c) Optimal

AIM:

To implement the Optimal Page Replacement Algorithm in C and analyze its efficiency in reducing page faults.

PROBLEM DESCRIPTION:

In operating systems, memory management is crucial, especially when the number of pages in a program exceeds the available frames in memory. Page replacement algorithms are used to determine which page to remove from memory when a new page is needed. The Optimal Page Replacement Algorithm (OPT) is theoretically the best approach, as it replaces the page that will not be used for the longest period in the future. Although real-time implementation is impractical due to the need for future knowledge, this simulation allows us to analyze how OPT performs under known conditions.

HEADER FILES USED

```
#include <stdio.h>   // For standard input and output functions
#include <limits.h>  // For defining maximum integer value
#include <stdbool.h> // For using boolean data type
```

ALGORITHM:

The steps of the Optimal Page Replacement Algorithm are as follows:

1. **Initialize Frames**: Start with all frames empty.
2. **Process Each Page**:
    - For each page in the reference string, check if it is already in memory.
    - If it's already in memory, continue to the next page (no page fault).
3. **Handle Page Faults**:
    - If the page is not in memory (page fault occurs), check if there is an empty frame. If so, place the page in the empty frame.
    - If all frames are occupied, determine which page to replace. For each page currently in memory, find out when it will be used next.
    - Replace the page that is not needed for the longest period in the future.
4. **Output the Frames**: After processing each page, output the current state of the frames.
5. **Calculate Page Faults**: Track the total number of page faults.

PROGRAM CODE:

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

void optimalPageReplacement(int pages[], int num_pages, int num_frames) {
    int frames[num_frames];
    int page_faults = 0;

    // Initialize frames as empty
    for (int i = 0; i < num_frames; i++) {
        frames[i] = -1;
    }

    // Traverse each page in the reference string
    for (int i = 0; i < num_pages; i++) {
        int page = pages[i];
        bool found = false;

        // Check if page is already in any frame
        for (int j = 0; j < num_frames; j++) {
            if (frames[j] == page) {
                found = true;
                break;
```

```
    }
}

// If page is not found, a page fault occurs
if (!found) {
    page_faults++;
    int replace_index = -1;
    int farthest = i + 1;

    // Look for an empty frame if possible
    for (int j = 0; j < num_frames; j++) {
        if (frames[j] == -1) {
            replace_index = j;
            break;
        }
    }

    // If no empty frame, find the page to replace
    if (replace_index == -1) {
        for (int j = 0; j < num_frames; j++) {
            int next_use = INT_MAX;
            for (int k = i + 1; k < num_pages; k++) {
                if (pages[k] == frames[j]) {
                    next_use = k;
```

```
                    break;
                }
            }
            if (next_use > farthest) {
                farthest = next_use;
                replace_index = j;
            }
        }
    }

    // Replace the page in the identified frame
    frames[replace_index] = page;
  }

  // Print the current state of frames
  printf("Page %d -> Frames: ", page);
  for (int j = 0; j < num_frames; j++) {
    if (frames[j] == -1) printf(" - ");
    else printf("%d ", frames[j]);
  }
  printf("\n");
}

printf("\nTotal Page Faults: %d\n", page_faults);
```

```
}

int main() {
    int num_pages, num_frames;

    printf("Enter the number of pages: ");
    scanf("%d", &num_pages);
    int pages[num_pages];

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

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

    printf("\nOptimal Page Replacement Simulation:\n");
    optimalPageReplacement(pages, num_pages, num_frames);

    return 0;
}
```

SAMPLE INPUT AND OUTPUT:

Sample Input

Enter the number of pages: 7

Enter the page reference string:

0 1 2 3 2 1 4

Enter the number of frames: 3

Sample Output

Optimal Page Replacement Simulation:

Page 0 -> Frames: 0 - -

Page 1 -> Frames: 0 1 -

Page 2 -> Frames: 0 1 2

Page 3 -> Frames: 3 1 2

Page 2 -> Frames: 3 1 2

Page 1 -> Frames: 3 1 2

Page 4 -> Frames: 4 1 2

Total Page Faults: 5

RESULT:

Thus the optimal page replacement algorithm has been implemented and executed successfully.