1.Multithreaading:

```c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <pthread.h>
4
5  void* print_message(void* thread_id) {
6      long tid = (long)thread_id;
7      printf("Hello from thread %ld\n", tid);
8      pthread_exit(NULL);
9  }
10
11 int main() {
12     pthread_t threads[2];
13     int rc;
14     long t;
15
16     for (t = 0; t < 2; t++) {
17         printf("Creating thread %ld\n", t);
18         rc = pthread_create(&threads[t], NULL, print_message, (void
                *)t);
19         if (rc) {
20             printf("Error:unable to create thread, %d\n", rc);
21             exit(-1);
22         }
23     }
24
25     for (t = 0; t < 2; t++) {
```

Output:
```
Creating thread 0
Creating thread 1
Hello from thread 0
Hello from thread 1
Main thread exiting.

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

```c
23     }
24
25     for (t = 0; t < 2; t++) {
26         pthread_join(threads[t], NULL);
27     }
28
29     printf("Main thread exiting.\n");
30     return 0;
31 }
32
```

2. FIFO Paging:

```c
#include <stdio.h>

#define MAX 100

void fifoPageReplacement(int pages[], int n, int capacity) {
    int frames[capacity];
    int front = 0, rear = 0, count = 0;
    int pageFaults = 0;
    int i, j, found;

    for (i = 0; i < n; i++) {
        found = 0;

        // Check if page is already in a frame
        for (j = 0; j < count; j++) {
            if (frames[j] == pages[i]) {
                found = 1;
                break;
            }
        }

        if (!found) {
            // If frames not full, add page at rear
            if (count < capacity) {
                frames[rear] = pages[i];
                rear = (rear + 1) % capacity;
                count++;
            } else {
                // Replace the oldest page (FIFO)
                frames[front] = pages[i];
                front = (front + 1) % capacity;
                rear = (rear + 1) % capacity;
            }
            pageFaults++;
        }

        // Display current frame state
        printf("Step %d: ", i + 1);
        for (j = 0; j < count; j++) {
            printf("%d ", frames[(front + j) % capacity]);
        }
        if (found)
            printf(" (No Page Fault)");
        else
            printf(" (Page Fault)");
        printf("\n");
    }

    printf("\nTotal Page Faults: %d\n", pageFaults);
}

int main() {
    int pages[MAX], n, capacity, i;

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

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

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

    fifoPageReplacement(pages, n, capacity);

    return 0;
}
```

Output:

```
Enter number of pages: 12
Enter the page reference string:
1 3 0 3 5 6 3 0 1 2 1 2
Enter number of frames: 3
Step 1: 1   (Page Fault)
Step 2: 1 3   (Page Fault)
Step 3: 1 3 0   (Page Fault)
Step 4: 1 3 0   (No Page Fault)
Step 5: 3 0 5   (Page Fault)
Step 6: 0 5 6   (Page Fault)
Step 7: 5 6 3   (Page Fault)
Step 8: 6 3 0   (Page Fault)
Step 9: 3 0 1   (Page Fault)
Step 10: 0 1 2   (Page Fault)
Step 11: 0 1 2   (No Page Fault)
Step 12: 0 1 2   (No Page Fault)

Total Page Faults: 9

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

3. LRU Paging:

```c
1   #include <stdio.h>
2
3   #define MAX 100
4
5   // Function to find the least recently used page
6   int findLRU(int time[], int n) {
7       int i, minimum = time[0], pos = 0;
8       for (i = 1; i < n; i++) {
9           if (time[i] < minimum) {
10              minimum = time[i];
11              pos = i;
12          }
13      }
14      return pos;
15  }
16
17  void lruPageReplacement(int pages[], int n, int capacity) {
18      int frames[capacity], time[capacity];
19      int pageFaults = 0, counter = 0;
20      int i, j, pos, flag1, flag2;
21
22      for (i = 0; i < capacity; i++) {
23          frames[i] = -1;
24      }
25
26      for (i = 0; i < n; i++) {
```

