Ex. No.: 11a) **231901039**

Date: 7/4/2025

FIFO PAGE REPLACEMENT

Aim:

To find out the number of page faults that occur using First-in First-out (FIFO) page replacement technique.

Algorithm:

1. Declare the size with respect to page length
2. Check the need of replacement from the page to memory
3. Check the need of replacement from old page to new page in memory 4.
   Form a queue to hold all pages
5. Insert the page require memory into the queue
6. Check for bad replacement and page fault
7. Get the number of processes to be inserted
8. Display the values

Program Code:

```
#include <stdio.h>

void fifoPageReplacement(int pages[], int
pageCount, int frameCount) {
    int frame[frameCount];
    int front = 0, pageFaults = 0;

    for (int i = 0; i < frameCount; i++)
        frame[i] = -1;

    for (int i = 0; i < pageCount; i++) {
        int found = 0;

        // Check if page is already in frame
        for (int j = 0; j < frameCount; j++) {
            if (frame[j] == pages[i]) {
                found = 1;
                break;
            }
        }

        // If not found, it's a page fault
        if (!found) {
            frame[front] = pages[i];
```

```
            front = (front + 1) % frameCount;
            pageFaults++;
        }

        // Display frame state
        printf("Page %d: ", pages[i]);
        for (int j = 0; j < frameCount; j++) {
            if (frame[j] != -1)
                printf("%d ", frame[j]);
            else
                printf("- ");
        }
        if (!found)
            printf("(Page Fault)");
        printf("\n");
    }

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

int main() {
    int pageCount, frameCount;

    printf("Enter the number of pages: ");
    scanf("%d", &pageCount);
    int pages[pageCount];
    printf("Enter the page reference string:
");
    for (int i = 0; i < pageCount; i++) {
        scanf("%d", &pages[i]);
    }
    printf("Enter the number of frames: ");
    scanf("%d", &frameCount);

    fifoPageReplacement(pages, pageCount,
frameCount);
return 0;
}
```

Sample Output:

```
[root@localhost student]# python fifo.py

Enter the size of reference string: 20
Enter [ 1] : 7
```

```
Enter [ 2] : 0
Enter [ 3] : 1
Enter [ 4] : 2
Enter [ 5] : 0
Enter [ 6] : 3
Enter [ 7] : 0
Enter [ 8] : 4
Enter [ 9] : 2
Enter [10] : 3
Enter [11] : 0
Enter [12] : 3
Enter [13] : 2
Enter [14] : 1
Enter [15] : 2
Enter [16] : 0
Enter [17] : 1
Enter [18] : 7
Enter [19] : 0
Enter [20] : 1

Enter page frame size : 3

7
->
7  -
-  0
->
7  0
-  1
->
7  0
1
2 -> 2 0 1
0 -> No Page Fault


3 -> 2 3 1
0 -> 2 3 0
4 -> 4 3 0
2 -> 4 2 0
3 -> 4 2 3
0 -> 0 2 3
3 -> No Page Fault
2  ->
No  Page  Fault
1 -> 0 1 3
2 -> 0 1 2
0 -> No Page Fault
1 -> No Page Fault
```

```
        7 -> 7 1 2
0 -> 7 0 2
1 -> 7 0 1   Total page faults: 15.
        [root@localhost student]#
```

Result :

Ex. No.: 11b) **231901054**

Date: 9/4/2025

LRU

Aim:

To write a c program to implement LRU page replacement algorithm.

Algorithm:

1: Start the process
2: Declare the size
3: Get the number of pages to be inserted
4: Get the value
5: Declare counter and stack
6: Select the least recently used page by counter value 7: Stack them according the selection.
8: Display the values
9: Stop the process

