OPERATING SYSTEM-CSA0401
PROGRAM 31 - 40

KRITHIYA G
192421150

31. First-In-First-Out (FIFO) Page Replacement

main.c

```c
#include <stdio.h>

void FIFO(int pages[], int n, int capacity) {
    int frames[capacity];
    int front = 0, count = 0, faults = 0;

    for (int i = 0; i < capacity; i++)
        frames[i] = -1;

    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < capacity; j++) {
            if (frames[j] == pages[i]) {
                found = 1;
                break;
            }
        }

        if (!found) {
```

Output

```
Frames: 1 -1 -1
Frames: 1 3 -1
Frames: 1 3 0
Frames: 1 3 0
Frames: 5 3 0
Frames: 5 6 0
Total Page Faults: 5


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

32. Least Recently Used (LRU) Page Replacement

main.c

```c
#include <stdio.h>

int findLRU(int time[], int n) {
    int min = time[0], pos = 0;
    for (int i = 1; i < n; i++) {
        if (time[i] < min) {
            min = time[i];
            pos = i;
        }
    }
    return pos;
}

void LRU(int pages[], int n, int capacity) {
    int frames[capacity], time[capacity], counter = 0, faults =
        0;
    for (int i = 0; i < capacity; i++)
        frames[i] = -1;

```

Output

```
Frames: 7 -1 -1
Frames: 7 0 -1
Frames: 7 0 1
Frames: 2 0 1
Frames: 2 0 1
Frames: 2 0 3
Frames: 2 0 3
Frames: 4 0 3
Frames: 4 0 2
Total Page Faults: 7


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

33. Optimal Page Replacement

main.c

```c
#include <stdio.h>

int predict(int pages[], int n, int index, int frames[], int capacity) {
    int res = -1, farthest = index;
    for (int i = 0; i < capacity; i++) {
        int j;
        for (j = index; j < n; j++) {
            if (frames[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }
        if (j == n)
            return i;
    }
```

Output

```
Frames: 7 -1 -1
Frames: 7 0 -1
Frames: 7 0 1
Frames: 2 0 1
Frames: 2 0 1
Frames: 2 0 3
Frames: 2 0 3
Frames: 2 4 3
Frames: 2 4 3
Total Page Faults: 6


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

34. Sequential File Allocation Simulation

main.c

```c
#include <stdio.h>

int main() {
    int files[10], start[10], length[10], n;

    printf("Enter number of files: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter start block and length of file %d: ", i + 1);
        scanf("%d%d", &start[i], &length[i]);

        for (int j = start[i]; j < start[i] + length[i]; j++)
            files[j] = i + 1;
    }

    printf("\nFile Allocation Table:\n");
    printf("File\tStart\tLength\n");
```

Output

```
Enter number of files: 3
Enter start block and length of file 1: 5 3
Enter start block and length of file 2: 7 4
Enter start block and length of file 3: 9 2

File Allocation Table:
File    Start   Length
1   5   3
2   7   4
3   9   2


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

35. Indexed File Allocation Simulation

main.c

```c
#include <stdio.h>

int main() {
    int indexBlock[10], files[10][10], n, blocks;

    printf("Enter number of files: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter index block for file %d: ", i + 1);
        scanf("%d", &indexBlock[i]);
        printf("Enter number of blocks: ");
        scanf("%d", &blocks);
        printf("Enter %d blocks: ", blocks);
        for (int j = 0; j < blocks; j++)
            scanf("%d", &files[i][j]);
        files[i][blocks] = -1;
    }

```

Output

```
Enter number of files: 2
Enter index block for file 1: 5
Enter number of blocks: 3
Enter 3 blocks: 10 12 15
Enter index block for file 2: 6
Enter number of blocks: 4
Enter 4 blocks: 20 22 24 26

Indexed Allocation Table:
File 1 Index 5 -> 10 12 15
File 2 Index 6 -> 20 22 24 26


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

36. Linked File Allocation Simulation

main.c

```c
#include <stdio.h>

struct Block {
    int data;
    int next;
};

int main() {
    struct Block file[20];
    int start, end, n;

    printf("Enter number of blocks: ");
    scanf("%d", &n);

    printf("Enter start and end block of file: ");
    scanf("%d%d", &start, &end);

    for (int i = start; i <= end; i++) {
        file[i].data = i;
```

Output

```
Enter number of blocks: 10
Enter start and end block of file: 3 7

Linked Allocation:
Block -> Next
3 -> 4
4 -> 5
5 -> 6
6 -> 7
7 -> -1


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

37. First-Come-First-Serve (FCFS) Disk Scheduling

main.c

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int n, head, seek = 0;

    printf("Enter number of requests: ");
    scanf("%d", &n);

    int req[n];
    printf("Enter request sequence: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &req[i]);

    printf("Enter initial head position: ");
    scanf("%d", &head);

    for (int i = 0; i < n; i++) {
        seek += abs(head - req[i]);
```

Output

```
Enter number of requests: 5
Enter request sequence: 98 183 37 122 14
Enter initial head position: 53
Total seek time: 469


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

38. SCAN Disk Scheduling

main.c

```c
#include <stdio.h>
#include <stdlib.h>

int compare(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}

int main() {
    int n, head, direction, size = 200;
    printf("Enter number of requests: ");
    scanf("%d", &n);

    int req[n + 1];
    printf("Enter request sequence: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &req[i]);

    printf("Enter head position: ");
    scanf("%d", &head);
```

Output

```
Enter number of requests: 8
Enter request sequence: 98 183 37 122 14 124 65 67
Enter head position: 53
Enter direction (0 for left, 1 for right): 1
Seek sequence: 53 65 67 98 122 124 183 199 37 14
Total seek time: 169


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

39. C-SCAN Disk Scheduling

main.c

```c
#include <stdio.h>
#include <stdlib.h>

int compare(const void* a, const void* b) {
    return (*(int*)a - *(int*)b);
}

int main() {
    int n, head, size = 200;
    printf("Enter number of requests: ");
    scanf("%d", &n);

    int req[n + 1];
    printf("Enter request sequence: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &req[i]);

    printf("Enter initial head position: ");
    scanf("%d", &head);
```

Output

```
Enter number of requests: 8
Enter request sequence: 98 183 37 122 14 124 65 67
Enter initial head position: 53
Seek sequence: 53 65 67 98 122 124 183 0 14 37
Total seek time: 368


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

40. Illustrate File Access Permissions and User Types in Linux

main.c

```c
#include <stdio.h>
#include <sys/stat.h>
#include <unistd.h>

void printPermissions(mode_t mode) {
    printf("User: ");
    printf((mode & S_IRUSR) ? "r" : "-");
    printf((mode & S_IWUSR) ? "w" : "-");
    printf((mode & S_IXUSR) ? "x" : "-");

    printf(" | Group: ");
    printf((mode & S_IRGRP) ? "r" : "-");
    printf((mode & S_IWGRP) ? "w" : "-");
    printf((mode & S_IXGRP) ? "x" : "-");

    printf(" | Others: ");
    printf((mode & S_IROTH) ? "r" : "-");
    printf((mode & S_IWOTH) ? "w" : "-");
    printf((mode & S_IXOTH) ? "x" : "-");
```

Output

```
Enter the file name: example.txt
Error retrieving file info: No such file or directory


=== Code Exited With Errors ===
```