Output:
```
Enter number of pages: 12
Enter the page reference string:
1 3 0 3 5 6 3 0 1 2 1 2
Enter number of frames: 3
Step 1: 1  (Page Fault)
Step 2: 1 3  (Page Fault)
Step 3: 1 3 0  (Page Fault)
Step 4: 1 3 0  (No Page Fault)
Step 5: 5 3 0  (Page Fault)
Step 6: 5 3 6  (Page Fault)
Step 7: 5 3 6  (No Page Fault)
Step 8: 0 3 6  (Page Fault)
Step 9: 0 3 1  (Page Fault)
Step 10: 0 2 1  (Page Fault)
Step 11: 0 2 1  (No Page Fault)
Step 12: 0 2 1  (No Page Fault)

Total Page Faults: 8

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

```c
26      for (i = 0; i < n; i++) {
27          flag1 = flag2 = 0;
28
29          // Check if page is already in frame
30          for (j = 0; j < capacity; j++) {
31              if (frames[j] == pages[i]) {
32                  counter++;
33                  time[j] = counter;
34                  flag1 = flag2 = 1;
35                  break;
36              }
37          }
38
39          // If page not found, insert it
40          if (flag1 == 0) {
41              for (j = 0; j < capacity; j++) {
42                  if (frames[j] == -1) {
43                      counter++;
44                      pageFaults++;
45                      frames[j] = pages[i];
46                      time[j] = counter;
47                      flag2 = 1;
48                      break;
49                  }
50              }
```

Output:
```
Enter number of pages: 12
Enter the page reference string:
1 3 0 3 5 6 3 0 1 2 1 2
Enter number of frames: 3
Step 1: 1  (Page Fault)
Step 2: 1 3  (Page Fault)
Step 3: 1 3 0  (Page Fault)
Step 4: 1 3 0  (No Page Fault)
Step 5: 5 3 0  (Page Fault)
Step 6: 5 3 6  (Page Fault)
Step 7: 5 3 6  (No Page Fault)
Step 8: 0 3 6  (Page Fault)
Step 9: 0 3 1  (Page Fault)
Step 10: 0 2 1  (Page Fault)
Step 11: 0 2 1  (No Page Fault)
Step 12: 0 2 1  (No Page Fault)

Total Page Faults: 8

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

```c
68              printf(" (Page Fault)");
69          printf("\n");
70      }
71
72      printf("\nTotal Page Faults: %d\n", pageFaults);
73  }
74
75  int main() {
76      int pages[MAX], n, capacity, i;
77
78      printf("Enter number of pages: ");
79      scanf("%d", &n);
80
81      printf("Enter the page reference string:\n");
82      for (i = 0; i < n; i++) {
83          scanf("%d", &pages[i]);
84      }
85
86      printf("Enter number of frames: ");
87      scanf("%d", &capacity);
88
89      lruPageReplacement(pages, n, capacity);
90
91      return 0;
92  }
```

Output:
```
Enter number of pages: 12
Enter the page reference string:
1 3 0 3 5 6 3 0 1 2 1 2
Enter number of frames: 3
Step 1: 1  (Page Fault)
Step 2: 1 3  (Page Fault)
Step 3: 1 3 0  (Page Fault)
Step 4: 1 3 0  (No Page Fault)
Step 5: 5 3 0  (Page Fault)
Step 6: 5 3 6  (Page Fault)
Step 7: 5 3 6  (No Page Fault)
Step 8: 0 3 6  (Page Fault)
Step 9: 0 3 1  (Page Fault)
Step 10: 0 2 1  (Page Fault)
Step 11: 0 2 1  (No Page Fault)
Step 12: 0 2 1  (No Page Fault)

Total Page Faults: 8

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

4. Optimal Paging:

```c
1  #include <stdio.h>
2  #define MAX 100
3  int predict(int pages[], int frames[], int n, int index, 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                 break;
14             }
15         }
16         if (j == n) {
17             return i;
18         }
19     }
20     return (res == -1) ? 0 : res;
21 }
22
23 void optimalPageReplacement(int pages[], int n, int capacity) {
24     int frames[capacity];
25     int count = 0, pageFaults = 0;
```

Output

