EX.NO:8 PAGE REPLACEMENT ALGORITHMS

DATE:

AIM:

To write a C program for implementation of FIFO, LRU, and Optimal page replacement algorithm.

FIFO PAGE REPLACEMENT:

ALGORITHM:

STEP 1: Start.

STEP 2: Declare the necessary variables.

STEP 3: Enter the number of frames.

STEP 4: Enter the reference string end with zero.

STEP 5: FIFO page replacement selects the page that has been in memory the longest time and when the

page must be replaced the oldest page is chosen.

STEP 6: When a page is brought into memory, it is inserted at the tail of the queue.

STEP 7: Initially all the three frames are empty.

STEP 8: The page fault range increases as the no of allocated frames also increases.

STEP 9: Print the total number of page faults.

STEP 10: Stop

CODE:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
char str[20];
 printf("Enter a reference string:");
 gets(str);
 int  len=strlen(str);
 int pageFaults = 0;
 int frames = 3;
 int m, n, s, pages,i;
```

```
 int incomingStream[20];
for(i=0;i<len;i++){
incomingStream[i]=str[i]-48;
}
 pages = sizeof(incomingStream)/sizeof(incomingStream[0]);
 printf(" Incoming \t\t Frame 1 \t\t Frame 2 \t\t Frame 3 ");
 int temp[ frames ];
 for(m = 0; m < frames; m++)
 {
 temp[m] = -1;
 }
 for(m = 0; m < pages; m++)
 {
 s = 0;
 for(n = 0; n < frames; n++)
 {
 if(incomingStream[m] == temp[n])
 {
 s++;
 pageFaults--;
 }
 }
 pageFaults++;
 if((pageFaults <= frames) && (s == 0))
 {
 temp[m] = incomingStream[m];
 }
 else if(s == 0)
 {
 temp[(pageFaults - 1) % frames] = incomingStream[m];
```

```
 }
 printf("\n");
 printf("%d\t\t\t",incomingStream[m]);
 for(n = 0; n < frames; n++)
 {
 if(temp[n] != -1)
 printf(" %d\t\t\t", temp[n]);
 else
 printf(" - \t\t\t");
 } }
 printf("\nTotal Page Faults:\t%d\n", pageFaults);
 return 0;
}
```

OUTPUT:

```
Enter  a reference string:7012030423032120170
 Incoming               Frame 1                Frame 2                Frame 3
7                       7                      -                      -
0                       7                      0                      -
1                       7                      0                      1
2                       2                      0                      1
0                       2                      0                      1
3                       2                      3                      1
0                       2                      3                      0
4                       4                      3                      0
2                       4                      2                      0
3                       4                      2                      3
0                       0                      2                      3
3                       0                      2                      3
2                       0                      2                      3
1                       0                      1                      3
2                       0                      1                      2
0                       0                      1                      2
1                       0                      1                      2
7                       7                      1                      2
0                       7                      0                      2
0                       7                      0                      2
Total Page Faults:      14
```

LRU PAGE REPLACEMENT:

ALGORITHM:

STEP 1: Start.

STEP 2: Declare the size

STEP 3: Get the number of pages to be inserted

STEP 4: Get the value

STEP 5: Declare counter and stack

STEP 6: Select the least recently used page by counter value

STEP 7: Stack them according the selection.

STEP 8: Display the values

STEP 9: Stop.

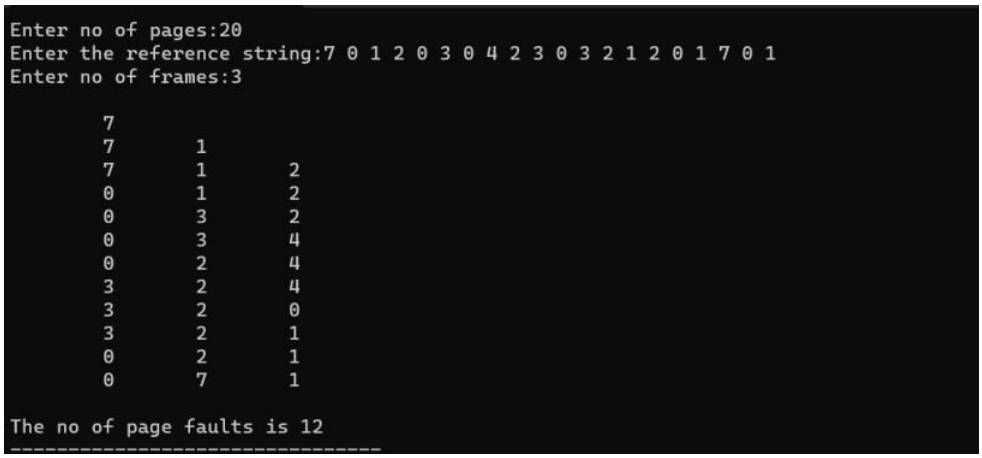
CODE:

```
#include<stdio.h>
main()
{
int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];
printf("Enter no of pages:");
scanf("%d",&n);
printf("Enter the reference string:");
for(i=0;i<n;i++)
scanf("%d",&p[i]);
printf("Enter no of frames:");
scanf("%d",&f);
q[k]=p[k];
printf("\n\t%d\n",q[k]);
c++;
k++;
for(i=1;i<n;i++)
 {
 c1=0;
 for(j=0;j<f;j++)
 {
 if(p[i]!=q[j])
 c1++;
 }
 if(c1==f)
 { c++;
 if(k<f)
 { q[k]=p[i];
 k++;
```

