Operating Systems

Experiment 9

Name: Kartik Jolapara **SAP ID: 60004200107**

Div.: B1 **Branch: Computer Engineering**

Aim -

To implement page replacement algorithms.

Theory -

Page Replacement Algorithms

The page replacement algorithm decides which memory page is to be replaced. The process of replacement is sometimes called swap out or write to disk. Page replacement is done when the requested page is not found in the main memory (page fault).

Page Replacement Algorithms:

1. First In First Out (FIFO)

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

2. Least Recently Used (LRU)

In this algorithm, page will be replaced which is least recently used.

Page Fault – A page fault happens when a running program accesses a memory page that is mapped into the virtual address space but not loaded in physical memory.

Code –

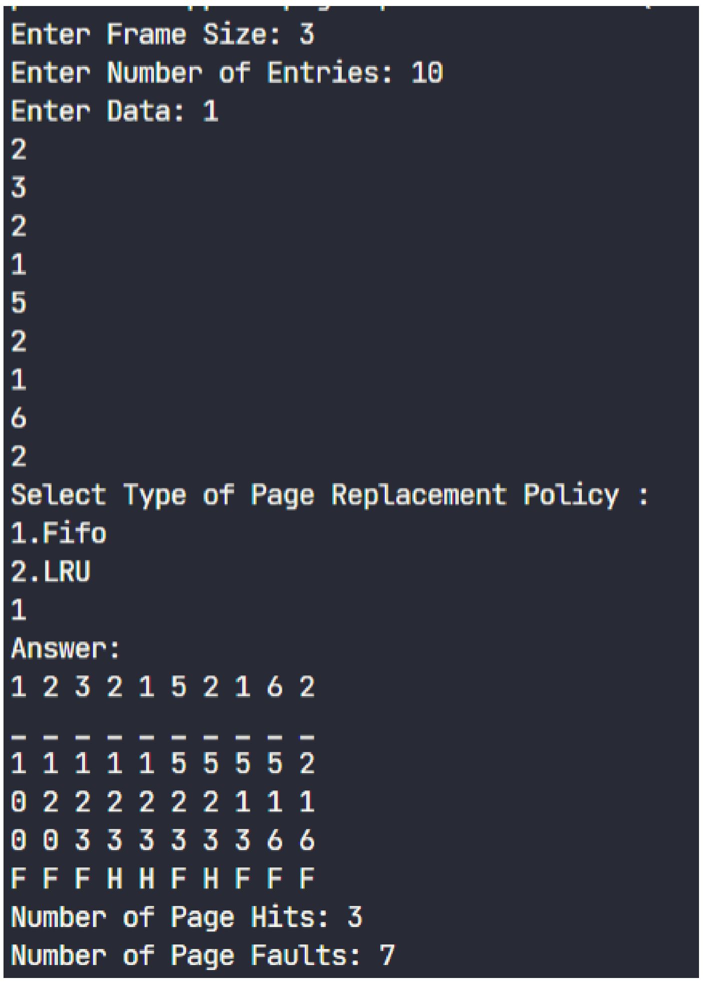
```cpp
#include <iostream>

using namespace std;
int fSize, n;

void answer(int *data, int **ans)
{
    int hit = 0;
    cout << "Answer:" << endl;
    for (int i = 0; i < n; i++)
    {
        cout << data[i] << " ";
    }
    cout << endl;
    for (int i = 0; i < n; i++)
    {
        cout << "_"
             << " ";
    }
    cout << endl;
    for (int j = 0; j < fSize; j++)
    {
        for (int i = 0; i < n; i++)
        {
            cout << ans[i][j] << " ";
        }
        cout << endl;
    }
    for (int i = 0; i < n; i++)
    {
        if (ans[i][fSize] == -1)
        {
            cout << "F ";
        }
        else
        {
            hit++;
            cout << "H ";
        }
    }
    cout << endl
         << "Number of Page Hits: " << hit;
    cout << endl
         << "Number of Page Faults: " << (n - hit);
}
void fifo(int *data, int **ans)
```

