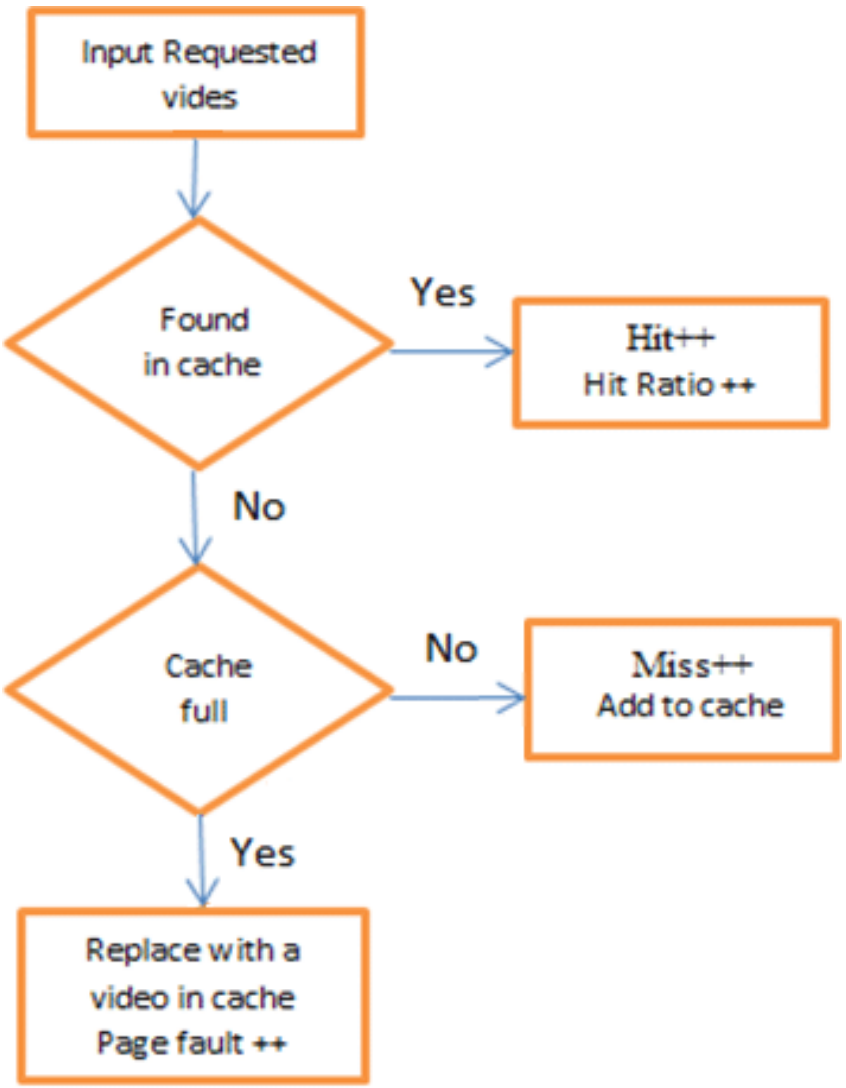
Name:	Sujal Sandeep Dingankar
UID:	2024301005
Experiment No.	9

AIM:	To Study Page Replacement Algorithms.
Program 1	
PROBLEM STATEMENT:	To perform First In First Out (FIFO) Page Replacement Algorithm.
FLOWCHART:	

ALGORITHM:	1. Input: - Number of frames (frame size) - Number of pages (reference string length) - Page reference sequence 2. Initialize: - Create frame array of size 'frames' with -1 - Set frameIndex = 0 (tracks next frame to replace) - Set pageFaults = 0, pageHits = 0 3. For each page in reference sequence: a. Search for page in frames - If found: increment pageHits - If not found: * Place page in frame[frameIndex] * Increment frameIndex (circular using modulo) * Increment pageFaults b. Display current frame status - Show all frame contents - Indicate if it was a hit or fault 4. Output final statistics: - Total page faults - Total hits

