Name :- Khan Faisal
Roll no :- 197

Batch :- C12

SAP ID :- 60004240019
Subj :- OS

EXPERIMENT. 9

* Aim :- Implement various page replacement policies. (Optimal, LRU, FIFO, & LFU)

* Theory :-

- Page replacement is needed in the OS that use virtual memory using Demand paging. As we know in demand paging, only a set of pages of a process is loaded into the memory. This is done so that we can have more processes in the memory at the same time.

- Page replacement algos :-

1] Optimal
2] LRU
3] FIFO
4] LFU

1] Optimal :-

	1	2	3	4	2	1	5	3	2	4	6
f4				4	4	4	4	4	4	~~4~~	6
f3			3	3	3	3	3	3	3	3	3
f2		2	2	2	2	2	2	2	2	2	2
f1	1	1	1	1	1	~~1~~	5	5	5	5	5
	*	*	*	*	h	h	*	h	h	h	*

PF = 6 Ratio = 54.5 %
PH = 5 Ratio = 45.5 %

FOR EDUCATIONAL USE

2] LRU [Least Recently Used]:

	1	2	3	4	2	1	5	3	2	4	6
f_4				4	4	4	~~4~~	3	3	3	3
f_3			3	3	3	~~3~~	5	5	5	~~5~~	6
f_2		2	2	2	2	2	2	2	2	2	2
f_1	1	1	1	1	1	1	1	1	~~1~~	4	4
	*	*	*	*	h	h	*	*	h	*	*

PF = 8 Ratio = 72.7%
PH = 3 Ratio = 27.3%

3] FIFO [First In First out]:-

	1	2	3	4	2	1	5	3	2	4	6
f_4				4	4	4	4	4	4	4	4
f_3			3	3	3	3	3	3	3	3	3
f_2		2	2	2	2	2	2	2	~~2~~	~~2~~	6
f_1	1	1	1	1	1	~~1~~	5	5	5	5	5
	*	*	*	*	h	h	*	h	h	h	*

PF = 6 Ratio = 54.5%
PH = 5 Ratio = 45.5%

4] LFU [Least Frequently Used]:-

	1	2	x 3	x 4	2	1	5	3	2	4	6
f4				4	4	4	~~X~~	3	3	~~3~~	6
f3			3	3	3	~~3~~	5	5	~~5~~	4	4
f2		2	2	2	2	2	2	2	2	2	2
f1	1	1	1	1	1	1	1	1	1	1	1
	*	*	*	*	h	h	*	*	h		

* Frequencies :-

1 = ~~0~~ ~~1~~ 2　　　2 = ~~0~~ ~~1~~ ~~2~~ 3　　　3 = ~~0~~ ~~1~~ ~~0~~ ~~1~~ 0

4 = ~~0~~ ~~1~~ ~~0~~ 1　　　5 = ~~0~~ ~~1~~ 0　　　6 = ~~0~~ 1

PF = 8　　　Ratio = 72.7%
PH = 3　　　Ratio = 27.2%

* Conclusion :-

Thus, I have understood the concept of Replacement algos & their implementation in c program. I have successfully performed this experiment.

