Write a C program for FIFO P.R, CRV P.R and Optimal P.R.

FIFO

```
#include <stdio.h>
#define Max_frames 3 // number of mem. frames
int frames [Max_frames]; //array to hold mem. frames.
int frame pointer = 0; //Pointer to the current frame being replaced
int page fault count = 0; // counter for page faults
// function to check if a page is already in memory.
int isPageInMemory (int page){
for (int i=0; i< Max_frames, i++)
{
if (frames [i] == page) { // checks if given page is already
    return 1;              present in the memo. frames.
}
}
return 0;  // Page not in memory.
}
void display frame () // display the contents of mem. frames.
{
for (int i=0; i< Max_frames; i++){
~~if (frames [i] == page) {~~ if (frames [i] != -1) {
Printf ("%d", frame [i]);
}else {
 Printf ("-");
}
}
Printf ("\n");
}
void fifo (int pages [], int page count){
for (int i=0; i< page count; i++) {
  Printf (" for page %d ", pages [i]);
if (! is Page in memory (pages [i]))
 {
frames [frame pointer] = pages [i];
```

```
framePointer = (framePointer + 1) % Max_frames;
pageFaultCount++;
// Printf("Page fault occured.");
} else {
Printf("No page fault.");
}
displayFrames();
}
Printf("\n Total Page faults: %d \n", PageFaultCount);
}
int main(){
int pageCount;
Printf("Enter the number of pages: ");
scanf("%d", &pageCount);
int pages [pageCount];
Printf("Enter the pages: ");
for(int i=0; i< pageCount; i++) {
scanf("%d", &pages[i]);
}
for(int i=0; i< Max_frames ; i++) {
frames [i] = -1;
}
fifo(pages, pageCount);
return 0;
}
```

Output

```
Enter no. of pages : 5
Enter the pages : 14 25 34 16 17
Enter no. of frames : 4
For 14 : 14 -
For 25 : 14 25
For 34 : 14 25 34
For 16 : 14 25 34 16
For 17 : 14 25 34 16 17
Total no. of page faults = 5
```

LRU

```
#include <stdio.h>
void displayframes (int fr[], int fn){
for(int i=0; i<fn; i++) {
if (fr[i] != -1){
Printf("%d", fr[i]);
}else {
Printf("__");
}
}
Printf("\n");
}
Int isPage inmemory (int page, int fn, int frames[fn])
{
for(int i=0; i< fn; i++){
if (frames [i] == pages){
return 1;
}
}
return 0;
}
void lruPage (int pg[], int pn, int fn)
{
int fr[fn];
for (int i=0; i< fn; i++)
fr[i] = -1;
int hit=0;
for (int i=0; i< pn; i++) {
if !is page in memory (pg[i], fn, fr)
) {
hit ++;
Printf(" Page %d: Hit: ", pg[i]);
}
else
```

```
int emptyFrame = -1;
for(int j=0; j<fn; j++)
{
if(fr[j] == -1){
fr[j] = pg[i];
emptyFrame = j;
break;
}
}
if(emptyFrame != -1)
Printf("Page %d: Miss:", pg[i]);
else{
int minCounter = pn+1;
replaceIndex = -1;
for(int j=0; j<fn; j++)
{
int k;
for(k=i-1; k>=0; k--){
if(fr[j] == pg[k]){
if(k<minCounter){
minCounter = k;
replaceIndex = j;
}
break;
}
}
if(k == -1){
replaceIndex = j;
break;}
}
fr[replaceIndex] = pg[i];
Printf("Page %d: Miss:", pg[i]);
}
}
```

```
displayframes(fr, fn);
}
Printf("\nNo. of hits = %d\n", hit);
Printf("No. of misses = %d\n", pn-hit);
}
int main(){
int pn;
Printf("Enter no. of pages:");
Scanf("%d", &pn);
inpg[pn];
Printf("Enter the pages:");
for(int i=0; i<pn; i++){
 scanf("%d", &pg[i]);
}
int fn=3;
lrupage(pg, pn, fn);
return 0;
}
```

Output

Enter the no. of pages: 5
Enter the page sequence:
14 25 34 16 17.
Enter no. of frames: 4
For 14: 14
For 25: 14 25
For :34: 14 25 34
For 16: 14 25 34 16
For 17: ~~14~~ 17 25 34 16
Total no. of page faute: 5.

optimal

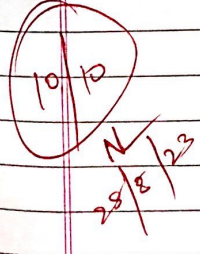
```
#include <stdio.h>
void displayFrames(int fr[], int fn){
for(int i=0; i<fn; i++){
if(fr[i]!=-1){
printf("%d", fr[i]);
}else{
printf("-");
}
}
printf("\n");
}
void optimalPage(int pg[], int pn, int fn)
{
int fr[fn];
for(int i=0; i<fn; i++){
fr[i]=-1;
}
int hit=0;
for(int i=0; i<pn; i++){
int found=0;
for(int j=0; j<fn; j++){
if(fr[j]==pg[i]){
hit++;
found=1;
break;
}
}
if(found){
printf("Page %d: Hit:", pg[i]);}
else{
int emptyFrame=-1;
for(int j=0; j<fn; j++){
if(fr[j]==-1){
fr[j]=pg[i];
```

```
        emptyframe = j;
        break;
    }
  }
  if (emptyframe != -1) {
  Printf("Page %d: Miss:", pg[i], pg[i]);
  }
  else {
  int farthest = -1, replaceIndex = -1;
  for(int j=0; j<fn; j++) {
  int k;
  for(k = i+1; k<pn; k++) {
  if(fr[j] == pg[k]) {
  if(k > farthest) {
   farthest = k;
  replaceIndex = j; }
  break;
   }
  }
  if(k == pn) {
  replaceIndex = j;
  break;
   }
  }
  fr[replaceIndex] = pg[i];
  Printf("Page %d: Miss:", pg[i], fr[replaceIndex]);
  }
  }
  display frames(fr, fn);
  }
  Printf("\n No. of hits = %d\n", hit);
  Printf("\n NO. of misses = %d\n", pn-hit);
  }
  int main() {
```

```
int pn;
Printf("Enter no. of pages:");
scanf("%d", &pn);
int pg[pn];
Printf("Enter the pages:");
for(int i=0; i<pn; i++){
scanf("%d", &pg[i]);
}
int fn=3;
optimal page(pg, pn, fn);
return 0;
}
```

Output

```
Enter no. of pages: 5
Enter page sequence: 14 25 34 16 17
Enter no. of frames: 4
  For 14: 14
  For 25: 25
  For 34: 34
  For 16: 16
  For 17: 17
Total no. of page faults: 5.
```



Scanned with CamScanner

```
PS D:\VS Code\OS> cd "d:\VS Code\OS\" ; if ($?) { gcc PR.c -o PR } ; if ($?) { .\PR }
Enter Size of memory:
3
Enter number of process in queue:
6
Enter 6 process
7 4 10 4 2 1

FIFO:
Memory: 7 0 0
Memory: 7 4 0
Memory: 7 4 10
Memory: 7 4 10
Memory: 1 4 10
LRU:
Memory: 7 0 0
Memory: 7 4 0
Memory: 7 4 10
Memory: 7 4 10
Memory: 7 4 1

Optimal:
Memory: 7 0 0
Memory: 7 4 0
Memory: 7 4 10
Memory: 7 4 10
Memory: 1 4 10
PS D:\VS Code\OS>
```