Name : Bramha Nimbalkar

Roll No: 7

Div : E

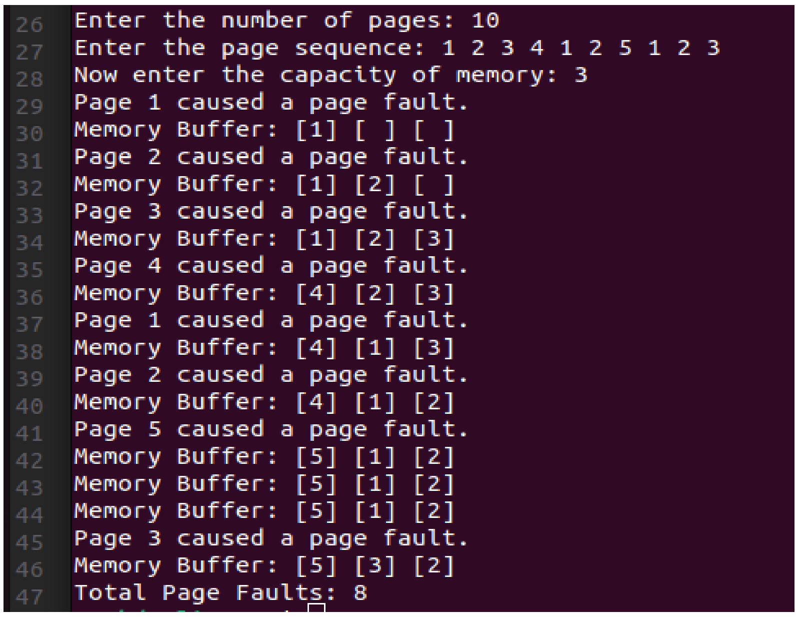
ASSIGNMENT 10

Problem : Implement in C language following page replacement algorithm

1. First In First Out(FIFO) page replacement algortihm

Code:

```
#include <stdio.h>

#define MAX_PAGES 50

void fifoPageReplacement(int pages[], int n, int capacity) {
    int memory[capacity];
    int pageFaults = 0;
    int pointer = 0;

    for (int i = 0; i < capacity; i++) {
        memory[i] = -1;
    }

    for (int i = 0; i < n; i++) {
        int currentPage = pages[i];
        int pagePresent = 0;


        for (int j = 0; j < capacity; j++) {
            if (memory[j] == currentPage) {
                pagePresent = 1;
                break;
            }
        }
        if (!pagePresent) {

            printf("Page %d caused a page fault.\n", currentPage);
            pageFaults++;

            memory[pointer] = currentPage;


            pointer = (pointer + 1) % capacity;
        }


        printf("Memory: ");
        for (int j = 0; j < capacity; j++) {
            if (memory[j] == -1) {
                printf("[ ] ");
            } else {
                printf("[%d] ", memory[j]);
            }
        }
        printf("\n");
    }

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

int main() {
    int pages[MAX_PAGES];
    int n, capacity;


    printf("Enter the number of pages: ");
    scanf("%d", &n);

    printf("Enter the page sequence: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &pages[i]);
    }

    printf("Enter the capacity of memory: ");
    scanf("%d", &capacity);


    fifoPageReplacement(pages, n, capacity);
    return 0;
}
```