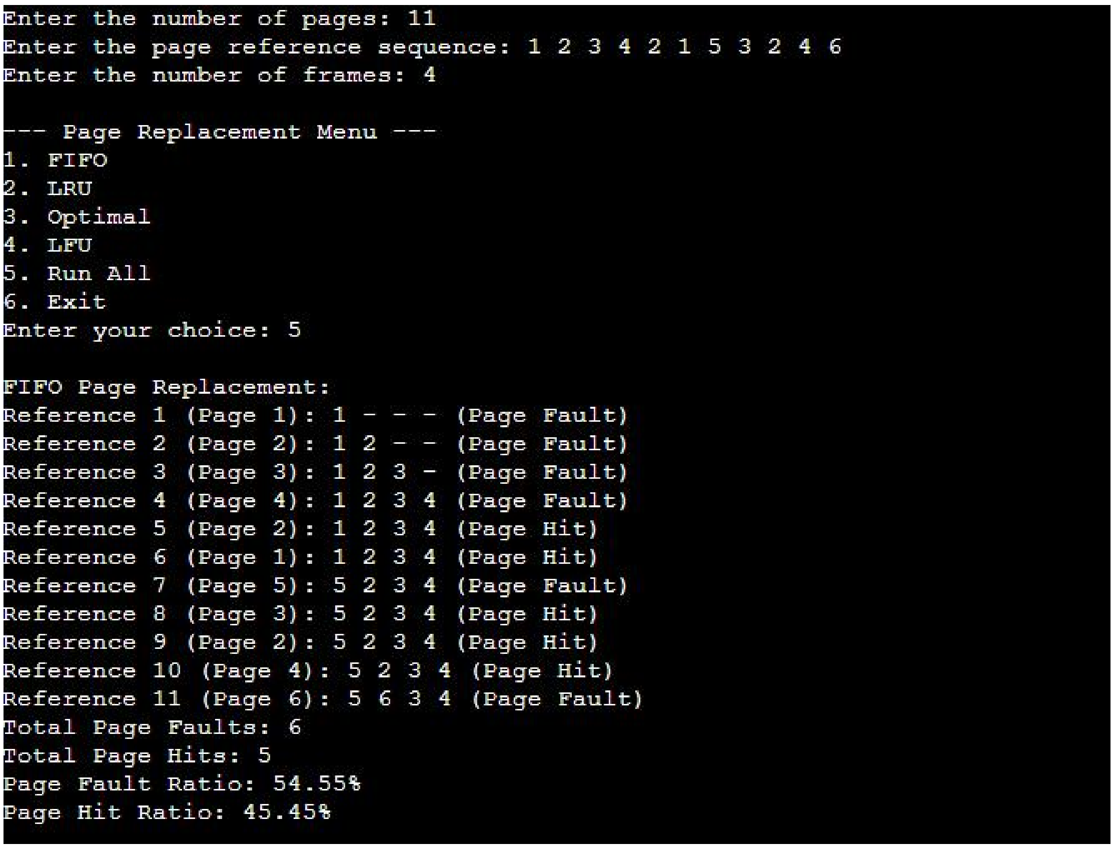
Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <limits.h>

// Function to check if page is in frame
bool isPagePresent(int frames[], int n, int page) {
   for (int i = 0; i < n; i++) {
      if (frames[i] == page) return true;
   }
   return false;
}

// Function to print the frames
void printFrames(int frames[], int n, bool isPageFault) {
   for (int i = 0; i < n; i++) {
      if (frames[i] == -1) printf("- ");
      else printf("%d ", frames[i]);
   }
   printf("%s\n", isPageFault ? "(Page Fault)" : "(Page Hit)");
}

void printSummary(int pageFaults, int pageHits, int total) {
   float faultRatio = (float)pageFaults / total * 100;
   float hitRatio = (float)pageHits / total * 100;
   printf("Total Page Faults: %d\n", pageFaults);
   printf("Total Page Hits: %d\n", pageHits);
   printf("Page Fault Ratio: %.2f%%\n", faultRatio);
   printf("Page Hit Ratio: %.2f%%\n", hitRatio);
}

// FIFO
void fifo(int pages[], int n, int frameCount) {
    int frames[frameCount], pageFaults = 0, pageHits = 0, pointer
= 0;
    for (int i = 0; i < frameCount; i++) frames[i] = -1;

    printf("\nFIFO Page Replacement:\n");
    for (int i = 0; i < n; i++) {
        printf("Reference %d (Page %d): ", i + 1, pages[i]);
        if (!isPagePresent(frames, frameCount, pages[i])) {
            frames[pointer] = pages[i];
            pointer = (pointer + 1) % frameCount;
            pageFaults++;
            printFrames(frames, frameCount, true);
        } else {
            pageHits++;
            printFrames(frames, frameCount, false);
        }
    }
    printSummary(pageFaults, pageHits, n);
}

// LRU
void lru(int pages[], int n, int frameCount) {
    int frames[frameCount], counter[frameCount], time = 0,
pageFaults = 0, pageHits = 0;
    for (int i = 0; i < frameCount; i++) {
        frames[i] = -1;
        counter[i] = 0;
    }

    printf("\nLRU Page Replacement:\n");
    for (int i = 0; i < n; i++) {
        time++;
        printf("Reference %d (Page %d): ", i + 1, pages[i]);
        if (!isPagePresent(frames, frameCount, pages[i])) {
            int lruIndex = 0;
            for (int j = 1; j < frameCount; j++) {
                if (counter[j] < counter[lruIndex]) lruIndex = j;
            }
            frames[lruIndex] = pages[i];
            counter[lruIndex] = time;
            pageFaults++;
            printFrames(frames, frameCount, true);
        } else {
            for (int j = 0; j < frameCount; j++) {
                if (frames[j] == pages[i]) {
                    counter[j] = time;
                    break;
                }
            }
            pageHits++;
            printFrames(frames, frameCount, false);
        }
    }
    printSummary(pageFaults, pageHits, n);
}

