Ex. No: 11a Date: 21/3/25

FIFO PAGE REPLACEMENT

AIM:

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

ALGORITHM:

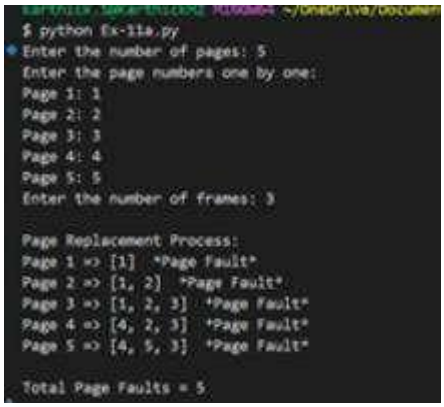
1. Declare the size with respect to page length
2. Check the need for replacement from the page to memory
3. Check the need for replacement from the old page to the new page in memory
4. Form a queue to hold all pages
5. Insert the page required memory into the queue
6. Check for bad replacement and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

PROGRAM:

```
def fifo_page_replacement(pages, frame_size): frames = []
page_faults = 0
front = 0
print("\nPage Replacement Process:") for page in pages:
if page not in frames:
if len(frames) < frame_size: frames.append(page)
else:
frames[front] = page
front = (front + 1) % frame_size page_faults += 1
print(f"Page {page} => {frames} *Page Fault*") else:
```

```
print(f"Page {page} => {frames}") print(f"\nTotal Page Faults = {page_faults}")

if _name_____== "_main_":

n = int(input("Enter the number of pages: ")) pages = []

print("Enter the page numbers one by one:") for i in range(n):

page = int(input(f"Page {i+1}: ")) pages.append(page)

frame_size = int(input("Enter the number of frames: ")) fifo_page_replacement(pages, frame_size)
```

OUTPUT:



RESULT:

The Fifo Page Replacement is Successfully Implemented using Python.

Ex. No: 11b Exp 11 b-LRU Date: 25/3/25

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:

```
#include <stdio.h>
int main() {
int pages[50], frames[10], counter[10];
int n, frameSize, i, j, k, flag, least, time = 0, faults = 0; printf("Enter the number of frames: ");
scanf("%d", &frameSize);
printf("Enter the number of pages: "); scanf("%d", &n);

printf("Enter the page reference string: "); for(i = 0; i < n; i++) {
scanf("%d", &pages[i]);
}
for(i = 0; i < frameSize; i++) { frames[i] = -1;
counter[i] = 0;
}
```

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

for(j = 0; j < frameSize; j++) { if(frames[j] == pages[i]) { counter[j] = ++time;
flag = 1; break;
}
}
if(flag == 0) {
int pos = -1, min = 9999;
for(j = 0; j < frameSize; j++) { if(frames[j] == -1) {
pos = j; break;
} else if(counter[j] < min) {
min = counter[j]; pos = j;
}
}

frames[pos] = pages[i]; counter[pos] = ++time; faults++;
}
printf("Frames after inserting %d: ", pages[i]); for(k = 0; k < frameSize; k++) {
if(frames[k] != -1)
printf("%d ", frames[k]); else
printf("- ");
}
printf("\n");
}

printf("\nTotal Page Faults: %d\n", faults); return 0;
}
```

OUTPUT:

```
$ bash lru_page.sh
Enter number of frames: 2
Enter number of pages: 1
Enter page reference string (space-separated): 3

Page Replacement Process:
Page 3 -> [ 3 - ] (Page Fault)

Total Page Faults: 1
lru_page.sh: line 106: bc: command not found
Hit Ratio: %
lru_page.sh: line 108: bc: command not found
Miss Ratio: %
```

RESULT:

The LRU Program is Successfully Implemented using C.

Ex. No: 11c Date: 25/3/25

Optimal

AIM:

To write a c program to implement the 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> #include <stdlib.h>

int isInFrame(int frame[], int count, int page) { for (int i = 0; i < count; i++)
if (frame[i] == page) return 1; return 0;
}
int predict(int pages[], int frame[], int n, int index, int count) { int farthest = index, res = -1;
for (int i = 0; i < count; i++) { int j;
for (j = index; j < n; j++) { if (frame[i] == pages[j]) {
if (j > farthest) { farthest = j; res = i;
}
break;
}
}
if (j == n) return i; // If page not found in future
```

```
}
return (res == -1) ? 0 : res;
}

int main() {
int n, frameCount, pageFaults = 0, filled = 0;
printf("Enter number of pages: "); scanf("%d", &n);
int* pages = malloc(n * sizeof(int));

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

printf("Enter number of frames: "); scanf("%d", &frameCount);
int* frame = malloc(frameCount * sizeof(int)); for (int i = 0; i < frameCount; i++)
frame[i] = -1;

for (int i = 0; i < n; i++) {
if (!isInFrame(frame, frameCount, pages[i])) { if (filled < frameCount)
frame[filled++] = pages[i]; else
frame[predict(pages, frame, n, i, frameCount)] = pages[i]; pageFaults++;
}
printf("Frame: ");
for (int j = 0; j < frameCount; j++)
frame[j] == -1 ? printf("- ") : printf("%d ", frame[j]); printf("\n");
}
printf("\nTotal Page Faults = %d\n", pageFaults); free(pages);
free(frame); return 0;
```

OUTPUT:

```
$ bash optimal_page.sh
Enter number of frames: 1
Enter number of pages: 1
Enter page reference string (space-separated): 1

Page Replacement Process:
Page 1 -> [ 1 ] (Page Fault)

Total Page Faults: 1
optimal_page.sh: line 98: bc: command not found
Hit Ratio: %
optimal_page.sh: line 100: bc: command not found
Miss Ratio: %
```

RESULT:

The Optimal page replacement Program is Successfully Implemented using C.

2116231801161