Theory for Practical Exam

📌 What is Page Replacement?

When a process is executed, not all of its pages can fit into RAM. So operating systems use **page replacement algorithms** to decide **which page to remove from memory** when a new page needs to be loaded.

📌 FIFO (First In, First Out) Page Replacement Algorithm

- FIFO is the **simplest page replacement algorithm**
- It replaces the **oldest page** in memory (the one that came first)
- Works like a **queue** → first inserted page is removed first
- **Disadvantage:** Doesn't consider page usage frequency → may replace a frequently used page (Belady's anomaly)

📌 Terms Used

Term	Meaning
Page	Logical chunk of memory requested by process
Frame	Physical block in RAM where a page is stored
Page Hit	Requested page is already in memory
Page Fault	Requested page is **not** in memory → must be loaded

---

📌 Example:

Frames = 3
Reference String: 7 0 1 2 0 3 0 4

Page	Frames	Status
7	7 - -	Fault
0	7 0 -	Fault
1	7 0 1	Fault
2	2 0 1	Fault (7 removed)
0	2 0 1	Hit
3	2 3 1	Fault (0 removed)
0	2 3 0	Fault (1 removed)

Page	Frames	Status
4	4 3 0	Fault (2 removed)

- **Total Faults** = 7
- **Total Hits** = 1

✅ **Code Explanation (Line by Line)**

#include <stdio.h>

→ Standard input/output library

int main() {

→ Program starts

int frames, pages[100], frame[10], n, i, j, f = 0;

int faults = 0, hits = 0;

- frames → Number of memory frames
- pages[] → Holds reference string
- frame[] → Stores current pages inside memory
- f → Pointer used for FIFO replacement (circular index)
- faults, hits → Counters

printf("Enter number of frames: ");

scanf("%d", &frames);

→ User inputs number of frames

printf("Enter number of pages: ");

scanf("%d", &n);

→ User inputs size of reference string

printf("Enter page reference string:\n");

for(i = 0; i < n; i++)

  scanf("%d", &pages[i]);

→ Reads all pages to be accessed

for(i = 0; i < frames; i++)

  frame[i] = -1;

→ Initialize all frames as empty (-1)

🔁 Main Loop – FIFO Logic

```
for(i = 0; i < n; i++) {
   int hit = 0;
```

→ For each page request, assume not hit

```
for(j = 0; j < frames; j++)
   if(frame[j] == pages[i])
      hit = 1;
```

→ Check if current page already in memory (Hit)

```
if(!hit) {   // Page Fault
   frame[f] = pages[i];
   f = (f + 1) % frames;
   faults++;
} else {
   hits++;
}
```

✓ If **fault:**

- Replace page at frame index f
- Move pointer circularly using (f+1)%frames

✓ If **hit:** just increase hits counter

🖳 Output Display Section

```
printf("%d\t", pages[i]);
for(j = 0; j < frames; j++)
   printf("%s ", frame[j] == -1 ? "-" : (char [3]){frame[j]+'0',0});
```

→ Displays frames (if empty prints -, else page number)

```
printf("\t\t%s", hit ? "Hit" : "Page Fault");
if(hit) printf(" (%d)", pages[i]);
printf("\n");
```

→ Prints whether **Hit** or **Page Fault**

✅ Final Output

```
printf("\nTotal Page Faults: %d", faults);

printf("\nTotal Page Hits: %d\n", hits);
```