// Optimal
void optimal(int pages[], int n, int frameCount) {
    int frames[frameCount], pageFaults = 0, pageHits = 0;
    for (int i = 0; i < frameCount; i++) frames[i] = -1;

    printf("\nOptimal Page Replacement:\n");
    for (int i = 0; i < n; i++) {
        printf("Reference %d (Page %d): ", i + 1, pages[i]);
        if (!isPagePresent(frames, frameCount, pages[i])) {
            int farthest = i, replaceIndex = 0;
            bool found;

            for (int j = 0; j < frameCount; j++) {
                found = false;
                for (int k = i + 1; k < n; k++) {
                    if (frames[j] == pages[k]) {
                        if (k > farthest) {
                            farthest = k;
                            replaceIndex = j;
                        }
                        found = true;
                        break;
                    }
                }
                if (!found) {
                    replaceIndex = j;
                    break;
                }
            }

            frames[replaceIndex] = pages[i];
            pageFaults++;
            printFrames(frames, frameCount, true);
        } else {
            pageHits++;
            printFrames(frames, frameCount, false);
        }
    }
    printSummary(pageFaults, pageHits, n);
}

// LFU
void lfu(int pages[], int n, int frameCount) {
    int frames[frameCount], frequency[frameCount], pageFaults
= 0, pageHits = 0;
    for (int i = 0; i < frameCount; i++) {
        frames[i] = -1;
        frequency[i] = 0;
    }
    printf("\nLFU Page Replacement:\n");
    for (int i = 0; i < n; i++) {
        printf("Reference %d (Page %d): ", i + 1, pages[i]);
        if (!isPagePresent(frames, frameCount, pages[i])) {
            int lfuIndex = 0;
            for (int j = 1; j < frameCount; j++) {
                if (frequency[j] < frequency[lfuIndex]) {
                    lfuIndex = j;
                }
            }
            frames[lfuIndex] = pages[i];
            frequency[lfuIndex] = 1;
            pageFaults++;
            printFrames(frames, frameCount, true);
        } else {
            for (int j = 0; j < frameCount; j++) {
                if (frames[j] == pages[i]) {
                    frequency[j]++;
                    break;
                }
            }
            pageHits++;
            printFrames(frames, frameCount, false);
        }
    }
    printSummary(pageFaults, pageHits, n);
}

// Main Menu
int main() {
    int n, frameCount, choice;
    char cont;

    printf("Enter the number of pages: ");
```

```
scanf("%d", &n);

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

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

do {
    printf("\n--- Page Replacement Menu ---\n");
    printf("1. FIFO\n");
    printf("2. LRU\n");
    printf("3. Optimal\n");
    printf("4. LFU\n");
    printf("5. Run All\n");
    printf("6. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1: fifo(pages, n, frameCount); break;
        case 2: lru(pages, n, frameCount); break;
        case 3: optimal(pages, n, frameCount); break;
        case 4: lfu(pages, n, frameCount); break;
        case 5:
            fifo(pages, n, frameCount);
            lru(pages, n, frameCount);
            optimal(pages, n, frameCount);
            lfu(pages, n, frameCount);
            break;
        case 6: exit(0);
        default: printf("Invalid choice!\n");
        }

        printf("\nDo you want to continue? (y/n): ");
        scanf(" %c", &cont);
    } while (cont == 'y' || cont == 'Y');