Program Code:

```
#include <stdio.h>

int findLRU(int time[], int frameCount) {
    int min = time[0], pos = 0;
    for (int i = 1; i < frameCount; i++) {
        if (time[i] < min) {
            min = time[i];
            pos = i;
        }
    }
    return pos;
}

void lruPageReplacement(int pages[], int
pageCount, int frameCount) {
    int frame[frameCount], time[frameCount];
    int pageFaults = 0, counter = 0;
    int flag1, flag2;

    for (int i = 0; i < frameCount; i++) {
        frame[i] = -1;
        time[i] = 0;
    }
```

```
    for (int i = 0; i < pageCount; i++) {
        flag1 = flag2 = 0;

        for (int j = 0; j < frameCount; j++) {
            if (frame[j] == pages[i]) {
                counter++;
                time[j] = counter;
                flag1 = flag2 = 1;
                break;
            }
        }

        if (!flag1) {
            for (int j = 0; j < frameCount;
j++) {
                if (frame[j] == -1) {
                    counter++;
                    frame[j] = pages[i];
                    time[j] = counter;
                    flag2 = 1;
                    pageFaults++;
                    break;
                }
            }
        }

        if (!flag2) {
            int pos = findLRU(time,
frameCount);
            counter++;
            frame[pos] = pages[i];
            time[pos] = counter;
            pageFaults++;
        }

        printf("Page %d: ", pages[i]);
        for (int j = 0; j < frameCount; j++) {
            if (frame[j] != -1)
                printf("%d ", frame[j]);
            else
                printf("- ");
        }
        printf("\n");
    }
```

```
    printf("\nTotal Page Faults = %d\n",
pageFaults);
}

int main() {
    int pageCount, frameCount;

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

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

    printf("Enter the number of frames: ");
    scanf("%d", &frameCount);

    lruPageReplacement(pages, pageCount,
frameCount);

    return 0;
}
```

Sample Output :

Enter number of frames: 3
Enter number of pages: 6
Enter reference string: 5 7 5 6 7 3
5 -1 -1
5 7 -1
5 7 -1
5 7 6
5 7 6
3 7 6
Total Page Faults = 4

Result:

Ex. No.: 11c) **231901054**
Date:12/4/2025

Optimal

Aim:

To write a c program to implement Optimal page replacement algorithm.

ALGORITHM:

1. Start the process
2. Declare the size
3. Get the number of pages to be inserted
4. Get the value
5. Declare counter and stack
6. Select the least frequently used page by counter value
7. Stack them according the selection.
8. Display the values
9. Stop the process

PROGRAM:

```
#include <stdio.h>

int isHit(int page, int frame[], int
frameCount) {
    for (int i = 0; i < frameCount; i++) {
        if (frame[i] == page)
            return 1;
    }
    return 0;
}

int predict(int pages[], int frame[], int
pageCount, int index, int frameCount) {
    int res = -1, farthest = index;
    for (int i = 0; i < frameCount; i++) {
        int j;
```

```
        for (j = index; j < pageCount; j++) {
            if (frame[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }

        // If page not found in future,
replace it immediately
        if (j == pageCount)
            return i;
    }

    // If all pages are found in future
    return (res == -1) ? 0 : res;
}

void optimalPageReplacement(int pages[], int
pageCount, int frameCount) {
    int frame[frameCount];
    int pageFaults = 0;
    int filled = 0;

    for (int i = 0; i < frameCount; i++)
        frame[i] = -1;

    for (int i = 0; i < pageCount; i++) {
        printf("Page %d: ", pages[i]);

        // Check if page is already in frame
        if (isHit(pages[i], frame,
frameCount)) {
            printf("Hit\n");
        } else {
            if (filled < frameCount) {
                frame[filled++] = pages[i];
            } else {
                int pos = predict(pages,
frame, pageCount, i + 1, frameCount);
                frame[pos] = pages[i];
            }
            pageFaults++;
```

```
            // Print frame content
            for (int j = 0; j < frameCount;
j++) {
                if (frame[j] != -1)
                    printf("%d ", frame[j]);
                else
                    printf("- ");
            }
            printf("(Page Fault)\n");
        }
    }

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

int main() {
    int pageCount, frameCount;
    printf("Enter the number of pages: ");
    scanf("%d", &pageCount);

    int pages[pageCount];
    printf("Enter the page reference string:
");
    for (int i = 0; i < pageCount; i++) {
        scanf("%d", &pages[i]);
    }
    printf("Enter the number of frames: ");
    scanf("%d", &frameCount);

    optimalPageReplacement(pages, pageCount,
frameCount);
return 0;
}
```

Output: