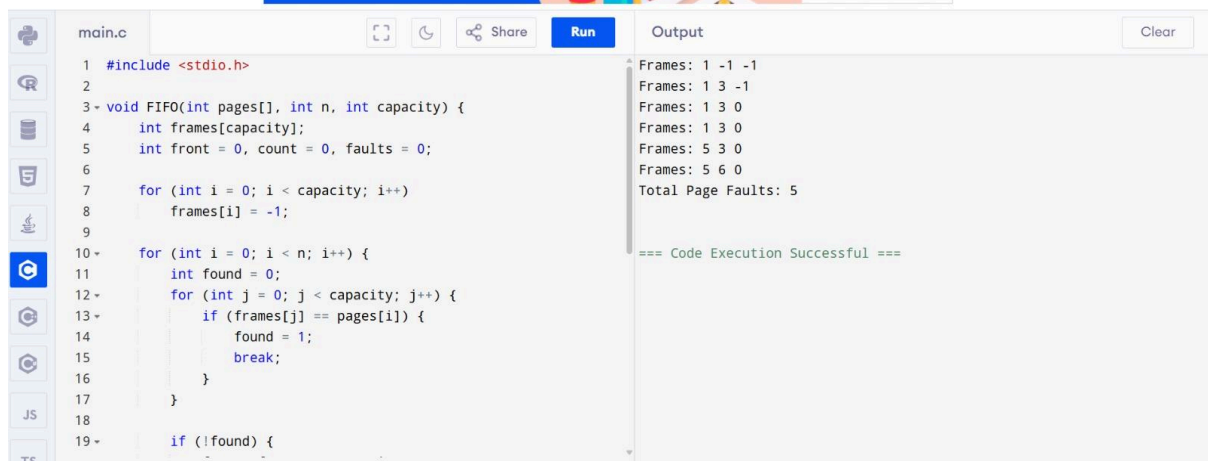


# OPERATING SYSTEM-CSA0401

## PROGRAM 31 - 40

KRITHIYA G  
192421150

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

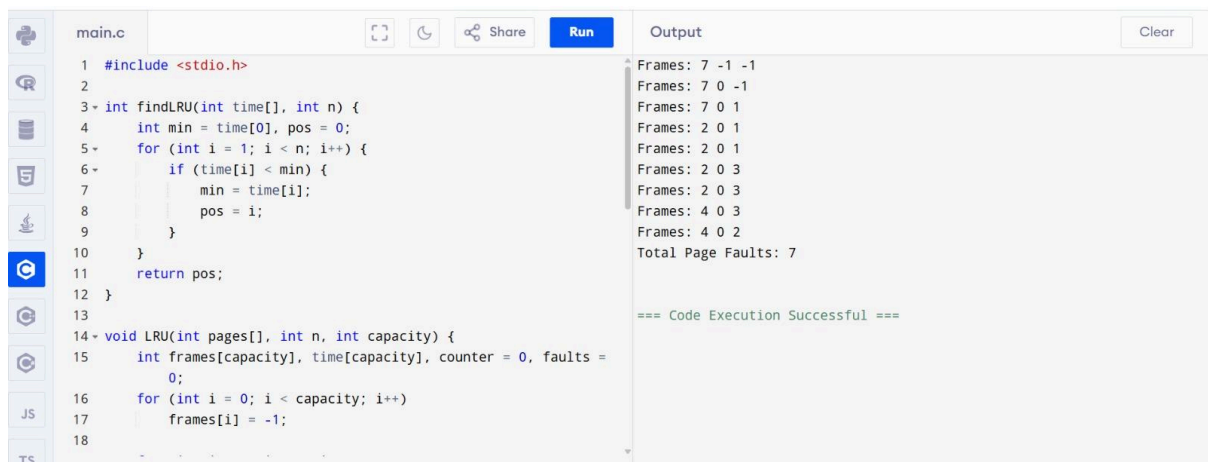


```
main.c
1 #include <stdio.h>
2
3 void FIFO(int pages[], int n, int capacity) {
4     int frames[capacity];
5     int front = 0, count = 0, faults = 0;
6
7     for (int i = 0; i < capacity; i++)
8         frames[i] = -1;
9
10    for (int i = 0; i < n; i++) {
11        int found = 0;
12        for (int j = 0; j < capacity; j++) {
13            if (frames[j] == pages[i]) {
14                found = 1;
15                break;
16            }
17        }
18        if (!found) {
19            // Fault occurred, replace the page at front
20            frames[front] = pages[i];
21            front = (front + 1) % capacity;
22            faults++;
23        }
24    }
25    printf("Total Page Faults: %d\n", faults);
26}
```

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
1 #include <stdio.h>
2
3 int findLRU(int time[], int n) {
4     int min = time[0], pos = 0;
5     for (int i = 1; i < n; i++) {
6         if (time[i] < min) {
7             min = time[i];
8             pos = i;
9         }
10    }
11    return pos;
12 }
13
14 void LRU(int pages[], int n, int capacity) {
15     int frames[capacity], time[capacity], counter = 0, faults = 0;
16     for (int i = 0; i < capacity; i++)
17         frames[i] = -1;
18
19     for (int i = 0; i < n; i++) {
20         int pos = findLRU(frames, counter);
21         if (frames[pos] != pages[i]) {
22             // Fault occurred, replace the page at pos
23             frames[pos] = pages[i];
24             time[pos] = counter;
25             counter++;
26             faults++;
27         }
28     }
29    printf("Total Page Faults: %d\n", faults);
30}
```

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
1 #include <stdio.h>
2
3 int predict(int pages[], int n, int index, int frames[], int capacity) {
4     int res = -1, farthest = index;
5     for (int i = 0; i < capacity; i++) {
6         int j;
7         for (j = index; j < n; j++) {
8             if (frames[i] == pages[j]) {
9                 if (j > farthest) {
10                     farthest = j;
11                     res = i;
12                 }
13             }
14         }
15     }
16     if (j == n)
17         return i;
18 }
```

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
1 #include <stdio.h>
2
3 int main() {
4     int files[10], start[10], length[10], n;
5
6     printf("Enter number of files: ");
7     scanf("%d", &n);
8
9     for (int i = 0; i < n; i++) {
10         printf("Enter start block and length of file %d: ", i + 1);
11         scanf("%d%d", &start[i], &length[i]);
12
13         for (int j = start[i]; j < start[i] + length[i]; j++)
14             files[j] = i + 1;
15     }
16
17     printf("\nFile Allocation Table:\n");
18     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
1 #include <stdio.h>
2
3 int main() {
4     int indexBlock[10], files[10][10], n, blocks;
5
6     printf("Enter number of files: ");
7     scanf("%d", &n);
8
9     for (int i = 0; i < n; i++) {
10         printf("Enter index block for file %d: ", i + 1);
11         scanf("%d", &indexBlock[i]);
12         printf("Enter number of blocks: ");
13         scanf("%d", &blocks);
14         printf("Enter %d blocks: ", blocks);
15         for (int j = 0; j < blocks; j++)
16             scanf("%d", &files[i][j]);
17         files[i][blocks] = -1;
18     }
19 }
```

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
1 #include <stdio.h>
2
3 struct Block {
4     int data;
5     int next;
6 };
7
8 int main() {
9     struct Block file[20];
10    int start, end, n;
11
12    printf("Enter number of blocks: ");
13    scanf("%d", &n);
14
15    printf("Enter start and end block of file: ");
16    scanf("%d%d", &start, &end);
17
18    for (int i = start; i <= end; i++) {
19        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
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int n, head, seek = 0;
6
7     printf("Enter number of requests: ");
8     scanf("%d", &n);
9
10    int req[n];
11    printf("Enter request sequence: ");
12    for (int i = 0; i < n; i++)
13        scanf("%d", &req[i]);
14
15    printf("Enter initial head position: ");
16    scanf("%d", &head);
17
18    for (int i = 0; i < n; i++) {
19        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 ===
```

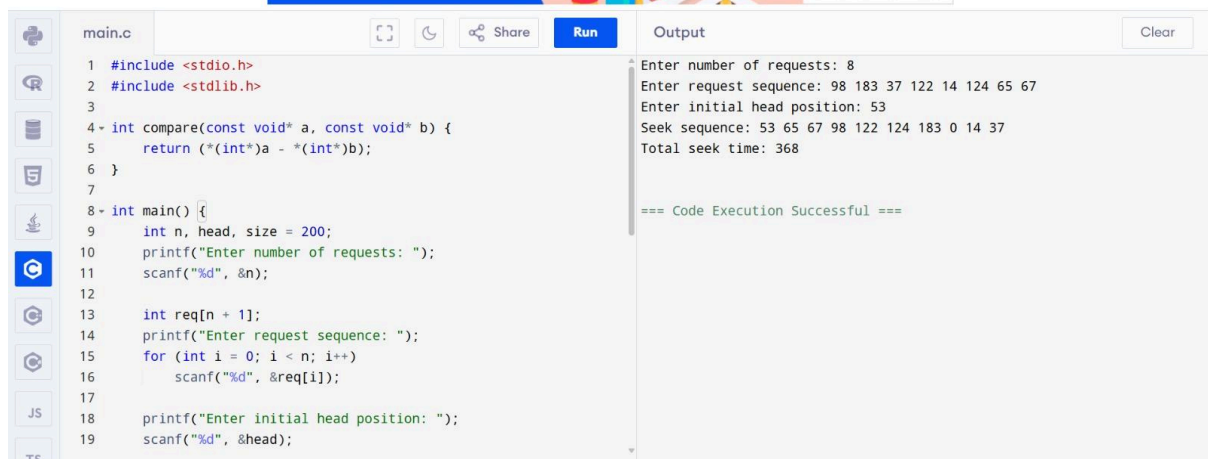
### 38. SCAN Disk Scheduling

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int compare(const void *a, const void *b) {
5     return (*(int*)a - *(int*)b);
6 }
7
8 int main() {
9     int n, head, direction, size = 200;
10    printf("Enter number of requests: ");
11    scanf("%d", &n);
12
13    int req[n + 1];
14    printf("Enter request sequence: ");
15    for (int i = 0; i < n; i++)
16        scanf("%d", &req[i]);
17
18    printf("Enter head position: ");
19    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