PROGRAM:

```
#include <stdio.h>
int main() {
   int frames, pages, pageFaults = 0,
   pageHits = 0;
   int frameIndex = 0;

   printf("Enter the number of frames: ");
   scanf("%d", &frames);

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

   int pageSequence[pages], frame[frames];

   printf("Enter the page sequence : ");
   for (int i = 0; i < pages; i++) {
       scanf("%d", &pageSequence[i]);
   }

   for (int i = 0; i < frames; i++) {
       frame[i] = -1;
   }

   for (int i = 0; i < pages; i++) {
       int page = pageSequence[i];
       int found = 0;
  for (int j = 0; j < frames; j++) {

                if (frame[j] ==
                page) {found = 1;
                pageHits++;
                break;
            }
       }
  printf("Frame status after page %d: ",
page);

       if (!found) {
           frame[frameIndex] = page;
  frameIndex = (frameIndex + 1) % frames;
           pageFaults++;

        for (int k = 0; k < frames;k++) {
           if (frame[k] != -1)
```

```
                printf("%d ", frame[k]);
                else
                    printf("- ");
            }
            printf("Fault\n");
        } else {
         for (int k = 0; k < frames; k++) {
                if(frame[k]!=-1)
               printf("%d ", frame[k]);
                else
                    printf("- ");
            }
            printf("Hit\n");
        }
    }
      printf("\nTotal Page Faults: %d\n",
   pageFaults);
    printf("Total Hits: %d\n", pageHits);
    return 0;
}
```

RESULT:

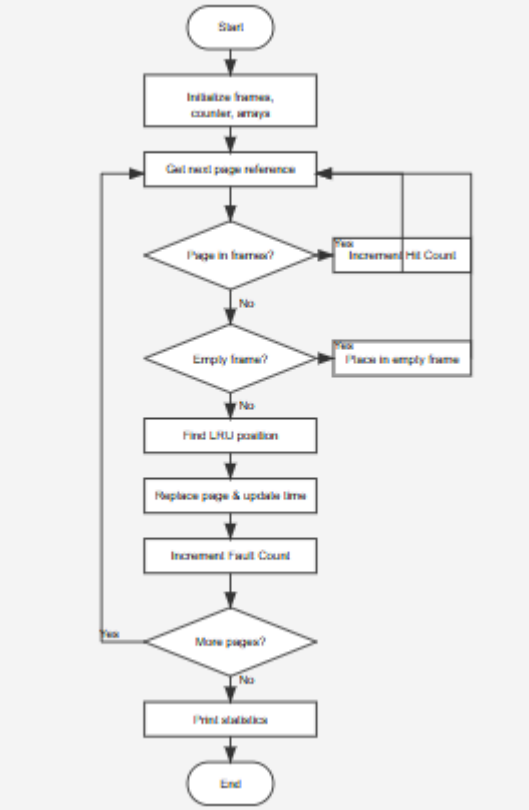
```
/tmp/VRShAXJ30r.o
Enter the number of frames: 3
Enter the number of pages: 20
Enter the page sequence : 1 2 3 2 4 5 6 4 7 7 3 4 6 6 4 5 6 8 2 3
Frame status after page 1: 1 - - Fault
Frame status after page 2: 1 2 - Fault
Frame status after page 3: 1 2 3 Fault
Frame status after page 2: 1 2 3 Hit
Frame status after page 4: 4 2 3 Fault
Frame status after page 5: 4 5 3 Fault
Frame status after page 6: 4 5 6 Fault
Frame status after page 4: 4 5 6 Hit
Frame status after page 7: 7 5 6 Fault
Frame status after page 7: 7 5 6 Hit
Frame status after page 3: 7 3 6 Fault
Frame status after page 4: 7 3 4 Fault
Frame status after page 6: 6 3 4 Fault
Frame status after page 6: 6 3 4 Hit
Frame status after page 4: 6 3 4 Hit
Frame status after page 5: 6 5 4 Fault
Frame status after page 6: 6 5 4 Hit
Frame status after page 8: 6 5 8 Fault
Frame status after page 2: 2 5 8 Fault
Frame status after page 3: 2 3 8 Fault

Total Page Faults: 14
Total Hits: 6


=== Code Execution Successful ===
```

Program 2

PROBLEM STATEMENT:	To perform Least Recently Used (LRU) Page Replacement Algorithm.

