Assignment 6

PART A :-

Page Replacement Algorithms implemented ->

1. First In First Out :-
-> In this a queue is made having as many number of blocks as there are page_frames. Each queue is initially initialised with the character 'X'. Then there is a pointer first_in which points the element added the most fomerly. Initially till the time all the positions in the queue are not filled, there is a pointer empty_pos which points to the empty position but when the queue fills completely the first_in and empty_pos points to the same place because the element which is in first should be replaced first.

2. Least Frequently Used :-
-> In this strategy a frequency_arr is initialised with all elements 0 in the beginning of the same size as the frame. Then whenever a page appears in the page string, then we will check for the page in the present frame. If it is already present then the frequency of it will be increased. And if we want to replace a page to add another one in the frame if it is full then we will traverse in the frequency array to find the index with the least frequency and then replace the page present in this index with the new one and then the frequency corresponding to it will be written as 0. If we have a tie between two pages, i.e., **if two pages have the least and same frequency then the page present at the lower index will be replaced.**

3. Least Recently Used :-
-> In this strategy a call_time array initialised with all -1, then as the pages appear we will store the time of apperance (iteration in which it appears) in the call_time array. Whenever a page already present in the queue appears in the refrence string then the call_time of it is updated to the new one. And for replacing we choose the page which has the lowest call time, i.e., which is the least recently called.

4. Clock Algorithm :-
-> In this a second chance array is formed in which all the blocks are initialised to 0 in the starting and since in this algorithm each block of the frame is chosen for replacement folowing the round robbinsion method, hence to implement this we also have a pointer named "pointer". Initilly the pointer is at 0 and then as the frames starts filling up the pointers value is increased each time a new page is added; and if the page in the ref_string is already present in the queue then the second chance bit of that index is changed to 1. And now if a page needs to be added and the queue is completely filled, then first we will try to replace the page at which the pointer is present, if the second chance bit of that page is 0 then it will be replaced else the second chance bit will change to 0 and the pointer would go for the very next position. This same process will repeat in circle till we find a page with second chance bit 0 and then we will replace it.

5. N-Chance Algorithm :-

-> It's implementation is the same as the clock algorithm but the only difference is that in this the entries of the chance array ranges from 0 to N-1. So, we may have to repeat the search for a page with no chance by repeated circular traversal and in each traversal the chance of every page traversed is reeuced by 1.

In this assignment to generate random reference string, I coded an ref_str_generator in python.

<u>SUBMITTED BY :-</u>

Nikita Bhamu,
2018CS50413.