    return 0;
}
```

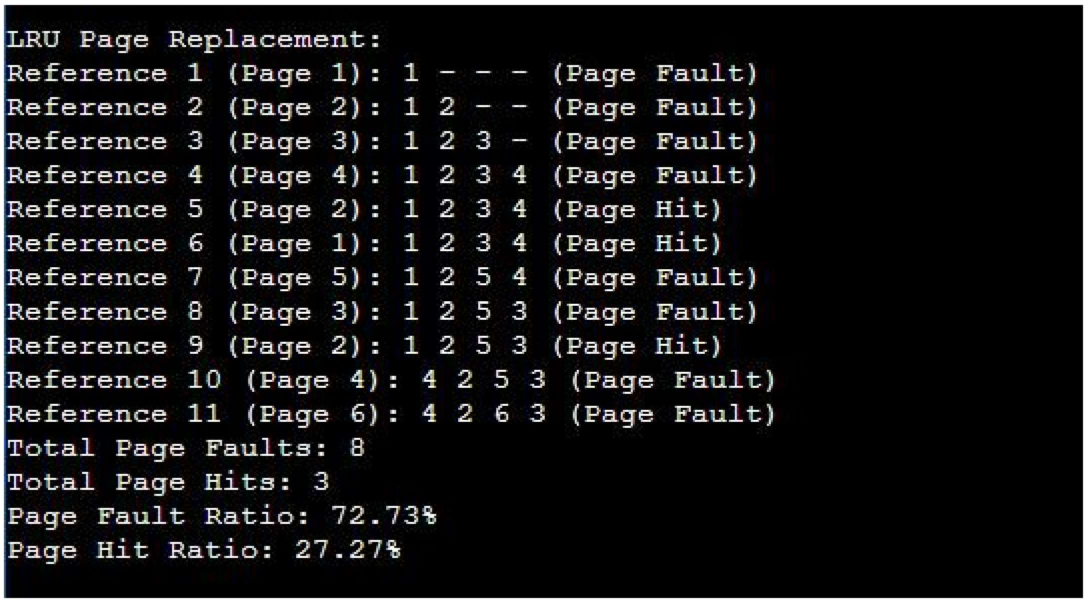
Output :

FIFO:



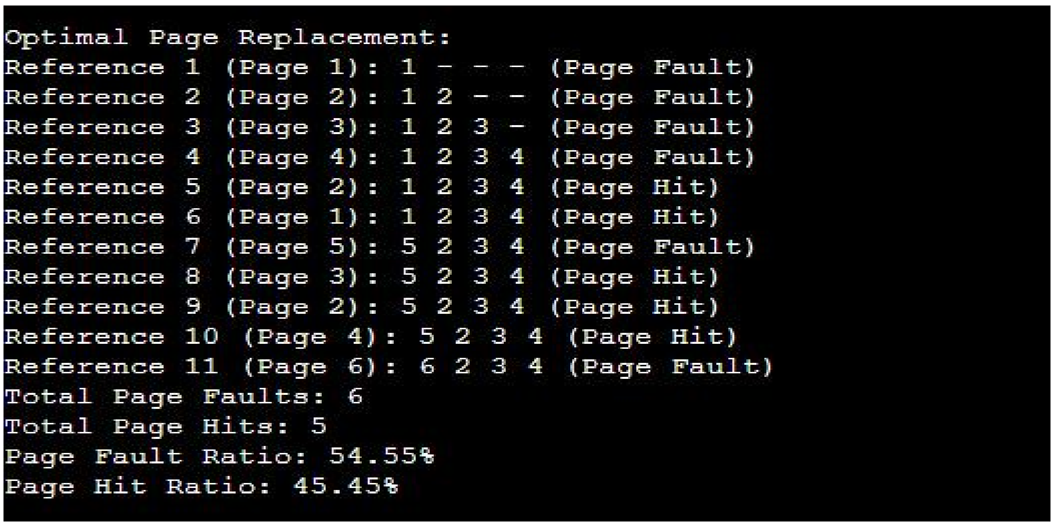
```
Enter the number of pages: 11
Enter the page reference sequence: 1 2 3 4 2 1 5 3 2 4 6
Enter the number of frames: 4

--- Page Replacement Menu ---
1. FIFO
2. LRU
3. Optimal
4. LFU
5. Run All
6. Exit
Enter your choice: 5

