Ex. No.: 11a) FIFO PAGE REPLACEMENT
Date:23.4.25

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
7. Display the values

Program code:

```
#include <stdio.h>

int main() {
    int referenceLength, frameSize;
    printf("Enter the size of reference string: ");
    scanf("%d", &referenceLength);

    int referenceString[referenceLength];

    for (int i = 0; i < referenceLength; i++) {
        printf("Enter [%2d] : ", i + 1);
        scanf("%d", &referenceString[i]);
    }

    printf("Enter page frame size : ");
    scanf("%d", &frameSize);

    int frames[frameSize];
    int front = 0, pageFaults = 0;

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

    printf("\nPage Replacement Process:\n");
```

```
    for (int i = 0; i < referenceLength; i++) {
        int currentPage = referenceString[i];
        int found = 0;

        for (int j = 0; j < frameSize; j++) {
            if (frames[j] == currentPage) {
                found = 1;
                break;
            }
        }

        if (!found) {
            frames[front] = currentPage;
            front = (front + 1) % frameSize;
            pageFaults++;
        }

        printf("%d -> ", currentPage);
        for (int j = 0; j < frameSize; j++) {
            if (frames[j] == -1)
                printf("- ");
            else
                printf("%d ", frames[j]);
        }

        if (!found)
            printf("Page Fault\n");
        else
            printf("No Page Fault\n");
    }

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

    return 0;
}
```

Sample Output:

```
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

Page Replacement Process:
7 -> 7 - - Page Fault
0 -> 7 0 - Page Fault
1 -> 7 0 1 Page Fault
2 -> 2 0 1 Page Fault
0 -> 2 0 1 No Page Fault
3 -> 0 3 1 Page Fault
0 -> 0 3 1 No Page Fault
4 -> 3 4 1 Page Fault
2 -> 4 2 1 Page Fault
3 -> 2 3 1 Page Fault
0 -> 3 0 1 Page Fault
3 -> 3 0 1 No Page Fault
2 -> 0 2 1 Page Fault
1 -> 2 1 1 Page Fault
2 -> 2 1 1 No Page Fault
0 -> 2 1 0 Page Fault
1 -> 1 0 0 Page Fault
7 -> 0 7 0 Page Fault
0 -> 0 7 0 No Page Fault
1 -> 7 0 1 Page Fault

Total Page Faults = 15

Result :

Thus the fifo has been successfully executed.

Ex:11b **LRU**
Date:23.4.25

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

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

    int frames[capacity];
    for (int i = 0; i < capacity; i++)
        frames[i] = -1;

    int page_faults = 0;
    int index = 0;

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

```
        int found = 0;
        for (int j = 0; j < capacity; j++) {
            if (frames[j] == pages[i]) {
                found = 1;
                break;
            }
        }

        if (!found) {
            frames[index] = pages[i];
            index = (index + 1) % capacity;
            page_faults++;
        }

        for (int j = 0; j < capacity; j++) {
            printf("%d ", frames[j]);
        }
        printf("\n");
    }

    printf("Total Page Faults = %d\n", page_faults);

    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:

Thus,the lru has been successfully executed.

Ex. No.:11c) **Optimal**
Date:23.4.25

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

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

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

```
int page_faults = 0;

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

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

    if (!found) {
        int empty_index = -1;
        for (int j = 0; j < capacity; j++) {
            if (frames[j] == -1) {
                empty_index = j;
                break;
            }
        }

        if (empty_index != -1) {
            frames[empty_index] = pages[i];
        } else {
            int farthest = i + 1, index_to_replace = -1;

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

                if (next_use == -1) {
                    index_to_replace = j;
                    break;
                }

                if (next_use > farthest) {
                    farthest = next_use;
                    index_to_replace = j;
                } else if (index_to_replace == -1) {
                    index_to_replace = j;
                }
            }
```

```
                frames[index_to_replace] = pages[i];
            }

            page_faults++;
        }

        // Print frame contents
        for (int j = 0; j < capacity; j++) {
            printf("%d ", frames[j]);
        }
        printf("\n");
    }

    printf("Total Page Faults = %d\n", page_faults);
    return 0;
}
```

Output:

```
Enter number of frames: 3
Enter number of pages: 12
Enter reference string: 7 0 1 2 0 3 0 4 2 3 0 3

7 -1 -1
7 0 -1
7 0 1
2 0 1
2 0 1
2 0 3
2 0 3
4 0 3
4 0 2
4 0 3
4 0 3
4 0 3
Total Page Faults = 9
```

Result:

Thus the optimal algorithm has been successfully executed.