Ex. No.: 11 a
Date: 30.04.2024

FIFO PAGE REPLACEMENT

Aim:

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

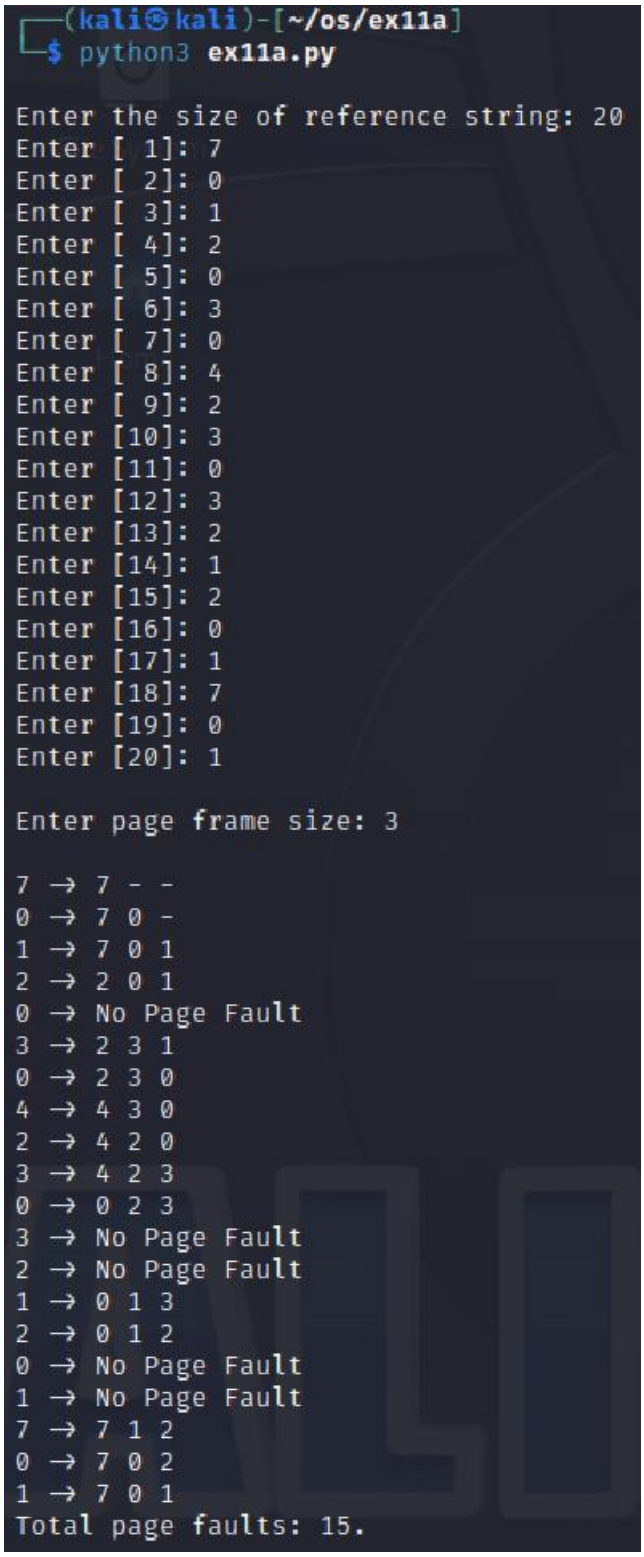
Algorithm:

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:

```
def fifo():
    global a, n, m
    f = -1
    page_faults = 0
    page = [-1] * m
    for i in range(n):
        flag = 0
        for j in range(m):
            if page[j] == a[i]:
                flag = 1
                break
        if flag == 0:
            f = (f + 1) % m
            page[f] = a[i]
            page_faults += 1
            print("\n%d ->" % a[i], end=" ")
            for j in range(m):
                if page[j] != -1: print(page[j], end=" ")
                else: print("-", end=" ")
        else: print("\n%d -> No Page Fault" % a[i], end=" ")
    print("\nTotal page faults: %d." % page_faults)
a = []
n = int(input("\nEnter the size of reference string: "))
for i in range(n):
    a.append(int(input("Enter [%2d]: " % (i + 1))))
m = int(input("\nEnter page frame size: "))
fifo()
```

Output:



Result:

The above program executed successfully and output got verified.

