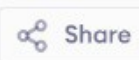




main.c



Run

Output

Clear

```
1 #include <stdio.h>
2
3 #define MAX_FRAMES 10
4 #define MAX_PAGES 50
5
6 void fifoPageReplacement(int pages[], int n, int frames) {
7     int pageFrame[MAX_FRAMES];
8     int front = 0, rear = 0, count = 0, pageFaults = 0;
9
10    for (int i = 0; i < frames; i++) {
11        pageFrame[i] = -1; // Initialize frames to -1 (empty)
12    }
13
14    for (int i = 0; i < n; i++) {
15        int page = pages[i];
16        int found = 0;
17
18        // Check if the page is already in frame
19        for (int j = 0; j < frames; j++) {
20            if (pageFrame[j] == page) {
```

```
Enter the number of pages: 5
Enter the page reference sequence: 1

ui
Enter the number of frames: Step 1:
Step 2:
Step 3:
Step 4:
Step 5:
Total Page Faults: 5
```

=== Code Execution Successful ===

main.c

```
59- int main() {  
60     int pages[MAX_PAGES], n, frames;  
61  
62     printf("Enter the number of pages: ");  
63     scanf("%d", &n);  
64  
65     printf("Enter the page reference sequence: ");  
66-   for (int i = 0; i < n; i++) {  
67       scanf("%d", &pages[i]);  
68   }  
69  
70     printf("Enter the number of frames: ");  
71     scanf("%d", &frames);  
72  
73     lruPageReplacement(pages, n, frames);  
74  
75     return 0;  
76 }  
77
```

Share

Run

Output

Clear

Enter the number of pages: 2
Enter the page reference sequence: 1
1
Enter the number of frames: 2
Step 1: 1 -
Step 2: 1 -
Total Page Faults: 1

=== Code Execution Successful ===

Build an AI-driven tech platform.
Sponsored by: Intuit

Learn More

Programiz PRO >

main.c

Run

Output

Clear

```
72 int main() {  
73     int pages[MAX_PAGES], n, frames;  
74  
75     printf("Enter the number of pages: ");  
76     scanf("%d", &n);  
77  
78     printf("Enter the page reference sequence: ");  
79     for (int i = 0; i < n; i++) {  
80         scanf("%d", &pages[i]);  
81     }  
82  
83     printf("Enter the number of frames: ");  
84     scanf("%d", &frames);  
85  
86     optimalPageReplacement(pages, n, frames);  
87  
88     return 0;  
89 }  
90
```

Enter the number of pages: 1
Enter the page reference sequence: 2
Enter the number of frames: 3
Step 1: 2 - -
Total Page Faults: 1

=== Code Execution Successful ===



main.c



Share

Run

Output

Clear

```
22 int main() {
23     Record records[MAX_RECORDS];
24     int n;
25
26     printf("Enter the number of records: ");
27     scanf("%d", &n);
28     getchar(); // Consume newline
29
30     for (int i = 0; i < n; i++) {
31         records[i].id = i + 1;
32         printf("Enter data for record %d: ", i + 1);
33         fgets(records[i].data, RECORD_SIZE, stdin);
34         records[i].data[strcspn(records[i].data, "\n")] = 0; //
            Remove newline
35     }
36
37     readSequentialFile(records, n);
38     return 0;
39 }
```

```
Enter the number of records: 2
Enter data for record 1: 1
Enter data for record 2: 2
```

```
Reading file sequentially:
Record 1: 1
Record 2: 2
```

