In general  page replacement algorithms are are needed to decide which page needed to be replaced when new page comes in. If a new page is not present in memory, page fault occurs and according to given algorithm it OS replaces one of the existing pages with newly needed page.

1. FIrst algorithm I implemented was Optimal page replacement algorithm. This algorithm in short , pages which are not used for the longest duration of time in the future are replaced. So make prediction. In the program at first we initialize all elements of our frame array to -1. Then we check whether page is in frame already, if so we set flag to 1 which shows that page is available in frame and we will break the loop. Otherwise if flag is still 0,which means that  page is not available in frame, we will replace some page by required page. For it we first calculate distances to future occurence for every frame containing page by creating inner loop with variable temp which goes through page string elements starting from the next element and which will look for matches. If that page occurs in the future we will store this distance in the array distances[]. Then through different cases we will look for the best candidate index for page replacement in frame. Firstly we just take case when it is not occurred in the future, if so we will simply set this frame index to variable index. Secondly we take case when frame is not fully filled,then we set blank frame's index to index variable. in the last case we will take candidate that is having maximum occurence distance in  the future, for this we just compare distances and take maximum. Then finally we do page replacement and increment number of page faults and we will show that.

2. The second algorithm was LRU, in which page will be replaced which is least recently used. Here it is almost similar but it goes to past. Different thing in its implementation was when we create inner loop  we firstly take temp as i-1 and decreasingly will go untill temp>-1. Then steps are repeated and finally we will take candidate that is having maximum occurence distance in  the past also, for this we just compare distances and take maximum.

3. Then I implemented FIFO algorithm. It is the simplest page replacement algorithm, in which OS keeps track of all pages in the memory in a queue, and oldest page is in the front of the queue. When a page needs to be replaced page which in in the top of the queue will be selected for removal. In its program when our variable flag is still 0,which means that  page is not available in frame, we just use FIFO principle,which takes the oldest one, to replace that page by required page.

4. The last algorithm was The Second chance algorithm or Clock algorithm. It will be similar to our FIFO algo as long as a page already in memory is requested again. If so, in the program we will set usebit[] of that element to 1 meaning its reference bit is set to 1 and if the next time that page is selected to be replace, instead of replacing it, the reference bit will be set to zero, and the next candidate victim page is examined using the same way with the help of FIFO. So, here that selected page t was given second chance. And every time we replace elements in frame we increment  number of variable faults.