NAME:IQRA NAWAZ
ROLL NUMBER:DT-22005

LAB 12

QUESTION: Write a C program to simulate page replacement algorithms.

a) FIFO b) LRU c) Optimal d)MRU

ANSWER:

a)FIFO

CODE:

```
#include <stdio.h>


int main() {

  int i, j, k, frameIndex = 0,

  pageFaults = 0; int

  referenceString[25], frames[10], n,

  f;


  printf("Enter the length of the reference

  string: "); scanf("%d", &n);


  printf("Enter the reference

  string: "); for (i = 0; i < n; i++)

    scanf("%d", &referenceString[i]);


  printf("Enter the number of

  frames: "); scanf("%d", &f);
```

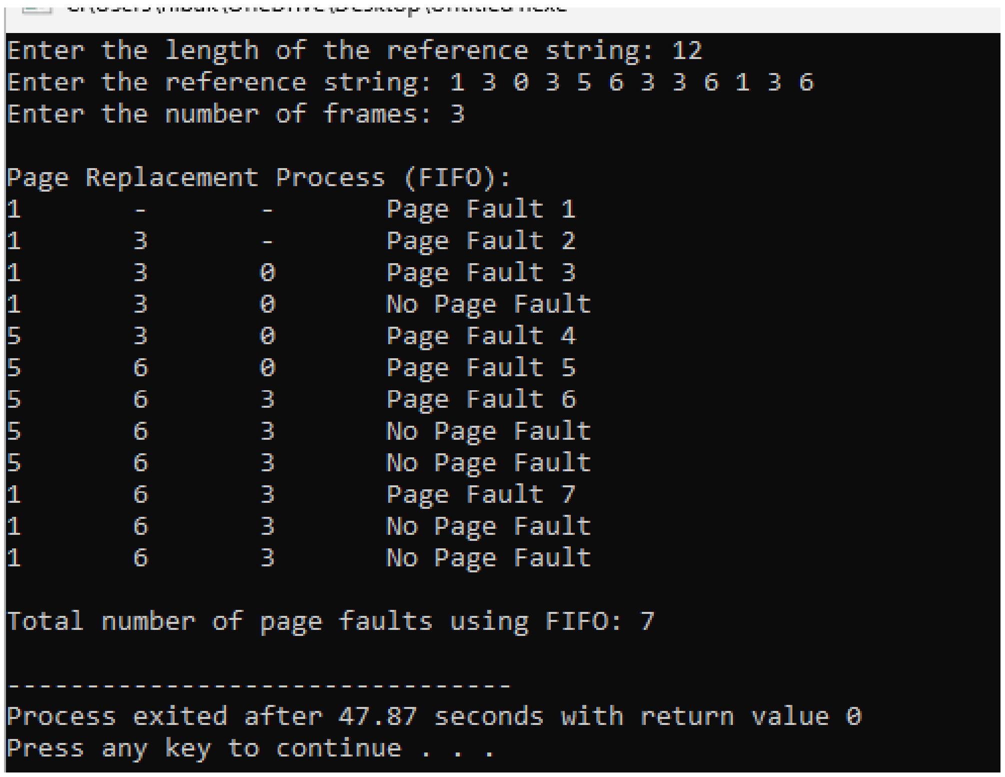
```
for (i = 0; i < f; i++)
   frames[i] = -1; // initialize all frames to -1


printf("\nPage Replacement Process (FIFO):\n");
```

```
for (i = 0; i < n; i++) {
  // Check if the page is already in a
  frame for (k = 0; k < f; k++) {
    if (frames[k] ==
      referenceString[i]) break;
  }

  // Page not found -> page
  fault if (k == f) {
    frames[frameIndex] =
    referenceString[i]; frameIndex =
    (frameIndex + 1) % f; pageFaults++;

    // Display current frame
    state for (j = 0; j < f; j++) {
      if (frames[j] != -1)
        printf("%d\t",
      frames[j]); else
        printf("-\t");
    }
    printf("Page Fault %d", pageFaults);
  } else {
      // Page hit - no
     fault for (j = 0; j <
              f; j++) { if
```

```
(frames[j] != -1)
   printf("%d\t", frames[j]);
```

```
            else
               printf("-\t");
         }
         printf("No Page Fault");
      }

      printf("\n");
   }

   printf("\nTotal number of page faults using FIFO: %d\n",
   pageFaults); return 0;
}
```

OUTPUT:



```
Enter the length of the reference string: 12
Enter the reference string: 1 3 0 3 5 6 3 3 6 1 3 6
Enter the number of frames: 3

Page Replacement Process (FIFO):
1       -       -       Page Fault 1
1       3       -       Page Fault 2
1       3       0       Page Fault 3
1       3       0       No Page Fault
5       3       0       Page Fault 4
5       6       0       Page Fault 5
5       6       3       Page Fault 6
5       6       3       No Page Fault
5       6       3       No Page Fault
1       6       3       Page Fault 7
1       6       3       No Page Fault
1       6       3       No Page Fault

Total number of page faults using FIFO: 7

--------------------------------
Process exited after 47.87 seconds with return value 0
Press any key to continue . . .
```

b) LRU

CODE:

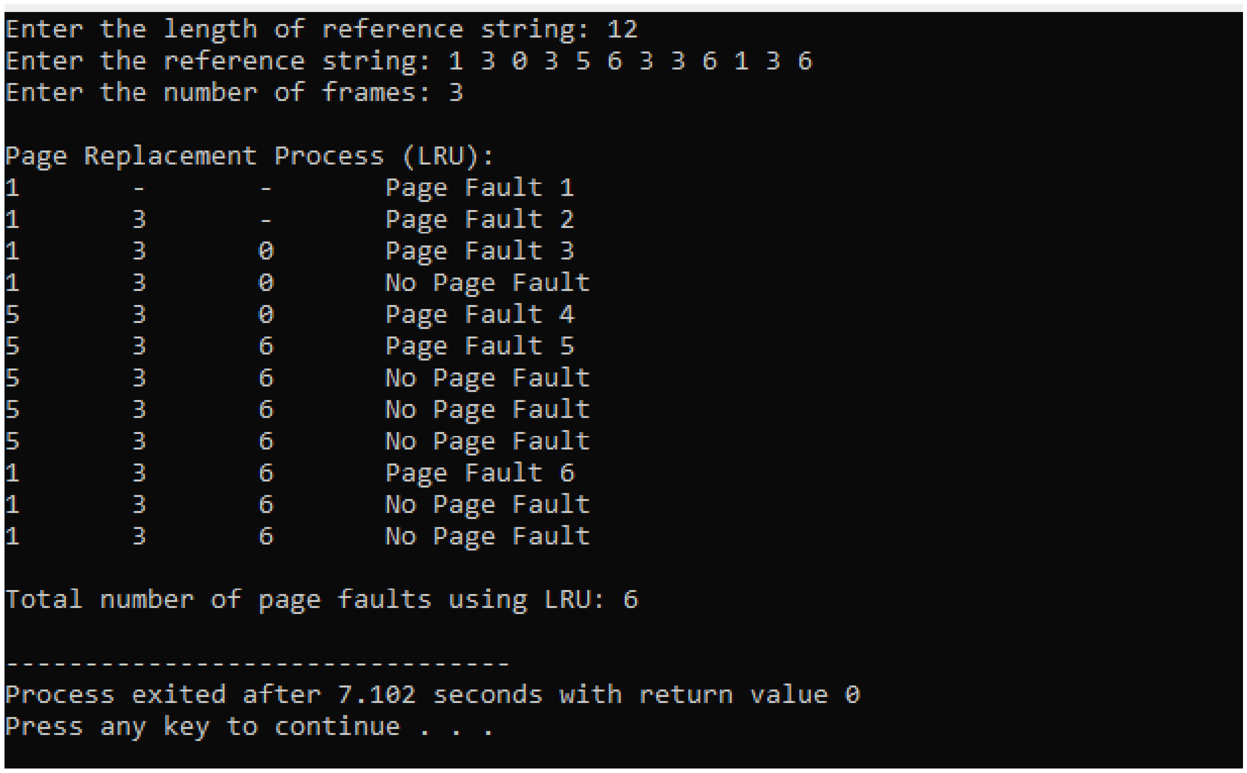
```
#include <stdio.h>


int main() {
  int i, j, k, min, n, f;
  int referenceString[25], frames[10], lastUsed[10], pageFaults =
  0, next = 1; int flag[25] = {0};


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


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


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


 for (i = 0; i < f; i++) {
    frames[i] = -1;
   lastUsed[i] = 0;
}
```

```
printf("\nPage Replacement Process (LRU):\n");

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

    for (j = 0; j < f; j++) {
      if (frames[j] ==
        referenceString[i]) { flag[i] =
        1;
        lastUsed[j] =
        next++; found = 1;
        break;
      }
    }

    if (!found) {
      if (i < f) {
        frames[i] =
        referenceString[i];
        lastUsed[i] = next++;
      } else {
        min = 0;
        for (j = 1; j < f; j++) {
          if (lastUsed[j] <
            lastUsed[min]) { min = j;
  }
}
frames[min] = referenceString[i];
```

```
        lastUsed[min] = next++;
      }
      pageFaults++;
    }

    for (j = 0; j < f;
      j++) { if
      (frames[j] !=
      -1)
        printf("%d\t",
      frames[j]); else
        printf("-\t");
    }

    if (!found)
      printf("Page Fault %d",
    pageFaults); else
      printf("No Page Fault");

    printf("\n");
  }

  printf("\nTotal number of page faults using LRU: %d\n", pageFaults);

  return 0;
```

```
}
```

OUTPUT:



```
Enter the length of reference string: 12
Enter the reference string: 1 3 0 3 5 6 3 3 6 1 3 6
Enter the number of frames: 3

Page Replacement Process (LRU):
1       -       -       Page Fault 1
1       3       -       Page Fault 2
1       3       0       Page Fault 3
1       3       0       No Page Fault
5       3       0       Page Fault 4
5       3       6       Page Fault 5
5       3       6       No Page Fault
5       3       6       No Page Fault
5       3       6       No Page Fault
1       3       6       Page Fault 6
1       3       6       No Page Fault
1       3       6       No Page Fault

Total number of page faults using LRU: 6

--------------------------------
Process exited after 7.102 seconds with return value 0
Press any key to continue . . .
```

c)Optimal

CODE:

```
#include <stdio.h>

int main() {
  int no_of_frames, no_of_pages;
  int frames[10], pages[30],
  temp[10]; int flag1, flag2,
  flag3;
  int i, j, k, pos, max, faults = 0;

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

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

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

  for (i = 0; i < no_of_frames; ++i) {
    frames[i] = -1;
}

printf("\nPage Replacement Process (Optimal):\n");
```

```
for (i = 0; i < no_of_pages;
  ++i) { flag1 = flag2 = 0;

  // Check if page is already in a
  frame for (j = 0; j <
  no_of_frames; ++j) {
    if (frames[j] ==
      pages[i]) { flag1 =
      flag2 = 1; break;
    }
  }

  // If page is not already in
  frame if (flag1 == 0) {
    // Check for empty frame
    for (j = 0; j < no_of_frames;
      ++j) { if (frames[j] == -1) {
        frames[j] = pages[i];
        faults++;
        flag2 = 1;
        break;
      }
    }
  }
```

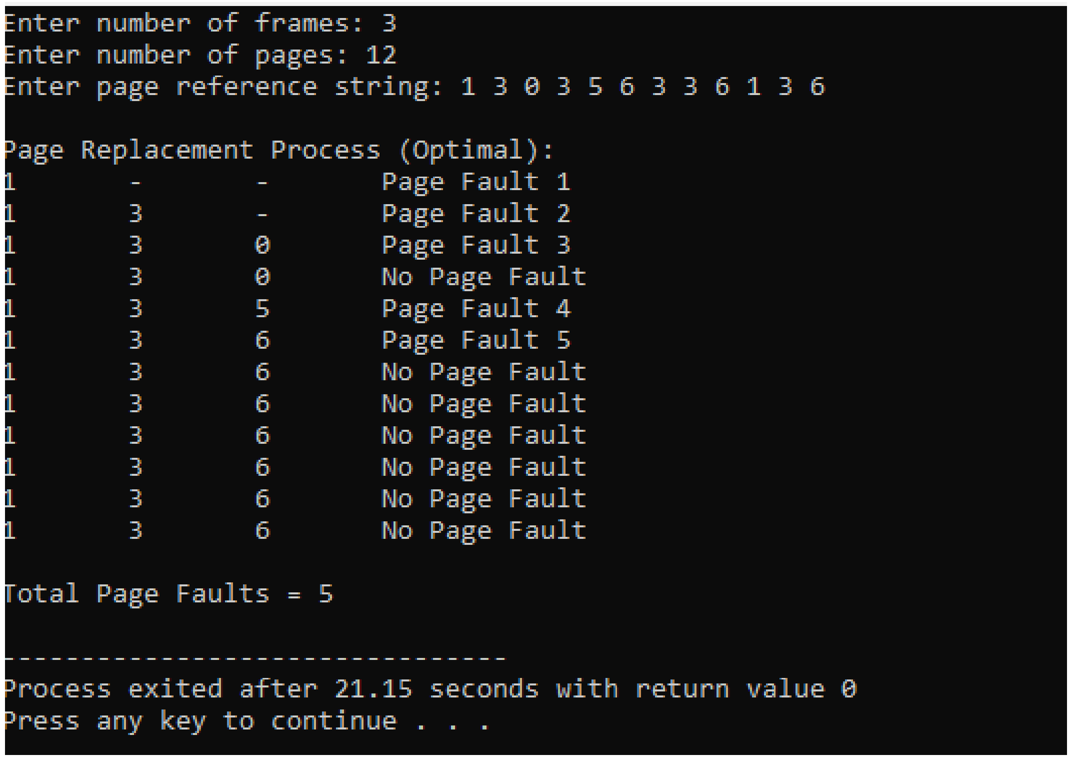
```
// If no empty frame, use optimal replacement
```

```
if (flag2 == 0) {
  flag3 = 0;

  for (j = 0; j < no_of_frames;
    ++j) { temp[j] = -1;

    for (k = i + 1; k < no_of_pages;
      ++k) { if (frames[j] ==
      pages[k]) {
        temp[j] = k;
        break;
      }
    }
  }

  for (j = 0; j < no_of_frames;
    ++j) { if (temp[j] == -1) {
      pos = j;
      flag3 = 1;
      break;
    }
  }

  if (flag3 == 0) {
    max =
    temp[0]; pos

    = 0;

    for (j = 1; j < no_of_frames; ++j) {
```

```
            if (temp[j] >
               max) { max =
               temp[j]; pos =
               j;
            }
         }
      }

      frames[pos] = pages[i];
      faults++;
   }

   // Print current state of
   frames for (j = 0; j <
   no_of_frames; ++j) {
      if (frames[j] != -1)
         printf("%d\t",
      frames[j]); else
         printf("-\t");
   }
   if (!flag1) printf("Page Fault %d",
   faults); else printf("No Page
   Fault"); printf("\n");
}
```

```
    printf("\nTotal Page Faults = %d\n",

    faults); return 0;

}
```

OUTPUT:



```
Enter number of frames: 3
Enter number of pages: 12
Enter page reference string: 1 3 0 3 5 6 3 3 6 1 3 6

Page Replacement Process (Optimal):
1       -       -       Page Fault 1
1       3       -       Page Fault 2
1       3       0       Page Fault 3
1       3       0       No Page Fault
1       3       5       Page Fault 4
1       3       6       Page Fault 5
1       3       6       No Page Fault
1       3       6       No Page Fault
1       3       6       No Page Fault
1       3       6       No Page Fault
1       3       6       No Page Fault
1       3       6       No Page Fault

Total Page Faults = 5

--------------------------------
Process exited after 21.15 seconds with return value 0
Press any key to continue . . .
```

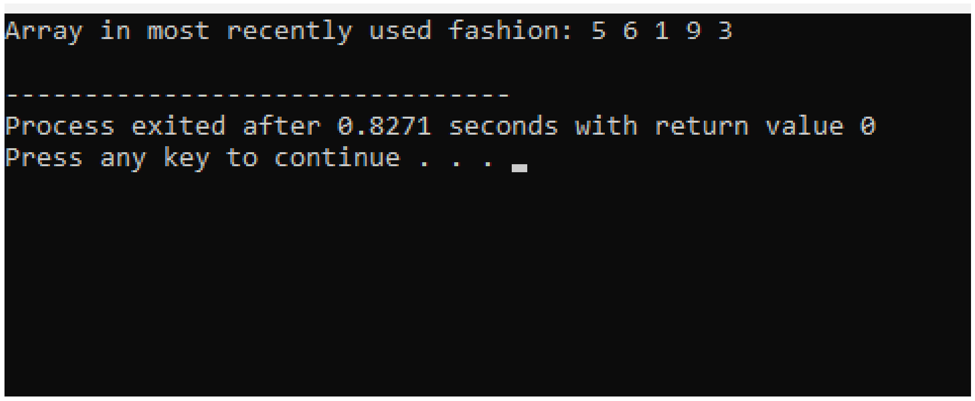
d)MRU

CODE:

```
#include <iostream>
using namespace
std;

// Function to update the array in most recently used (MRU)
fashion void recently(int* arr, int size, int elem) {
   int index = elem % size; // Find index using
   modulo int id = arr[index];          // Get the
   value at the index

   // Shift elements from index to
   front while (index > 0) {
      arr[index] = arr[index
      - 1]; index--;
   }

   // Place the accessed element at the
   front arr[0] = id;
}

// Function to print array
elements void print(int* arr,
int size) {
  for (int i = 0; i < size;
    i++) cout << arr[i] << "
    ";
  cout << endl;
}
```

```
int main() {
  int elem = 3;
  int arr[] = {6, 1, 9, 5, 3};
  int size = sizeof(arr) / sizeof(arr[0]);

  recently(arr, size, elem);

  cout << "Array in most recently used
  fashion: "; print(arr, size);

  return 0;
}
```

OUTPUT:



```
Array in most recently used fashion: 5 6 1 9 3

--------------------------------
Process exited after 0.8271 seconds with return value 0
Press any key to continue . . .
```