# **ASG 08**

**Name:** Yatharth Thakare

**Roll No:** 51

**PRN:** 12111403

**Subject:** OS

**Title: Paging**

1. **FIFO**

Code:

#include <bits/stdc++.h>

using namespace std;

// refrence string : 1 3 0 3 5 6 | this is the page number that the cpu wants to access

// each horz line represents one iteration of the page access

// if whatever is accessed is not there, then it is a page fault.

// frame size : 3 // page slots

int main()

{

int j, n, ref\_str[50], frame[10], no, flag, fcount = 0, hit = 0;

cout << "\nEnter the number of pages cpu wants to access: \n";

cin >> n;

cout << "\nEnter the page numbers: \n";

for (int i = 1; i <= n; i++)

cin >> ref\_str[i];

cout << "\nEnter the number of frames: \n";

cin >> no;

for (int i = 0; i < no; i++)

frame[i] = -1;

j = 0;

cout << "\nref string\t\tpage frames\t\tHit/Fault\n";

for (int i = 1; i <= n; i++)

{

cout << ref\_str[i] << "\t\t";

flag = 0;

for (int k = 0; k < no; k++)

{

if (frame[k] == ref\_str[i])

{

flag = 1;

cout << "\t";

for (k = 0; k < no; k++)

if (frame[k] != -1)

cout << frame[k] << "\t";

else

cout << "\_\t";

cout << "\tH";

hit++;

}

}

if (flag == 0)

{

frame[j] = ref\_str[i];

j = (j + 1) % no;

fcount++;

cout << "\t";

for (int k = 0; k < no; k++)

if (frame[k] != -1)

cout << frame[k] << "\t";

else

cout << "\_\t";

printf("\tF");

}

printf("\n");

}

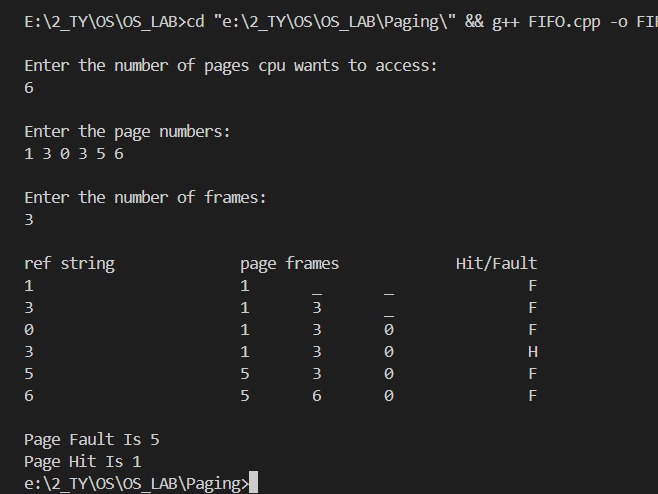
printf("\nPage Fault Is %d", fcount);

printf("\nPage Hit Is %d", hit);

return 0;

}

Output:



1. **LRU**

Code:

#include <stdio.h>

void lruPageReplacement(int pages[], int n, int capacity)

{

int frame[capacity];

int pageFaults = 0;

int hits = 0;

int recentCount = 0;

int recent[capacity];

int hit = 0;

for (int i = 0; i < capacity; i++)

{

frame[i] = -1;

recent[i] = 0;

}

printf("Incoming\tFrame\t\t Hit|Fault\n");

for (int i = 0; i < n; i++)

{

int page = pages[i];

int pageFound = 0;

// Check if the page is already present in the frame

for (int j = 0; j < capacity; j++)

{

if (frame[j] == page)

{

pageFound = 1;

recent[j] = recentCount++;

hits++;

// printf("\tH\n");

hit = 1;

break;

}

}

if (!pageFound)

{

// Page is not present, find the least recently used page in the frame

int lruIndex = 0;

for (int j = 1; j < capacity; j++)

{

if (recent[j] < recent[lruIndex])

{

lruIndex = j;

}

}

// Replace the least recently used page with the new page

frame[lruIndex] = page;

recent[lruIndex] = recentCount++;

pageFaults++;

}

// Print the current page and frame status

printf("%d\t\t", page);

for (int j = 0; j < capacity; j++)

{

if (frame[j] != -1)

printf("%d\t\t", frame[j]);

else

printf("-\t\t");