FLOWCHART:	
ALGORITHM:	1. Initialize: - Get number of frames and pages - Create arrays: frame[], time[], pageSequence[] - Set counter = 0, pageFaults = 0, pageHits = 0 - Fill frame[] with -1 (empty) 2. For each page in reference string: a) Check if page exists in frames: If found (Hit): - Increment pageHits - Increment counter - Update time[position] = counter b) If not found (Fault): If empty frame exists: - Place in next available frame (pos = i % frames) Else: - Find LRU position using findLRU() Then: - Place page in chosen position - Increment counter

	- Update time[position] = counter - Increment pageFaults c) Display frame status after each operation - Show current frame contents - Indicate Hit or Fault 3. Output final statistics: - Total page faults - Total hits

PROGRAM:	#include <stdio.h>
	int findLRU(int time[], int frames) {
	int min = time[0], pos = 0;
	for (int i = 1; i < frames; i++) {
	if (time[i] < min) {
	min = time[i];
	pos = i;
	}
	}
	return pos;
	}
	int main() {
	int frames, pages, pageFaults = 0, pageHits = 0;
	int counter = 0;
	
	printf("Enter the number of frames: ");
	scanf("%d", &frames);
	
	printf("Enter the number of pages: ");
	scanf("%d", &pages);
	
	int pageSequence[pages], frame[frames],
	time[frames];
	printf("Enter the page sequence : ");
	for (int i = 0; i < pages; i++) {
	scanf("%d", &pageSequence[i]);
	}
	
	for (int i = 0; i < frames; i++) {
	frame[i] = -1;
	}

```
for (int i = 0; i < pages; i++) {
    int page = pageSequence[i];
    int found = 0;

    for (int j = 0; j < frames; j++) {
        if (frame[j] == page) {
            found = 1;
            pageHits++;
            counter++;
            time[j] = counter;
            break;
        }
    }
 printf("Frame status after page %d: ", page);

    if (!found) {
        int pos;
       if (frame[0] == -1){
             pos = i % frames;
```

```
            } else {
                pos = findLRU(time, frames);
            }
            frame[pos] = page;
            counter++;
            time[pos] = counter;
            pageFaults++;

            for (int k = 0; k < frames; k++) {
                if (frame[k] != -1)
                    printf("%d ", frame[k]);
                else
                    printf("- ");
            }
            printf("Fault\n");
        } else {
            for (int k = 0; k < frames; k++) {
                if (frame[k] != -1)
                    printf("%d ", frame[k]);
                else
                    printf("- ");
            }
            printf("Hit\n");
        }
    }

    printf("\nTotal Page Faults: %d\n",
pageFaults);
    printf("Total Hits: %d\n", pageHits);
    return 0;
}
```

RESULT:

```
/tmp/H4iSYd69qs.o
Enter the number of frames: 3
Enter the number of pages: 20
Enter the page sequence : 1 2 3 2 4 5 6 4 7 7 3 4 6 6 4 5 6 8 2 3
Frame status after page 1: 1 - - Fault
Frame status after page 2: 1 2 - Fault
Frame status after page 3: 1 2 3 Fault
Frame status after page 2: 1 2 3 Hit
Frame status after page 4: 4 2 3 Fault
Frame status after page 5: 4 2 5 Fault
Frame status after page 6: 4 6 5 Fault
Frame status after page 4: 4 6 5 Hit
Frame status after page 7: 4 6 7 Fault
Frame status after page 7: 4 6 7 Hit
Frame status after page 3: 4 3 7 Fault
Frame status after page 4: 4 3 7 Hit
Frame status after page 6: 4 3 6 Fault
Frame status after page 6: 4 3 6 Hit
Frame status after page 4: 4 3 6 Hit
Frame status after page 5: 4 5 6 Fault
Frame status after page 6: 4 5 6 Hit
Frame status after page 8: 8 5 6 Fault
Frame status after page 2: 8 2 6 Fault
Frame status after page 3: 8 2 3 Fault

Total Page Faults: 13
Total Hits: 7


=== Code Execution Successful ===
```

CONCLUSION:	In this experiment, I implemented the FIFO, LRU, and Optimal page replacement algorithms to manage memory page requests. The FIFO algorithm replaces the oldest page, making it simple but prone to high page faults. The LRU algorithm reduces page faults by replacing the least recently used page, adapting better to usage patterns. The Optimal algorithm achieves the lowest page faults by replacing the page that will not be used for the longest period, though it requires future knowledge.