```c
25     int count = 0, pageFaults = 0;
26     int i, j, k, flag;
27     for (i = 0; i < capacity; i++) {
28         frames[i] = -1;
29     }
30     for (i = 0; i < n; i++) {
31         flag = 0;
32         for (j = 0; j < count; j++) {
33             if (frames[j] == pages[i]) {
34                 flag = 1;
35                 break;
36             }
37         }
38
39         if (!flag) {
40             if (count < capacity) {
41                 frames[count++] = pages[i];
42             } else {
43                 int pos = predict(pages, frames, n, i + 1, capacity
                       );
44                 frames[pos] = pages[i];
45             }
46             pageFaults++;
47         }
48         printf("Step %d: ", i + 1);
```
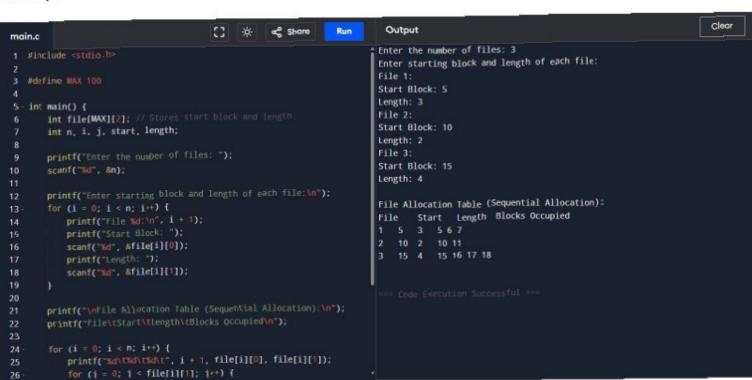
Output

```c
51        }
52        if (!flag)
53            printf(" (Page Fault)");
54        else
55            printf(" (No Page Fault)");
56        printf("\n");
57    }
58
59    printf("\nTotal Page Faults: %d\n", pageFaults);
60 }
61
62 int main() {
63    int pages[MAX], n, capacity, i;
64
65    printf("Enter number of pages: ");
66    scanf("%d", &n);
67
68    printf("Enter the page reference string:\n");
69    for (i = 0; i < n; i++) {
70        scanf("%d", &pages[i]);
71    }
72
73    printf("Enter number of frames: ");
74    scanf("%d", &capacity);
75
76    optimalPageReplacement(pages, n, capacity);
```

```
Enter number of pages: 12
Enter the page reference string:
1 3 0 3 5 6 3 0 1 2 1 2
Enter number of frames: 3
Step 1: 1     (Page Fault)
Step 2: 1 3   (Page Fault)
Step 3: 1 3 0 (Page Fault)
Step 4: 1 3 0 (No Page Fault)
Step 5: 5 3 0 (Page Fault)
Step 6: 6 3 0 (Page Fault)
Step 7: 6 3 0 (No Page Fault)
Step 8: 6 3 0 (No Page Fault)
Step 9: 1 3 0 (Page Fault)
Step 10: 1 2 0 (Page Fault)
Step 11: 1 2 0 (No Page Fault)
Step 12: 1 2 0 (No Page Fault)