Ex. No.: 11 b
Date: 04.05.2024

LRU

Aim:

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

Algorithm:

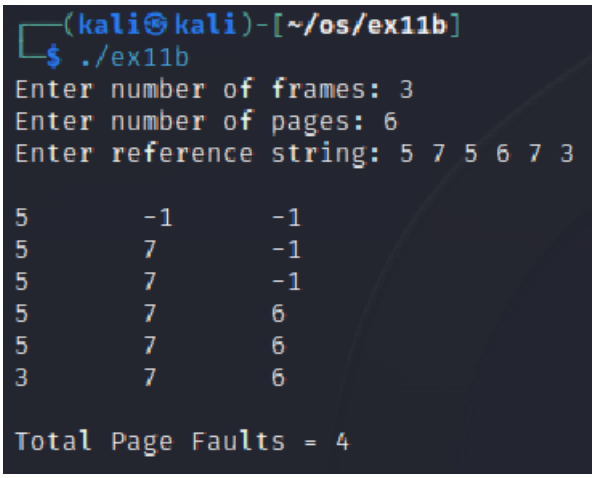
1: Start the process
2: Declare the size
3: Get the number of pages to be inserted
4: Get the value
5: Declare counter and stack
6: Select the least recently used page by counter value
7: Stack them according the selection.
8: Display the values
9: Stop the process

Program Code:

```
#include<stdio.h>

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

int main() {
   int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j,
pos, faults = 0;

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

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

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

```
        for(j = 0; j < no_of_frames; ++j) {
            if(frames[j] == pages[i]) {
                counter++;
                time[j] = counter;
                flag1 = flag2 = 1;
                break;
            }
        }
        if(flag1 == 0) {
            for(j = 0; j < no_of_frames; ++j) {
                if(frames[j] == -1) {
                    counter++;
                    faults++;
                    frames[j] = pages[i];
                    time[j] = counter;
                    flag2 = 1;
                    break;
                }
            }
        }
        if(flag2 == 0) {
            pos = findLRU(time, no_of_frames);
            counter++;
            faults++;
            frames[pos] = pages[i];
            time[pos] = counter;
        }
        printf("\n");
        for(j = 0; j < no_of_frames; ++j) {
            printf("%d\t", frames[j]);
        }
    }
    printf("\n\nTotal Page Faults = %d", faults);
    return 0;
}
```

Output:



Result:

The above program executed successfully and output got verified.

Ex. No.: 11 b
Date: 04.05.2024
LRU
Aim:

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

Program Code:

```
#include<stdio.h>

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

int main() {
   int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j,
pos, faults = 0;

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

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

   for(i = 0; i < no_of_pages; ++i) {
      flag1 = flag2 = 0;
      for(j = 0; j < no_of_frames; ++j) {
         if(frames[j] == pages[i]) {
            counter++;
            time[j] = counter;
            flag1 = flag2 = 1;
            break;
         }
      }
      if(flag1 == 0) {
         for(j = 0; j < no_of_frames; ++j) {
            if(frames[j] == -1) {
               counter++;
               faults++;
```

```
                frames[j] = pages[i];
                time[j] = counter;
                flag2 = 1;
                break;
            }
        }
    }
    if(flag2 == 0) {
        pos = findLRU(time, no_of_frames);
        counter++;
        faults++;
        frames[pos] = pages[i];
        time[pos] = counter;
    }
    printf("\n");
    for(j = 0; j < no_of_frames; ++j) {
        printf("%d\t", frames[j]);
    }
}
printf("\n\nTotal Page Faults = %d", faults);
return 0;
}
```

Output:

```
┌──(kali㉿kali)-[~/os/ex11b]
└─$ ./ex11b
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:

The above program executed successfully and output got verified.