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





main.c



Share

Run

Output

Clear

```
32 int n;
33
34 printf("Enter the number of blocks: ");
35 scanf("%d", &n);
36 getchar(); // Consume newline
37
38 indexBlock.blockCount = n;
39 for (int i = 0; i < n; i++) {
40     blocks[i].id = i + 1;
41     printf("Enter data for block %d: ", i + 1);
42     fgets(blocks[i].data, BLOCK_SIZE, stdin);
43     blocks[i].data[strcspn(blocks[i].data, "\n")] = 0; //
        Remove newline
44     indexBlock.blockPointers[i] = i;
45 }
46
47 readIndexedFile(blocks, indexBlock);
48 return 0;
49 }
50
```

```
Enter the number of blocks: 2
Enter data for block 1: 1
Enter data for block 2: 5

Reading file using indexed allocation:
Block 1: 1
Block 2: 5

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



main.c



Share

Run

Output

Clear

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define MAX_BLOCKS 100
6 #define BLOCK_SIZE 50
7
8 // Structure to represent a file block
9 typedef struct FileBlock {
10     int id;
11     char data[BLOCK_SIZE];
12     struct FileBlock *next;
13 } FileBlock;
14
15 // Function to read file using linked allocation
16 void readLinkedFile(FileBlock *head) {
17     printf("\nReading file using linked allocation:\n");
18     FileBlock *current = head;
19     while (current != NULL) {
20         printf("Block %d: %s\n", current->id, current->data);
```

```
Enter the number of blocks: 3
Enter data for block 1: 1
Enter data for block 2: 4
Enter data for block 3: 1

Reading file using linked allocation:
Block 1: 1
Block 2: 4
Block 3: 1

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



main.c



Share

Run

Output

Clear

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX_REQUESTS 100
5
6 // Function to simulate FCFS disk scheduling
7 void fcfsDiskScheduling(int requests[], int n, int head) {
8     int totalMovement = 0;
9
10    printf("\nDisk request order: ");
11    for (int i = 0; i < n; i++) {
12        printf("%d ", requests[i]);
13        totalMovement += abs(requests[i] - head);
14        head = requests[i];
15    }
16
17    printf("\nTotal head movement: %d\n", totalMovement);
18 }
19
20 int main() {
```

```
Enter the number of disk requests: 4
Enter the disk request sequence: 4
4
3
2
Enter the initial head position: 1

Disk request order: 4 4 3 2
Total head movement: 5

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




main.c



Share

Run

Output

Clear

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX_REQUESTS 100
5
6 // Function to simulate C-SCAN disk scheduling
7 void cScanDiskScheduling(int requests[], int n, int head, int
   diskSize) {
8     int totalMovement = 0, i;
9     int left[MAX_REQUESTS], right[MAX_REQUESTS], lCount = 0,
       rCount = 0;
10
11     // Separating requests into left and right of head
12     for (i = 0; i < n; i++) {
13         if (requests[i] < head)
14             left[lCount++] = requests[i];
15         else
16             right[rCount++] = requests[i];
17     }
18 }
```

```
Enter the number of disk requests: 6
Enter the disk request sequence: 1
2
3
45
5
6
Enter the initial head position: 8
Enter the disk size: 8

Disk request order: 1 2 3 4 5 6
Total head movement: 14

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

Airplane mode off



main.c



Share

Run

Output

Clear

```
17 }
18
19 // Sorting left and right requests
20 for (i = 0; i < lCount - 1; i++) {
21     for (int j = 0; j < lCount - i - 1; j++) {
22         if (left[j] > left[j + 1]) {
23             int temp = left[j];
24             left[j] = left[j + 1];
25             left[j + 1] = temp;
26         }
27     }
28 }
29 for (i = 0; i < rCount - 1; i++) {
30     for (int j = 0; j < rCount - i - 1; j++) {
31         if (right[j] > right[j + 1]) {
32             int temp = right[j];
33             right[j] = right[j + 1];
34             right[j + 1] = temp;
35         }
36     }
```

```
Enter the number of disk requests: 4
Enter the disk request sequence: 5

1
2
3
Enter the initial head position: 4
Enter the disk size: 4
Enter direction (1 for high, 0 for low): 1

Disk request order: 5 3 2 1
Total head movement: 5

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





main.c



Share

Run

Output

Clear

```
1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <sys/stat.h>
4 #include <unistd.h>
5
6 int main() {
7     const char *filename = "example.txt";
8
9     struct stat fileStat;
10
11     // Get the file status
12     if (stat(filename, &fileStat) < 0) {
13         perror("stat");
14         return 1;
15     }
16
17     // Display current permissions
18     printf("File: %s\n", filename);
19     printf("Permissions: ");
20     printf( (S_IRUSR & fileStat.st mode) ? "r" : "-");
```

stat: No such file or directory

=== Code Exited With Errors ===