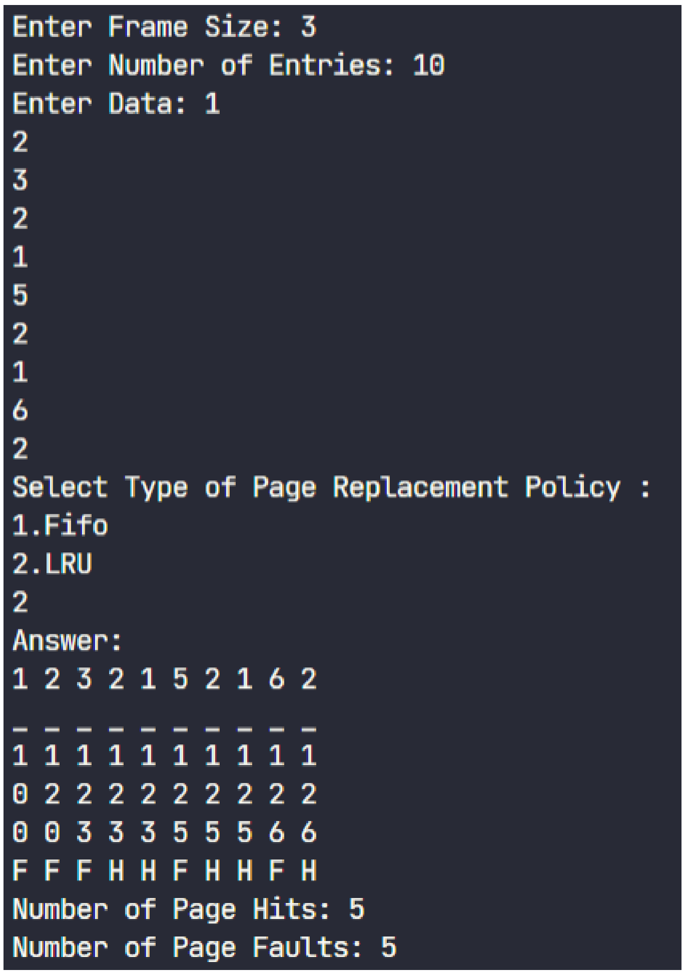
```
{
    int in = 0, temp = 0, counter = fSize;
    for (int i = 0; i < n; i++)
    {
        if (i < counter)
        {
            for (int j = 0; j < fSize; j++)
            {
                if (j < (i + fSize - counter))
                {
                    ans[i][j] = ans[i - 1][j];
                    if (data[i] == ans[i - 1][j])
                    {
                        temp = 1;
                    }
                }
                else
                    ans[i][j] = 0;
            }
            if (temp == 1)
            {
                ans[i][fSize] = 0;
                counter++;
            }
            else
            {
                ans[i][(i + fSize - counter)] = data[i];
                ans[i][fSize] = -1;
            }
            temp = 0;
        }
        else
        {
            for (int j = 0; j < fSize; j++)
            {
                ans[i][j] = ans[i - 1][j];
                if (data[i] == ans[i - 1][j])
                {
                    temp = 1;
                }
            }
            if (temp == 1)
            {
                ans[i][fSize] = 0;
            }
            else
            {
                ans[i][in] = data[i];
                in = (in + 1) % fSize;
                ans[i][fSize] = -1;
            }
        }
        temp = 0;
    }
    answer(data, ans);
}
void lru(int *data, int **ans)
{
    int in = 0, temp = 0, counter = fSize, key = 0;
    int *q = new int[fSize];
    for (int i = 0; i < n; i++)
    {
        if (i < counter)
        {
            for (int j = 0; j < fSize; j++)
            {
                if (j < (i + fSize - counter))
                {
                    ans[i][j] = ans[i - 1][j];
                    if (data[i] == ans[i - 1][j])
                    {
                        key = j;
                        temp = 1;
                    }
                }
                else
                    ans[i][j] = 0;
            }
            if (temp == 1)
            {
                ans[i][fSize] = 0;
                q[key] = i;
                counter++;
            }
            else
            {
                ans[i][(i + fSize - counter)] = data[i];
                ans[i][fSize] = -1;
                q[(i + fSize - counter)] = i;
            }
            temp = 0;
        }
        else
        {
            for (int j = 0; j < fSize; j++)
            {
                ans[i][j] = ans[i - 1][j];
                if (data[i] == ans[i - 1][j])
                {
                    temp = 1;
                    q[j] = i;
                }
            }
            if (temp == 1)
            {
                ans[i][fSize] = 0;
            }
            else
            {
                in = 0;
                for (int j = 0; j < fSize; j++)
                {
                    if (q[in] > q[j])
                    {
                        in = j;
                    }
                }
                ans[i][in] = data[i];
                q[in] = i;
                ans[i][fSize] = -1;
            }
        }
        temp = 0;
    }
    answer(data, ans);
}
int main()
{
    int choice;
    cout << "Enter Frame Size: ";
    cin >> fSize;
    cout << "Enter Number of Entries: ";
    cin >> n;
    int *data = new int[n];
    cout << "Enter Data: ";
    for (int i = 0; i < n; i++)
    {
        cin >> data[i];
    }
    int **ans = new int *[n];
    for (int i = 0; i < n; i++)
    {
        ans[i] = new int[fSize + 1];
    }
    cout << "Select Type of Page Replacement Policy :" << endl;
    cout << "1.Fifo\n2.LRU\n";
    cin >> choice;
    switch (choice)
    {
    case 1:
        fifo(data, ans);
        break;
    case 2:
        lru(data, ans);
        break;
    default:
        break;
    }
    return 0;
}
```

Output -



```
Enter Frame Size: 3
Enter Number of Entries: 10
Enter Data: 1
2
3
2
1
5
2
1
6
2
Select Type of Page Replacement Policy :
1.Fifo
2.LRU
1
Answer:
1 2 3 2 1 5 2 1 6 2

_ _ _ _ _ _ _ _ _ _
1 1 1 1 1 5 5 5 5 2
0 2 2 2 2 2 2 1 1 1
0 0 3 3 3 3 3 3 6 6
F F F H H F H F F F
Number of Page Hits: 3
Number of Page Faults: 7
```



```
Enter Frame Size: 3
Enter Number of Entries: 10
Enter Data: 1
2
3
2
1
5
2
1
6
2
Select Type of Page Replacement Policy :
1.Fifo
2.LRU
2
Answer:
1 2 3 2 1 5 2 1 6 2

_ _ _ _ _ _ _ _ _ _
1 1 1 1 1 1 1 1 1 1
0 2 2 2 2 2 2 2 2 2
0 0 3 3 3 5 5 5 6 6
F F F H H F H H F H
Number of Page Hits: 5
Number of Page Faults: 5
```

Conclusion

Page replacement algorithms 1.) FIFO and 2.) LRU are successfully executed.