```
for(j=0;j<k;j++)
printf("\t%d",q[j]);
printf("\n");
 }
else {
 for(r=0;r<f;r++)
 {
 c2[r]=0;
 for(j=i-1;j<n;j--)
 {
 if(q[r]!=p[j])
 c2[r]++;
else
break;
 } }
 for(r=0;r<f;r++)
 b[r]=c2[r];
 for(r=0;r<f;r++)
 { for(j=r;j<f;j++)
 {if(b[r]<b[j])
 {t=b[r];
 b[r]=b[j];
 b[j]=t;
 } }
 }
 for(r=0;r<f;r++)
 { if(c2[r]==b[0])
 q[r]=p[i];
 printf("\t%d",q[r]);
 }
 printf("\n");
 } }
```

```
printf("\nThe no of page faults is %d",c);
}
```

OUTPUT:



```
Enter no of pages:20
Enter the reference string:7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter no of frames:3

        7
        7       1
        7       1       2
        0       1       2
        0       3       2
        0       3       4
        0       2       4
        3       2       4
        3       2       0
        3       2       1
        0       2       1
        0       7       1

The no of page faults is 12
--------------------------------
```

OPTIMAL REPLACEMENT:

ALGORITHM:

STEP 1: Start.

STEP 2: Declare the size

STEP 3: Get the number of pages to be inserted

STEP 4: Get the value

STEP 5: Declare counter and stack

STEP 6: Select the least frequently used page by counter value

STEP 7: Stack them according the selection.

STEP 8: Display the values

STEP 9: Stop.

CODE:

```
#include <stdio.h>
#include<conio.h>
#include<string.h>
int search(int key, int frame_items[], int frame_occupied)
{
 for (int i = 0; i < frame_occupied; i++)
 if (frame_items[i] == key)
 return 1;
 return 0;
```

```
}
void printOuterStructure(int max_frames){
 printf("Incoming ");
 for(int i = 0; i < max_frames; i++)
 printf("\tFrame%d \t", i+1);
}
void printCurrFrames(int item, int frame_items[], int frame_occupied, int max_frames){
 printf("\n%d \t\t", item);
 for(int i = 0; i < max_frames; i++){
 if(i < frame_occupied)
 printf(" %d \t\t", frame_items[i]);
 else
 printf(" - \t\t");
 }
}
int predict(int ref_str[], int frame_items[], int refStrLen, int index, int frame_occupied)
{
 int result = -1, farthest = index;
 for (int i = 0; i < frame_occupied; i++) {
 int j;
for (j = index; j < refStrLen; j++)
 {
 if (frame_items[i] == ref_str[j])
 {
 if (j > farthest) {
 farthest = j;
 result = i;
 }
 break;
 }
 }
 if (j == refStrLen)
```

```
 return i;
 }
return (result == -1) ? 0 : result;
}
void optimalPage(int ref_str[], int refStrLen, int frame_items[], int max_frames)
{
 int frame_occupied = 0;
 printOuterStructure(max_frames);
 int hits = 0;
 for (int i = 0; i < refStrLen; i++) {
 if (search(ref_str[i], frame_items, frame_occupied)) {
 hits++;
 printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
 continue;
 }
 if (frame_occupied < max_frames){
 frame_items[frame_occupied] = ref_str[i];
 frame_occupied++;
 printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
 }
 else {
 int pos = predict(ref_str, frame_items, refStrLen, i + 1, frame_occupied);
 frame_items[pos] = ref_str[i];
 printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
 }
 }
 printf("\nNo. of Page Faults: %d", refStrLen - hits);
}
int main()
{
int i;
 char str[20];
```

```
 printf("Enter a reference string:");
 gets(str);
int len=strlen(str);
int ref_str[20];
for(i=0;i<len;i++){
ref_str[i]=str[i]-48;
}
 int refStrLen = sizeof(ref_str) / sizeof(ref_str[0]);
 int max_frames = 3;
 int frame_items[max_frames];
 optimalPage(ref_str, refStrLen, frame_items, max_frames);
 return 0;
}
```

OUTPUT:

```
Enter  a reference string:70120304230321201701
Incoming        Frame1          Frame2          Frame3
7                 7               -               -
0                 7               0               -
1                 7               0               1
2                 2               0               1
0                 2               0               1
3                 2               0               3
0                 2               0               3
4                 2               4               3
2                 2               4               3
3                 2               4               3
0                 2               0               3
3                 2               0               3
2                 2               0               3
1                 2               0               1
2                 2               0               1
0                 2               0               1
1                 2               0               1
7                 7               0               1
0                 7               0               1
1                 7               0               1
No. of Page Faults: 9
--------------------------------
Process exited after 23.67 seconds with return value 0
Press any key to continue . . .
```

Observation (20)	
Record(5)	
Total (25)	
Intial	

RESULT:

Thus the c program for implementing Paging technique for memory management are executed successfully and the outputs are verified