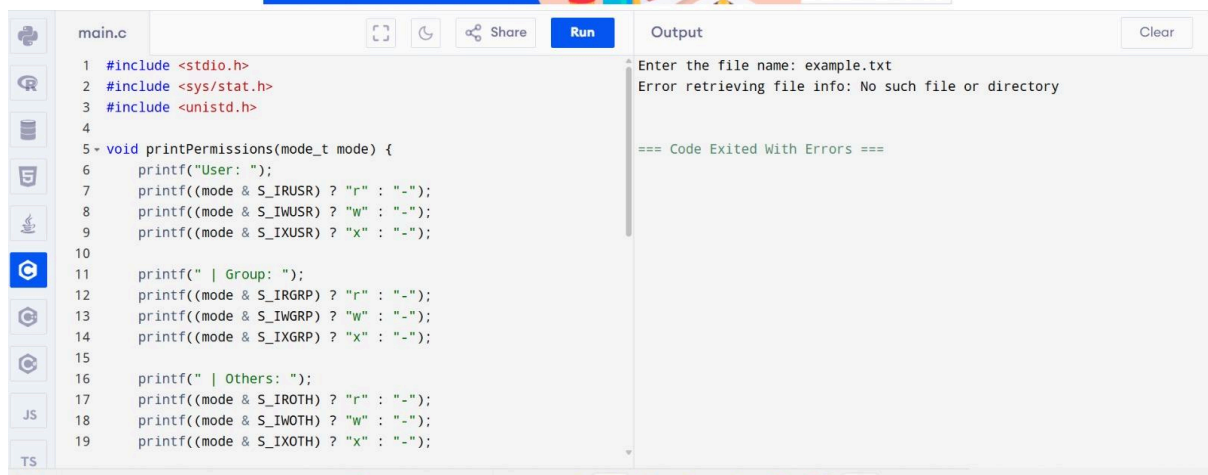
```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int compare(const void* a, const void* b) {
5     return (*(int*)a - *(int*)b);
6 }
7
8 int main() {
9     int n, head, size = 200;
10    printf("Enter number of requests: ");
11    scanf("%d", &n);
12
13    int req[n + 1];
14    printf("Enter request sequence: ");
15    for (int i = 0; i < n; i++)
16        scanf("%d", &req[i]);
17
18    printf("Enter initial head position: ");
19    scanf("%d", &head);
20
21    // C-SCAN Disk Scheduling
22    // Sort the request sequence
23    qsort(req, n, sizeof(int), compare);
24
25    // Calculate the seek sequence
26    int seekSeq[2 * n];
27    int seekSeqIndex = 0;
28
29    // Forward pass
30    for (int i = 0; i < n; i++) {
31        seekSeq[seekSeqIndex++] = req[i];
32    }
33
34    // Backward pass
35    for (int i = n - 1; i >= 0; i--) {
36        seekSeq[seekSeqIndex++] = req[i];
37    }
38
39    // Total seek time
40    int totalSeekTime = 0;
41    for (int i = 0; i < seekSeqIndex; i++) {
42        totalSeekTime += abs(seekSeq[i] - head);
43        head = seekSeq[i];
44    }
45
46    printf("Seek sequence: ");
47    for (int i = 0; i < seekSeqIndex; i++) {
48        printf("%d ", seekSeq[i]);
49    }
50    printf("\n");
51
52    printf("Total seek time: %d\n", totalSeekTime);
53
54    return 0;
55 }
```

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
1 #include <stdio.h>
2 #include <sys/stat.h>
3 #include <unistd.h>
4
5 void printPermissions(mode_t mode) {
6     printf("User: ");
7     printf((mode & S_IRUSR) ? "r" : "-");
8     printf((mode & S_IWUSR) ? "w" : "-");
9     printf((mode & S_IXUSR) ? "x" : "-");
10
11     printf(" | Group: ");
12     printf((mode & S_IRGRP) ? "r" : "-");
13     printf((mode & S_IWGRP) ? "w" : "-");
14     printf((mode & S_IXGRP) ? "x" : "-");
15
16     printf(" | Others: ");
17     printf((mode & S_IROTH) ? "r" : "-");
18     printf((mode & S_IWOTH) ? "w" : "-");
19     printf((mode & S_IXOTH) ? "x" : "-");
20 }
21
22 int main() {
23     char filename[100];
24     printf("Enter the file name: ");
25     scanf("%s", filename);
26
27     struct stat st;
28     if (stat(filename, &st) != 0) {
29         printf("Error retrieving file info: No such file or directory\n");
30     } else {
31         printPermissions(st.st_mode);
32     }
33
34     return 0;
35 }
```

Output

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

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