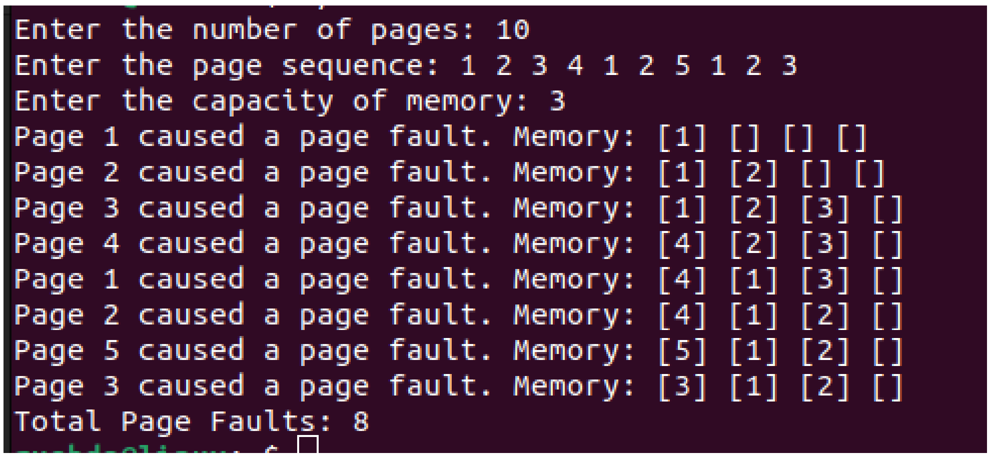


```
Enter the number of pages: 10
Enter the page sequence: 1 2 3 4 1 2 5 1 2 3
Now enter the capacity of memory: 3
Page 1 caused a page fault.
Memory Buffer: [1] [ ] [ ]
Page 2 caused a page fault.
Memory Buffer: [1] [2] [ ]
Page 3 caused a page fault.
Memory Buffer: [1] [2] [3]
Page 4 caused a page fault.
Memory Buffer: [4] [2] [3]
Page 1 caused a page fault.
Memory Buffer: [4] [1] [3]
Page 2 caused a page fault.
Memory Buffer: [4] [1] [2]
Page 5 caused a page fault.
Memory Buffer: [5] [1] [2]
Memory Buffer: [5] [1] [2]
Memory Buffer: [5] [1] [2]
Page 3 caused a page fault.
Memory Buffer: [5] [3] [2]
Total Page Faults: 8
```

2. Least Recently Used(LRU) page replacement algorithm

Code

```
#include <stdio.h>

#define MAX_PAGES 100

void lruPageReplacement(int pages[], int n, int capacity) {
    int memory[capacity];
    int count[capacity];
    int pageFaults = 0;

    for (int i = 0; i < capacity; i++) {
        memory[i] = -1;
        count[i] = 0;
    }

    for (int i = 0; i < n; i++) {
        int currentPage = pages[i];
        int pagePresent = 0;

        for (int j = 0; j < capacity; j++) {
            if (memory[j] == currentPage) {
                pagePresent = 1;
                count[j] = 0; // Reset count for the accessed page
                break;
            }
        }

        if (!pagePresent) {

            printf("Page %d caused a page fault. Memory: [", currentPage);
            pageFaults++;

            int maxCountIndex = 0;
            for (int j = 1; j < capacity; j++) {
                if (count[j] > count[maxCountIndex]) {
                    maxCountIndex = j;
                }
            }

            memory[maxCountIndex] = currentPage;

            for (int j = 0; j < capacity; j++) {
                if (j != maxCountIndex) {
                    count[j]++;
                } else {
                    count[j] = 0;
                }
            }

            printf("%d] [", memory[0]);
            for (int j = 1; j < capacity; j++) {
                if (memory[j] == -1) {
                    printf("] [");
                } else {
                    printf("%d] [", memory[j]);
                }
            }
            printf("]\n");
        }
    }

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

int main() {
    int pages[MAX_PAGES];
    int n, capacity;
    printf("Enter the number of pages: ");
    scanf("%d", &n);

    printf("Enter the page sequence: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &pages[i]);
    }

    printf("Enter the capacity of memory: ");
    scanf("%d", &capacity);

    lruPageReplacement(pages, n, capacity);

    return 0;
}
```



