":\t";
        for (int j = 0; j < capacity; ++j) {
            cout << pageTable[j] << " ";
        }
        cout << endl;
    }

    double hitRatio = (double)hits / pages.size();
    double missRatio = (double)(pageFaults) / pages.size();
    cout << "Hit Ratio: " << hitRatio << endl;
    cout << "Miss Ratio: " << missRatio << endl;
}

// Function to implement Least Recently Used (LRU) paging algorithm
void LRU(const vector<int>& pages, int capacity) {
    int pageTable[capacity];
    fill_n(pageTable, capacity, -1);
    int pageFaults = 0, hits = 0;

    cout << "LRU Output:" << endl;
    for (int i = 0; i < pages.size(); ++i) {
        bool found = false;
        for (int j = 0; j < capacity; ++j) {
            if (pageTable[j] == pages[i]) {
                found = true;
                ++hits;
```

```
            for (int k = j; k > 0; --k) {
                pageTable[k] = pageTable[k - 1];
            }
            pageTable[0] = pages[i];
            break;
        }
    }
    if (!found) {
        for (int j = capacity - 1; j > 0; --j) {
            pageTable[j] = pageTable[j - 1];
        }
        pageTable[0] = pages[i];
        ++pageFaults;
    }
    cout << "Process " << pages[i] << ":\t";
    for (int j = 0; j < capacity; ++j) {
        cout << pageTable[j] << " ";
    }
    cout << endl;
  }

  double hitRatio = (double)hits / pages.size();
  double missRatio = (double)(pageFaults) / pages.size();
  cout << "Hit Ratio: " << hitRatio << endl;
  cout << "Miss Ratio: " << missRatio << endl;
}

// Function to implement Optimal paging algorithm
void Optimal(const vector<int>& pages, int capacity) {
  int pageTable[capacity];
  fill_n(pageTable, capacity, -1);
```

```
int pageFaults = 0, hits = 0;


cout << "Optimal Output:" << endl;
for (int i = 0; i < pages.size(); ++i) {
   bool found = false;
   for (int j = 0; j < capacity; ++j) {
      if (pageTable[j] == pages[i]) {
         found = true;
         ++hits;
         break;
      }
   }
   if (!found) {
      int farthest = -1;
      int replaceIndex = -1;
      for (int j = 0; j < capacity; ++j) {
         int k;
         for (k = i + 1; k < pages.size(); ++k) {
            if (pageTable[j] == pages[k]) {
               break;
            }
         }
         if (k == pages.size()) {
            replaceIndex = j;
            break;
         }
         if (k > farthest) {
            farthest = k;
            replaceIndex = j;
         }
      }
```

```
            pageTable[replaceIndex] = pages[i];
            ++pageFaults;
        }
        cout << "Process " << pages[i] << ":\t";
        for (int j = 0; j < capacity; ++j) {
            cout << pageTable[j] << " ";
        }
        cout << endl;
    }

    double hitRatio = (double)hits / pages.size();
    double missRatio = (double)(pageFaults) / pages.size();
    cout << "Hit Ratio: " << hitRatio << endl;
    cout << "Miss Ratio: " << missRatio << endl;
}

int main() {
    int capacity, n;
    cout << "Enter the number of page frames: ";
    cin >> capacity;
    cout << "Enter the number of pages: ";
    cin >> n;

    vector<int> pages(n);
    cout << "Enter the page reference string: ";
    for (int i = 0; i < n; ++i) {
        cin >> pages[i];
    }

    FCFS(pages, capacity);
    LRU(pages, capacity);
```

```
    Optimal(pages, capacity);

    return 0;
}
```

Output :