Fr. Conceicao Rodrigues College of Engineering **Department of Computer Engineering**			
Student's Roll No	**9913**	**Students Name**	**Mark Lopes**
Date of Performance		**SE Computer – Div**	**A**

Aim: Study Paging

Lab Outcome:

CSL403.4: Implement various memory management techniques and evaluate their performances.

Problem Statements:
Implement various page replacement policies

(a)First In First Out **(b)Least Recently Used**

1. Find the number of Page hits, Page Miss, Page hit ratio, Page Miss ratio.

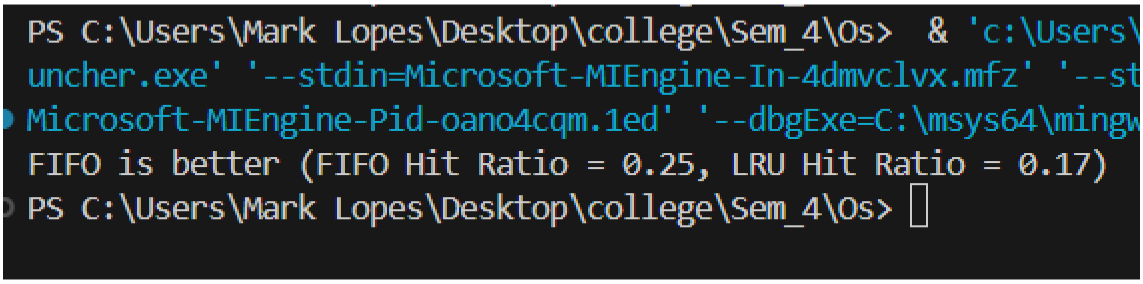
2. Compare the results of both algorithms for a page reference string.

References:

https://www.youtube.com/watch?v=ET43MRKRuYM&list=PLIY8eNdw5tW-BxRY0yK3fYTYVqytw8qhp&index=4
https://www.youtube.com/watch?v=L8BEoRRUVRE&list=PLIY8eNdw5tW-BxRY0yK3fYTYVqytw8qhp&index=6
https://www.youtube.com/watch?v=LCPFjNxQIVU&list=PLIY8eNdw5tW-BxRY0yK3fYTYVqytw8qhp&index=7

```
#include <stdio.h>
#include <stdbool.h>

// Function to calculate hit and miss ratios
void calculate_ratios(int hits, int misses, float *hit_ratio, float
*miss_ratio)
{
    int total = hits + misses;
    *hit_ratio = (float)hits / total;
    *miss_ratio = (float)misses / total;
}

// Function to check if a page is present in the page table
bool isInPageTable(int page, int page_table[], int capacity)
{
    for (int i = 0; i < capacity; i++)
    {
        if (page_table[i] == page)
        {
            return true;
        }
    }
    return false;
}

// FIFO page replacement algorithm
void fifo(int page_reference_string[], int length, int capacity, float
*fifo_hit_ratio, float *fifo_miss_ratio)
{
    int hits = 0;
    int misses = 0;
    int page_table[capacity]; // Page table to store pages

    // Initialize page table with -1 (indicating empty)
    for (int i = 0; i < capacity; i++)
    {
        page_table[i] = -1;
    }

    int page_table_index = 0; // Index to track the next page to replace

    for (int i = 0; i < length; i++)
    {
        int page = page_reference_string[i];

        if (isInPageTable(page, page_table, capacity))
        {
            hits++;
        }
        else
        {
            misses++;

            // Replace the oldest page in the page table
            page_table[page_table_index] = page;
            page_table_index = (page_table_index + 1) % capacity; // Move
index to next position
        }
    }
    // Calculate hit ratio and miss ratio for FIFO
    calculate_ratios(hits, misses, fifo_hit_ratio, fifo_miss_ratio);
}

// LRU page replacement algorithm
void lru(int page_reference_string[], int length, int capacity, float
*lru_hit_ratio, float *lru_miss_ratio)
{
    int hits = 0;
    int misses = 0;
    int page_table[capacity]; // Page table to store pages

    // Initialize page table with -1 (indicating empty)
    for (int i = 0; i < capacity; i++)
    {
        page_table[i] = -1;
    }

    for (int i = 0; i < length; i++)
    {
        int page = page_reference_string[i];
        bool page_found = false;

        // Check if page is already in the page table
        for (int j = 0; j < capacity; j++)
        {
            if (page_table[j] == page)
            {
                // Move the page to the end (most recently used)
                int temp = page_table[j];
                for (int k = j; k < capacity - 1; k++)
                {
                    page_table[k] = page_table[k + 1];
                }
                page_table[capacity - 1] = temp;
                page_found = true;
                hits++;
                break;
            }
        }

        if (!page_found)
        {
            // Page fault, replace the least recently used page
            for (int j = 0; j < capacity - 1; j++)
            {
                page_table[j] = page_table[j + 1];
            }
            page_table[capacity - 1] = page;
            misses++;
        }
    }

    // Calculate hit ratio and miss ratio for LRU
    calculate_ratios(hits, misses, lru_hit_ratio, lru_miss_ratio);
}

int main()
{
    // Page reference string and capacity of the page table
    int page_reference_string[] = {1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5};
    int length = 12;
    int capacity = 3;

    float fifo_hit_ratio, fifo_miss_ratio;
    float lru_hit_ratio, lru_miss_ratio;

    fifo(page_reference_string, length, capacity, &fifo_hit_ratio,
&fifo_miss_ratio);
    lru(page_reference_string, length, capacity, &lru_hit_ratio,
&lru_miss_ratio);

    // Compare hit ratios of FIFO and LRU to determine which algorithm is
better
    if (fifo_hit_ratio > lru_hit_ratio)
    {
        printf("FIFO is better (FIFO Hit Ratio = %.2f, LRU Hit Ratio =
%.2f)\n", fifo_hit_ratio, lru_hit_ratio);
    }
    else
    {
        printf("LRU is better (FIFO Hit Ratio = %.2f, LRU Hit Ratio =
%.2f)\n", fifo_hit_ratio, lru_hit_ratio);
    }
```

```
    return 0;
}
```



```
PS C:\Users\Mark Lopes\Desktop\college\Sem_4\Os>  & 'c:\Users\
uncher.exe' '--stdin=Microsoft-MIEngine-In-4dmvclvx.mfz' '--st
Microsoft-MIEngine-Pid-oano4cqm.1ed' '--dbgExe=C:\msys64\mingw
FIFO is better (FIFO Hit Ratio = 0.25, LRU Hit Ratio = 0.17)
PS C:\Users\Mark Lopes\Desktop\college\Sem_4\Os> 
```

On time Submission(2)	**Knowledge of Topic(4)**	**Implementation and Demonstraion(4)**	**Total (10)**
Signature of Faculty		**Date of Submission**	