Name: Shawn Louis

SE COMPS

Roll No: 31

Experiment no 9

Aim : Write a program to implement page replacement policy - Optimal algorithm

Theory :

The Optimal Page Replacement Algorithm − This algorithm replaces the page that will not be used for the longest period of time. The moment the page fault occurs, some set of pages are in memory. One of these page will be referenced on the very next instruction.
The optimal page algorithm simply removes the page with the highest number of such instructions implying that it will be needed in the most distant future. this algorithm was introduced long back and is difficult to implement because it requires future knowledge of the program behavior. However it is possible to implement optimal page replacement on the second run by using the page reference information collected on the first run.

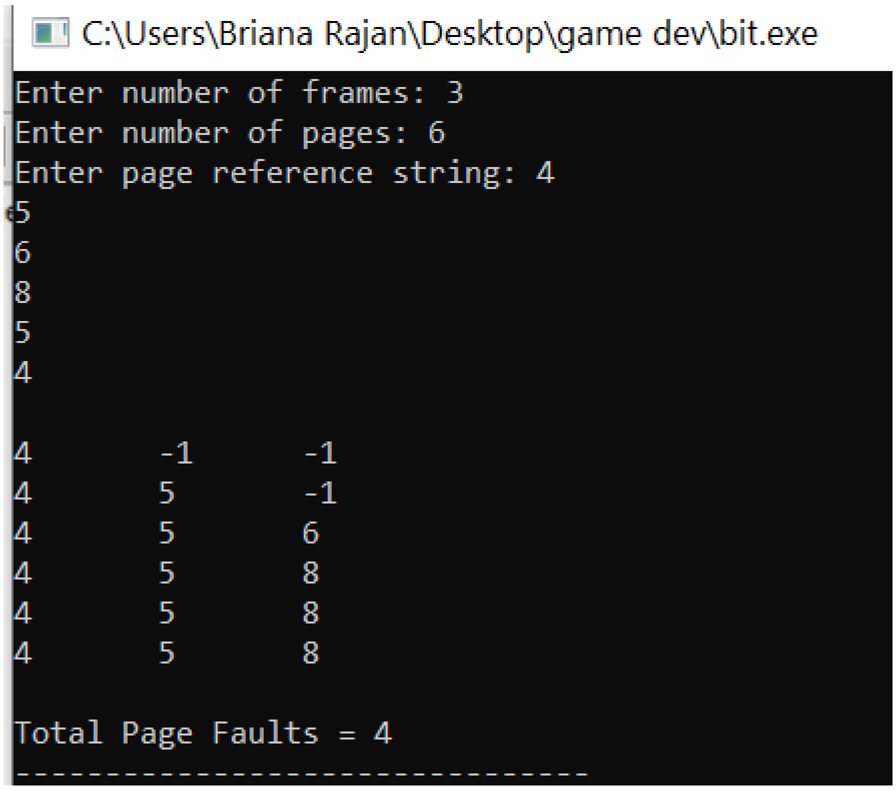
Output :

```
#include<stdio.h>

int main()
{
   int no_of_frames, no_of_pages, frames[10], pages[30], temp[10], flag1, flag2, flag3, i, j, k, pos, max, faults = 0;
   printf("Enter number of frames: ");
   scanf("%d", &no_of_frames);

   printf("Enter number of pages: ");
   scanf("%d", &no_of_pages);

   printf("Enter page 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]){
```

```
            flag1 = flag2 = 1;
            break;
        }
}

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

if(flag2 == 0){
    flag3 =0;

    for(j = 0; j < no_of_frames; ++j){
        temp[j] = -1;

        for(k = i + 1; k < no_of_pages; ++k){
            if(frames[j] == pages[k]){
                temp[j] = k;
                break;
            }
        }
    }

    for(j = 0; j < no_of_frames; ++j){
        if(temp[j] == -1){
            pos = j;
            flag3 = 1;
            break;
        }
    }

    if(flag3 ==0){
        max = temp[0];
        pos = 0;

        for(j = 1; j < no_of_frames; ++j){
            if(temp[j] > max){
                max = temp[j];
                pos = j;
            }
        }
    }

    frames[pos] = pages[i];
    faults++;
}

printf("\n");

for(j = 0; j < no_of_frames; ++j){
    printf("%d\t", frames[j]);
```

```
        }
    }

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

    return 0;
}
```



```
C:\Users\Briana Rajan\Desktop\game dev\bit.exe
Enter number of frames: 3
Enter number of pages: 6
Enter page reference string: 4
5
6
8
5
4

4       -1      -1
4       5       -1
4       5       6
4       5       8
4       5       8
4       5       8

Total Page Faults = 4
--------------------------------
```

Conclusion :Thus, page

replacement algorithm was

implemented and total

number of page faults were

also found. The advantages

and limitations of the

algorithm were also pondered

upon.