This C program simulates and compares **local and global FIFO page replacement policies** for two processes. It first generates a **random page reference string** using a probabilistic model, where each page has a chance to repeat before switching to another. The program then implements **local page replacement**, where each process has its own fixed number of page frames, and **global page replacement**, where both processes share a combined frame pool. The FIFO (First-In-First-Out) algorithm is used to track page faults in both cases. Finally, the program prints the generated reference string and the number of page faults for each approach.

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>


#define N 5 // Number of unique pages

#define PAGE_FRAMES 3 // Number of frames available per process

#define REFERENCES 20 // Length of the page reference string


// Function to generate a page reference string with more variation

void generate_reference_string(int *reference_string, double *probabilities) {

  int current_page = rand() % N; // Start with a random page

  for (int i = 0; i < REFERENCES; i++) {

    reference_string[i] = current_page;

    if ((double)rand() / RAND_MAX > probabilities[current_page]) { // Higher chance to
switch pages

      current_page = rand() % N; // Completely random new page

    }

  }
```

```
}

// Function to simulate FIFO page replacement and count page faults
int fifo_page_replacement(int *reference_string, int frames) {
    int page_faults = 0, index = 0;
    int page_table[frames];
    bool in_memory[N] = {false};

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

    for (int i = 0; i < REFERENCES; i++) {
        int page = reference_string[i];

        bool found = false;
        for (int j = 0; j < frames; j++) {
            if (page_table[j] == page) {
                found = true;
                break;
            }
        }

        if (!found) { // Page fault occurs
            page_faults++;
            in_memory[page_table[index]] = false; // Remove old page
            page_table[index] = page;
            in_memory[page] = true;
```

```
        index = (index + 1) % frames; // Move FIFO index
      }
    }
    return page_faults;
}

// Function to simulate global FIFO page replacement correctly
int global_fifo_replacement(int *reference_string, int total_frames) {
    int page_faults = 0, index = 0;
    int page_table[total_frames];
    bool in_memory[N] = {false};

    for (int i = 0; i < total_frames; i++) page_table[i] = -1;

    for (int i = 0; i < REFERENCES; i++) {
      int page = reference_string[i];

      bool found = false;
      for (int j = 0; j < total_frames; j++) {
        if (page_table[j] == page) {
          found = true;
          break;
        }
      }

      if (!found) { // Page fault occurs
```

```
        page_faults++;
        in_memory[page_table[index]] = false; // Remove old page
        page_table[index] = page;
        in_memory[page] = true;
        index = (index + 1) % total_frames; // Move FIFO index
      }
    }
    return page_faults;
}

int main() {
    srand(time(NULL));

    double probabilities[N] = {0.7, 0.6, 0.6, 0.5, 0.5}; // Adjusted to create more variation
    int reference_string[REFERENCES];
    generate_reference_string(reference_string, probabilities);

    printf("Generated Page Reference String: \n");
    for (int i = 0; i < REFERENCES; i++) {
      printf("%d ", reference_string[i]);
    }
    printf("\n\n");

    // Simulate local page replacement
    printf("Local Page Replacement (Separate page tables for each process):\n");
    int process1_faults = fifo_page_replacement(reference_string, PAGE_FRAMES);
```

```
    int process2_faults = fifo_page_replacement(reference_string, PAGE_FRAMES);
    printf("Process 1 Page Faults: %d\n", process1_faults);
    printf("Process 2 Page Faults: %d\n\n", process2_faults);

    // Simulate global page replacement with properly shared frames
    printf("Global Page Replacement (Shared page table):\n");
    int global_faults = global_fifo_replacement(reference_string, PAGE_FRAMES * 2);
    printf("Total Page Faults: %d\n", global_faults);

    return 0;
}
```