// printf("\t%d",)

}

hit == 1 ? printf("\tH") : printf("\tF");

printf("\n");

hit = 0;

}

printf("\nTotal Page Faults: %d\n", pageFaults);

printf("\nTotal Page Hits: %d\n", hits);

}

int main()

{

int n;

// Input the number of pages

printf("Enter the number of pages: ");

scanf("%d", &n);

int pages[n];

// Input the sequence of page requests

printf("Enter the sequence of page requests:\n");

for (int i = 0; i < n; i++)

scanf("%d", &pages[i]);

int capacity;

// Input the number of frames

printf("Enter the number of frames: ");

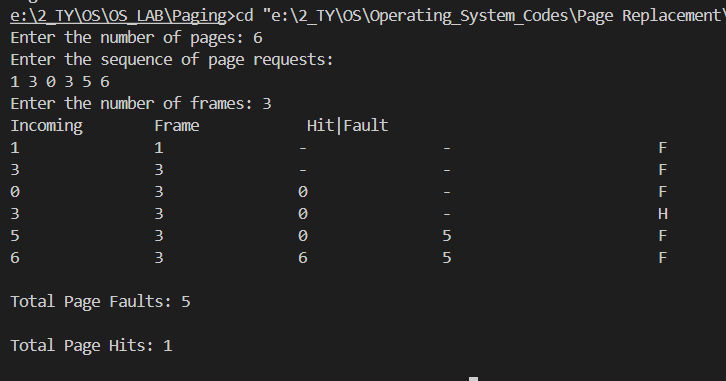
scanf("%d", &capacity);

lruPageReplacement(pages, n, capacity);

return 0;

}

Output :



1. **Optimal**

**Code:**

#include <bits/stdc++.h>

using namespace std;

// page string : 7 0 1 2 0 3 0 4 2 3 0 3 2 //13

// frame size : 4

int minFromMap(unordered\_map<int, int> mp)

{

int min = INT\_MAX;

int min\_key = -1;

for (auto i : mp)

{

if (i.second <= min)

{

min = i.second;

min\_key = i.first;

}

}

// mp.erase(min\_key);

return min\_key;

}

int main()

{

cout << "Enter the number of pages: ";

int n;

cin >> n;

int ref\_str[n];

cout << "Enter the page numbers: ";

for (int i = 0; i < n; i++)

cin >> ref\_str[i];

cout << "Enter the number of frames: ";

int no;

cin >> no;

int frame[no];

for (int i = 0; i < no; i++)

frame[i] = -1;

int j = 0;

unordered\_map<int, int> mp;

// priority\_queue<pair<int, int>,vector<pair<int,int>,greater<pair<int,int>>>> pq;

for (int i = 0; i < n; i++)

{

mp[ref\_str[i]]++;

}

cout << "ref string\t\tpage frames\t\tHit/Fault\n";

int fcount = 0, hit = 0, flag = 0;

for (int i = 0; i <= n; i++)

{

cout << ref\_str[i] << "\t\t";

flag = 0;

for (int k = 0; k < no; k++)

{

if (frame[k] == ref\_str[i])

{

mp[ref\_str[i]]--;

flag = 1;

cout << "\t";

for (k = 0; k < no; k++)

if (frame[k] != -1)

cout << frame[k] << "\t";

else

cout << "\_\t";

cout << "\tH";

hit++;

}

}

if (flag == 0)

{

if (j < no)

{

frame[j] = ref\_str[i];

mp[ref\_str[i]]--;

j = (j + 1) % no;

fcount++;

cout << "\t";

for (int k = 0; k < no; k++)

if (frame[k] != -1)

cout << frame[k] << "\t";

else

cout << "\_\t";

printf("\tF\n");

continue;

}

else

{

for (int x = 0; x < no; x++)

{

if (frame[x] == minFromMap(mp))

{

frame[x] = ref\_str[i];

break;

}

}

// frame[j] = ref\_str[i];

mp[ref\_str[i]]--;

fcount++;

cout << "\t";

for (int k = 0; k < no; k++)

if (frame[k] != -1)

cout << frame[k] << "\t";

else

cout << "\_\t";

printf("\tF");

}

}

printf("\n");

}

printf("\nPage Fault Is %d", fcount);

printf("\nPage Hit Is %d", hit);

}

Output:

