Exp 7: Simulate all page replacement algorithms

a) FIFO b) LRU c) LFU

a) FIFO (First In First Out)

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

// Function to check if a page is already in memory
bool isInMemory(int page, int frames[], int frameCount) {
    for (int i = 0; i < frameCount; i++) {
        if (frames[i] == page)
            return true;
    }
    return false;
}

// Function to display frames
void displayFrames(int frames[], int frameCount) {
    printf("[");
    for (int i = 0; i < frameCount; i++) {
        if (frames[i] == -1)
            printf(" ");
        else
            printf("%d", frames[i]);

        if (i < frameCount - 1)
            printf("|");
    }
    printf("]");
}

int main() {
    int *referenceString;
    int frameSize, referenceSize;

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

    printf("Enter the size of the reference string: ");
    scanf("%d", &referenceSize);

    referenceString = (int *)malloc(referenceSize * sizeof(int));

    if (referenceString == NULL) {
        printf("Memory allocation failed.\n");
    return -1;
}

printf("Enter the reference string: ");
for (int i = 0; i < referenceSize; i++) {
    scanf("%d", &referenceString[i]);
}

int frames[frameSize]; // Frames in memory
int frameCount = 0;     // Number of frames currently in use

// Initialize frames to -1 indicating empty frame
for (int i = 0; i < frameSize; i++) {
    frames[i] = -1;
}

int pageFaults = 0;

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

printf("Frame\t Pages\t   Page Fault\t Frames\n");

// Simulating FIFO page replacement
for (int i = 0; i < referenceSize; i++) {
    printf("%d\t   %d\t\t", i + 1, referenceString[i]);

    if (!isInMemory(referenceString[i], frames, frameCount)) {
        if (frameCount < frameSize) {
            frames[frameCount++] = referenceString[i];
        } else {
            for (int j = 0; j < frameSize - 1; j++) {
                frames[j] = frames[j + 1];
            }
            frames[frameSize - 1] = referenceString[i];
        }
        pageFaults++;
        printf("Yes\t\t");
    } else {
        printf("No\t\t");
    }
    displayFrames(frames, frameCount);
    printf("\n");
    }

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

    free(referenceString);

    return 0;
}
```

Test Case:

```
Enter the number of frames: 3
Enter the size of the reference string: 15
Enter the reference string: 7  0  1  2  0  3  0  4  2  3  0  3  1  2  0
Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 1 2 0
Frame       Pages       Page Fault        Frames
1           7              Yes            [7]
2           0              Yes            [7|0]
3           1              Yes            [7|0|1]
4           2              Yes            [0|1|2]
5           0              No             [0|1|2]
6           3              Yes            [1|2|3]
7           0              Yes            [2|3|0]
8           4              Yes            [3|0|4]
9           2              Yes            [0|4|2]
10          3              Yes            [4|2|3]
11          0              Yes            [2|3|0]
12          3              No             [2|3|0]
13          1              Yes            [3|0|1]
14          2              Yes            [0|1|2]
15          0              No             [0|1|2]

Total Page Faults: 12
```

b) LRU (Least Recently Used)

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

// Function to check if a page is already in memory
bool isInMemory(int page, int frames[], int frameCount) {
    for (int i = 0; i < frameCount; i++) {
        if (frames[i] == page)
            return true;
    }
    return false;
}

// Function to display frames
void displayFrames(int frames[], int frameCount) {
    printf("[");
    for (int i = 0; i < frameCount; i++) {
        if (frames[i] == -1)
            printf(" ");
        else
            printf("%d", frames[i]);

        if (i < frameCount - 1)
            printf("|");
    }
    printf("]");
}

// Function to find the index of the least recently used page
int findLRUIndex(int pages[], int frameCount, int referenceString[], int referenceSize, int
currentIndex) {
    int index = -1, farthest = currentIndex;
    for (int i = 0; i < frameCount; i++) {
        int j;
        for (j = currentIndex - 1; j >= 0; j--) {
            if (pages[i] == referenceString[j]) {
                if (j < farthest) {
                    farthest = j;
                    index = i;
                }
                break;
            }
        }
        if (j == -1)
            return i;
    }
    return (index == -1) ? 0 : index;
}

int main() {
    int *referenceString;
    int frameSize, referenceSize;

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

```
printf("Enter the size of the reference string: ");
scanf("%d", &referenceSize);

referenceString = (int *)malloc(referenceSize * sizeof(int));

if (referenceString == NULL) {
   printf("Memory allocation failed.\n");
   return -1;
}

printf("Enter the reference string: ");
for (int i = 0; i < referenceSize; i++) {
   scanf("%d", &referenceString[i]);
}

int frames[frameSize]; // Frames in memory
int frameCount = 0;     // Number of frames currently in use

// Initialize frames to -1 indicating empty frame
for (int i = 0; i < frameSize; i++) {
   frames[i] = -1;
}

int pageFaults = 0;

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

printf("Frame\t Pages\t   Page Fault\t Frames\n");