Total Page Faults: 7

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

5. Sequential File Allocation:

```c
1 #include <stdio.h>
2
3 #define MAX 100
4
5 int main() {
6    int file[MAX][2]; // Stores start block and length
7    int n, i, j, start, length;
8
9    printf("Enter the number of files: ");
10   scanf("%d", &n);
11
12   printf("Enter starting block and length of each file:\n");
13   for (i = 0; i < n; i++) {
14       printf("File %d:\n", i + 1);
15       printf("Start Block: ");
16       scanf("%d", &file[i][0]);
17       printf("Length: ");
18       scanf("%d", &file[i][1]);
19   }
20
21   printf("\nFile Allocation Table (Sequential Allocation):\n");
22   printf("File\tStart\tLength\tBlocks Occupied\n");
23
24   for (i = 0; i < n; i++) {
25       printf("%d\t%d\t%d\t", i + 1, file[i][0], file[i][1]);
26       for (j = 0; j < file[i][1]; j++) {
```

```
Enter the number of files: 3
Enter starting block and length of each file:
File 1:
Start Block: 5
Length: 3
File 2:
Start Block: 10
Length: 2
File 3:
Start Block: 15
Length: 4

File Allocation Table (Sequential Allocation):
File    Start   Length  Blocks Occupied
1   5   3   5 6 7
2   10  2   10 11
3   15  4   15 16 17 18

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

```c
26          for (j = 0; j < file[i][1]; j++) {
27              printf("%d ", file[i][0] + j);
28          }
29          printf("\n");
30      }
31
32      return 0;
33 }
34
35
```

```
51          }
52          if (!flag)
53              printf(" (Page Fault)");
54          else
55              printf(" (No Page Fault)");
56          printf("\n");
57      }
58
59      printf("\nTotal Page Faults: %d\n", pageFaults);
60  }
61
62  int main() {
63      int pages[MAX], n, capacity, i;
64
65      printf("Enter number of pages: ");
66      scanf("%d", &n);
67
68      printf("Enter the page reference string:\n");
69      for (i = 0; i < n; i++) {
70          scanf("%d", &pages[i]);
71      }
72
73      printf("Enter number of frames: ");
74      scanf("%d", &capacity);
75
76      optimalPageReplacement(pages, n, capacity);
```

```
Enter number of pages: 12
Enter the page reference string:
1 3 0 3 5 6 3 0 1 2 1 2
Enter number of frames: 3
Step 1: 1  (Page Fault)
Step 2: 1 3  (Page Fault)
Step 3: 1 3 0  (Page Fault)
Step 4: 1 3 0  (No Page Fault)
Step 5: 5 3 0  (Page Fault)
Step 6: 6 3 0  (Page Fault)
Step 7: 6 3 0  (No Page Fault)
Step 8: 6 3 0  (No Page Fault)
Step 9: 1 3 0  (Page Fault)
Step 10: 1 2 0  (Page Fault)
Step 11: 1 2 0  (No Page Fault)
Step 12: 1 2 0  (No Page Fault)

Total Page Faults: 7

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

5. Sequential File Allocation:

```c
1  #include <stdio.h>
2
3  #define MAX 100
4
5  int main() {
6      int file[MAX][2]; // Stores start block and length
7      int n, i, j, start, length;
8
9      printf("Enter the number of files: ");
10     scanf("%d", &n);
11
12     printf("Enter starting block and length of each file:\n");
13     for (i = 0; i < n; i++) {
14         printf("File %d:\n", i + 1);
15         printf("Start Block: ");
16         scanf("%d", &file[i][0]);
17         printf("Length: ");
18         scanf("%d", &file[i][1]);
19     }
20
21     printf("\nFile Allocation Table (Sequential Allocation):\n");
22     printf("File\tStart\tLength\tBlocks Occupied\n");
23
24     for (i = 0; i < n; i++) {
25         printf("%d\t%d\t%d\t", i + 1, file[i][0], file[i][1]);
26         for (j = 0; j < file[i][1]; j++) {
```

```
Enter the number of files: 3
Enter starting block and length of each file:
File 1:
Start Block: 5
Length: 3
File 2:
Start Block: 10
Length: 2
File 3:
Start Block: 15
Length: 4

File Allocation Table (Sequential Allocation):
File    Start   Length  Blocks Occupied
1    5    3    5 6 7
2    10   2    10 11
3    15   4    15 16 17 18

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

```c
26              for (j = 0; j < file[i][1]; j++) {
27                  printf("%d ", file[i][0] + j);
28              }
29              printf("\n");
30          }
31
32          return 0;
33      }
34
35
```