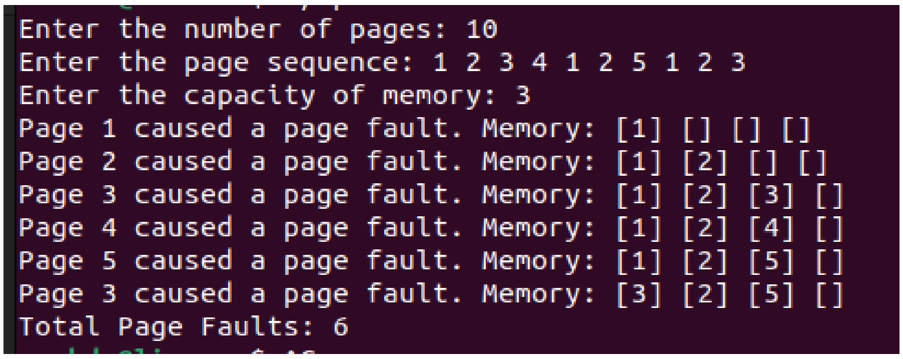
```
Enter the number of pages: 10
Enter the page sequence: 1 2 3 4 1 2 5 1 2 3
Enter the capacity of memory: 3
Page 1 caused a page fault. Memory: [1] [] [] []
Page 2 caused a page fault. Memory: [1] [2] [] []
Page 3 caused a page fault. Memory: [1] [2] [3] []
Page 4 caused a page fault. Memory: [4] [2] [3] []
Page 1 caused a page fault. Memory: [4] [1] [3] []
Page 2 caused a page fault. Memory: [4] [1] [2] []
Page 5 caused a page fault. Memory: [5] [1] [2] []
Page 3 caused a page fault. Memory: [3] [1] [2] []
Total Page Faults: 8
```

3. Optimal page replacement algorithm

Code

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

#define MAX_PAGES 100

void optimalPageReplacement(int pages[], int n, int capacity) {
    int memory[capacity];
    int nextUse[MAX_PAGES];
    int pageFaults = 0;

    for (int i = 0; i < capacity; i++) {
        memory[i] = -1;
        nextUse[i] = INT_MAX;
    }

    for (int i = 0; i < n; i++) {
        int currentPage = pages[i];
        int pagePresent = 0;

        for (int j = 0; j < capacity; j++) {
            if (memory[j] == currentPage) {
                pagePresent = 1;
                break;
            }
        }

        if (!pagePresent) {

            printf("Page %d caused a page fault. Memory: [", currentPage);
            pageFaults++;

            int replaceIndex = 0;
            int farthestUse = -1;

            for (int j = 0; j < capacity; j++) {
                int nextPageUse = -1;
                for (int k = i + 1; k < n; k++) {
                    if (pages[k] == memory[j]) {
                        nextPageUse = k;
                        break;
                    }
                }

                if (nextPageUse == -1) {
                    replaceIndex = j;
                    break;
                } else if (nextPageUse > farthestUse) {
                    farthestUse = nextPageUse;
                    replaceIndex = j;
                }
            }
```

```
            memory[replaceIndex] = currentPage;
            nextUse[replaceIndex] = farthestUse;

                if (memory[j] == -1) {
                    printf("] [");
                } else {
                    printf("%d] [", memory[j]);
                }
            }
            printf("]\n");
        }
    }

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

int main() {
    int pages[MAX_PAGES];
    int n, capacity;

    printf("Enter the number of pages: ");
    scanf("%d", &n);

    printf("Enter the page sequence: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &pages[i]);
    }

    printf("Enter the capacity of memory: ");
    scanf("%d", &capacity);

    optimalPageReplacement(pages, n, capacity);

    return 0;
}
```



```
Enter the number of pages: 10
Enter the page sequence: 1 2 3 4 1 2 5 1 2 3
Enter the capacity of memory: 3
Page 1 caused a page fault. Memory: [1] [] [] []
Page 2 caused a page fault. Memory: [1] [2] [] []
Page 3 caused a page fault. Memory: [1] [2] [3] []
Page 4 caused a page fault. Memory: [1] [2] [4] []
Page 5 caused a page fault. Memory: [1] [2] [5] []
Page 3 caused a page fault. Memory: [3] [2] [5] []
Total Page Faults: 6
```