// Simulating LRU page replacement
for (int i = 0; i < referenceSize; i++) {
   printf("%d\t   %d\t\t", i + 1, referenceString[i]);

   if (!isInMemory(referenceString[i], frames, frameCount)) {
      if (frameCount < frameSize) {
         frames[frameCount++] = referenceString[i];
      } else {
         int index = findLRUIndex(frames, frameCount, referenceString, referenceSize, i);
         frames[index] = referenceString[i];
      }
      pageFaults++;
      printf("Yes\t\t");
```

```
        } else {
            printf("No\t\t");
        }
        displayFrames(frames, frameCount);
        printf("\n");
    }

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

    free(referenceString);

    return 0;
}
```

Test Case:

```
Enter the number of frames: 3
Enter the size of the reference string: 15
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 1 2 0
Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 1 2 0
Frame   Pages    Page Fault     Frames
1        7           Yes            [7]
2        0           Yes            [7|0]
3        1           Yes            [7|0|1]
4        2           Yes            [2|0|1]
5        0           No             [2|0|1]
6        3           Yes            [2|0|3]
7        0           No             [2|0|3]
8        4           Yes            [4|0|3]
9        2           Yes            [4|0|2]
10       3           Yes            [4|3|2]
11       0           Yes            [0|3|2]
12       3           No             [0|3|2]
13       1           Yes            [0|3|1]
14       2           Yes            [2|3|1]
15       0           Yes            [2|0|1]

Total Page Faults: 12
```

C) LFU (Least Frequently Used)

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

// Function to check if a page is already in memory
bool isInMemory(int page, int frames[], int frameCount) {
    for (int i = 0; i < frameCount; i++) {
        if (frames[i] == page)
            return true;
    }
    return false;
}

// Function to display frames
void displayFrames(int frames[], int frameCount) {
    printf("[");
    for (int i = 0; i < frameCount; i++) {
        if (frames[i] == -1)
            printf(" ");
        else
            printf("%d", frames[i]);

        if (i < frameCount - 1)
            printf("|");
    }
    printf("]");
}

// Function to find the index of the least frequently used page
int findLFUIndex(int frames[], int frequency[], int frameCount, int referenceString[], int
referenceSize) {
    int index = 0;
    int minFrequency = frequency[frames[0]];

    for (int i = 1; i < frameCount; i++) {
        if (frequency[frames[i]] < minFrequency) {
            minFrequency = frequency[frames[i]];
            index = i;
        }
    }

    return index;
}

int main() {
    int *referenceString;
    int frameSize, referenceSize;

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

    printf("Enter the size of the reference string: ");
    scanf("%d", &referenceSize);

    referenceString = (int *)malloc(referenceSize * sizeof(int));

    if (referenceString == NULL) {
        printf("Memory allocation failed.\n");
        return -1;
    }

    printf("Enter the reference string: ");
    for (int i = 0; i < referenceSize; i++) {
        scanf("%d", &referenceString[i]);
    }

    int frames[frameSize];  // Frames in memory
    int frequency[frameSize]; // Frequency of each page in frames
    int frameCount = 0;      // Number of frames currently in use

    // Initialize frames to -1 indicating empty frame
    for (int i = 0; i < frameSize; i++) {
        frames[i] = -1;
        frequency[i] = 0;
    }

    int pageFaults = 0;

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

    printf("Frame\t Pages\t   Page Fault\t Frames\n");

    // Simulating LFU page replacement
    for (int i = 0; i < referenceSize; i++) {
        printf("%d\t   %d\t\t", i + 1, referenceString[i]);

        if (!isInMemory(referenceString[i], frames, frameCount)) {
            if (frameCount < frameSize) {
                frames[frameCount++] = referenceString[i];
            } else {
                int index = findLFUIndex(frames, frequency, frameCount, referenceString,
referenceSize);
                frames[index] = referenceString[i];
                frequency[frames[index]] = 0;
            }
            pageFaults++;
            printf("Yes\t\t");
        } else {
            frequency[referenceString[i]]++;
            printf("No\t\t");
        }
        displayFrames(frames, frameCount);
        printf("\n");
    }

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

    free(referenceString);

    return 0;
}
```

Test Case:

```
Enter the number of frames: 3
Enter the size of the reference string: 15
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 1 2 0
Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 1 2 0
Frame   Pages    Page Fault    Frames
1        7             Yes            [7]
2        0             Yes            [7|0]
3        1             Yes            [7|0|1]
4        2             Yes            [2|0|1]
5        0             No             [2|0|1]
6        3             Yes            [3|0|1]
7        0             No             [3|0|1]
8        4             Yes            [0|0|1]
9        2             Yes            [0|0|2]
10       3             Yes            [0|0|3]
11       0             No             [0|0|3]
12       3             No             [0|0|3]
13       1             Yes            [0|0|1]
14       2             Yes            [0|0|2]
15       0             No             [0|0|2]

Total Page Faults: 10
```