FIFO Page Replacement:
Reference 1 (Page 1): 1 - - - (Page Fault)
Reference 2 (Page 2): 1 2 - - (Page Fault)
Reference 3 (Page 3): 1 2 3 - (Page Fault)
Reference 4 (Page 4): 1 2 3 4 (Page Fault)
Reference 5 (Page 2): 1 2 3 4 (Page Hit)
Reference 6 (Page 1): 1 2 3 4 (Page Hit)
Reference 7 (Page 5): 5 2 3 4 (Page Fault)
Reference 8 (Page 3): 5 2 3 4 (Page Hit)
Reference 9 (Page 2): 5 2 3 4 (Page Hit)
Reference 10 (Page 4): 5 2 3 4 (Page Hit)
Reference 11 (Page 6): 5 6 3 4 (Page Fault)
Total Page Faults: 6
Total Page Hits: 5
Page Fault Ratio: 54.55%
Page Hit Ratio: 45.45%
```

LRU:



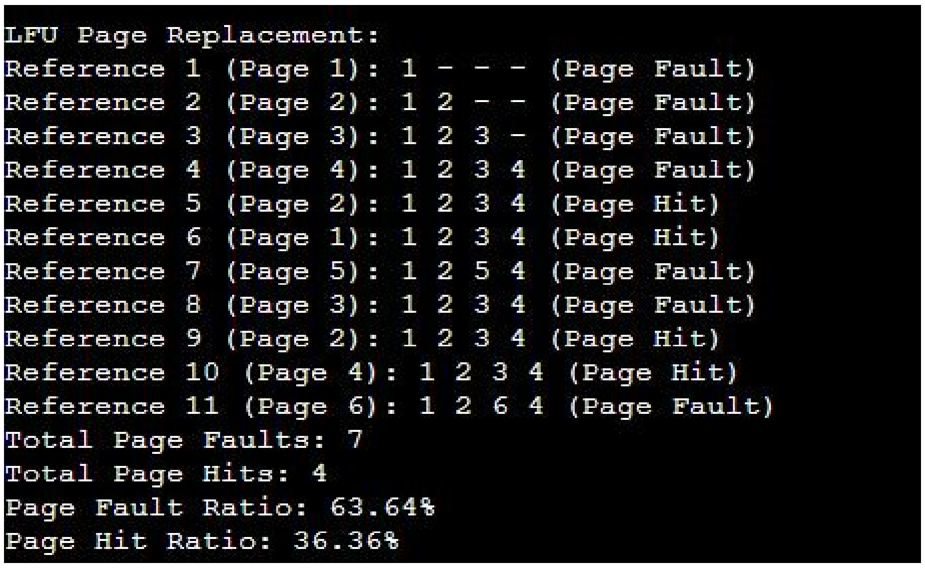
```
LRU Page Replacement:
Reference 1 (Page 1): 1 - - - (Page Fault)
Reference 2 (Page 2): 1 2 - - (Page Fault)
Reference 3 (Page 3): 1 2 3 - (Page Fault)
Reference 4 (Page 4): 1 2 3 4 (Page Fault)
Reference 5 (Page 2): 1 2 3 4 (Page Hit)
Reference 6 (Page 1): 1 2 3 4 (Page Hit)
Reference 7 (Page 5): 1 2 5 4 (Page Fault)
Reference 8 (Page 3): 1 2 5 3 (Page Fault)
Reference 9 (Page 2): 1 2 5 3 (Page Hit)
Reference 10 (Page 4): 4 2 5 3 (Page Fault)
Reference 11 (Page 6): 4 2 6 3 (Page Fault)
Total Page Faults: 8
Total Page Hits: 3
Page Fault Ratio: 72.73%
Page Hit Ratio: 27.27%
```

OPTIMAL:



```
Optimal Page Replacement:
Reference 1 (Page 1): 1 - - - (Page Fault)
Reference 2 (Page 2): 1 2 - - (Page Fault)
Reference 3 (Page 3): 1 2 3 - (Page Fault)
Reference 4 (Page 4): 1 2 3 4 (Page Fault)
Reference 5 (Page 2): 1 2 3 4 (Page Hit)
Reference 6 (Page 1): 1 2 3 4 (Page Hit)
Reference 7 (Page 5): 5 2 3 4 (Page Fault)
Reference 8 (Page 3): 5 2 3 4 (Page Hit)
Reference 9 (Page 2): 5 2 3 4 (Page Hit)
Reference 10 (Page 4): 5 2 3 4 (Page Hit)
Reference 11 (Page 6): 6 2 3 4 (Page Fault)
Total Page Faults: 6
Total Page Hits: 5
Page Fault Ratio: 54.55%
Page Hit Ratio: 45.45%
```

LFU:



```
LFU Page Replacement:
Reference 1 (Page 1): 1 - - - (Page Fault)
Reference 2 (Page 2): 1 2 - - (Page Fault)
Reference 3 (Page 3): 1 2 3 - (Page Fault)
Reference 4 (Page 4): 1 2 3 4 (Page Fault)
Reference 5 (Page 2): 1 2 3 4 (Page Hit)
Reference 6 (Page 1): 1 2 3 4 (Page Hit)
Reference 7 (Page 5): 1 2 5 4 (Page Fault)
Reference 8 (Page 3): 1 2 3 4 (Page Fault)
Reference 9 (Page 2): 1 2 3 4 (Page Hit)
Reference 10 (Page 4): 1 2 3 4 (Page Hit)
Reference 11 (Page 6): 1 2 6 4 (Page Fault)
Total Page Faults: 7
Total Page Hits: 4
Page Fault Ratio: 63.64%
Page Hit Ratio: 36.36%
```