UCS1411 Operating Systems Lab
AY: 2021-22

Name: Krithika Swaminathan
Roll No.: 205001057

Assignment 10 – Page Replacement Techniques: FIFO, Optimal, LRU, LFU

Date: **23/05/2022**

Aim:

To develop a C program to implement the page replacement algorithms (FIFO, Optimal, LRU and LFU) using linked list.

Algorithm:

1. Start
2. Read the number of frames.
3. Read the number of frames required by the process N.
4. Read the reference string for allocation of page frames.
5. Implement the chosen page replacement algorithm.
6. Stop

Page replacement algorithms:

1. *FIFO replacement:*
    1. Allocate the first N pages into the frames and increment the page faults accordingly.
    2. When the next frame in the reference string is not already available in the allocated list do:
        1. Look for the oldest one in the allocated frames and replace it with the next page frame.
        2. Increment the page fault whenever a frame is replaced.
    3. Display the allocated frame list after every replacement.

2. *Optimal replacement:*
    1. Allocate the first N pages into the frames and increment the page faults accordingly.
    2. When the next frame in the reference string is not already available in the allocated list do:
        1. Look for a frame in the reference string that will not be used for the longest period of time.
        2. Increment the page fault whenever a frame is replaced. (Hint: Locate the position of each allocated frame in the reference string; identify a frame for replacement with the largest index position)
    3. Display the allocated frame list after every replacement.

Name: Krithika Swaminathan
Roll No.: 205001057

3. *LRU replacement:*
    1. Allocate the first N pages into the frames and increment the page faults accordingly.
    2. When the next frame in the reference string is not already available in the allocated list do:
        1. Look for a frame which has not been used recently.
        2. Increment the page fault whenever a frame is replaced.
    3. Display the allocated frame list after every replacement.

4. *LFU replacement:*
    1. Allocate the first N pages into the frames and increment the page faults accordingly.
    2. When the next frame in the reference string is not already available in the allocated list do:
        1. Look for a frame which is least frequently used.
        2. Increment the page fault whenever a frame is replaced.
    3. Display the allocated frame list after every replacement.

Code:

```
//Program to implement page replacement techniques
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int order= 0;
typedef struct Node {
  int d;
  struct Node *next;
  int freq;
  int order;
} node;

node *createlist() {
  node *head = (node *)malloc(sizeof(node));
  head->d = 0;
  head->next = NULL;
  head->freq = 0;
  return head;
}

void insertlast(node *head, int d) {
  node *ins = (node *)malloc(sizeof(node));
  ins->d = d;
  ins->freq = 1;
  node *temp = head;
  while (temp->next != NULL) {
    temp = temp->next;
  }
  ins->next = NULL;
  temp->next = ins;
}

void insertfirst(node *head, int d) {
  node *ins = (node *)malloc(sizeof(node));
  ins->d = d;
  ins->freq = 1;
  node *temp = head->next;
  ins->next = temp;
  head->next = ins;
}

int delete (node *prev) {
  int d;
  if (!prev)
    return -1;
  if (!prev->next)
    return -1;
  node *tmp = prev->next;
  d = tmp->d;
  prev->next = prev->next->next;
  free(tmp);
  return d;
}

int deletefirst(node *head) {
  int d;
  if (head->next == NULL) {
    printf("Empty List");
    return -1;
  }
  node *temp = head->next;
  if (temp->next != NULL) {
    head->next = temp->next;
    d = temp->d;
    free(temp);
  } else {
    d = temp->d;
    head->next = NULL;
  }
  return d;
}
```

```
int deletelast(node *head) {
  node *temp = head;
  if (head->next == NULL) {
    printf("Empty List!\n");
    return -1;
  }
  while (temp->next->next != NULL) {
    temp = temp->next;
  }
  delete (temp);
}

void display(node *head) {
  node *tmp = head->next;
  if (tmp == NULL) {
    printf("Empty!\n");
    return;
  }
  while (tmp) {
    printf(" %d", tmp->d);
    tmp = tmp->next;
  }
}

node *search(node *head, int d) {
  if (head->next == NULL)
    return NULL;

  node *tmp = head;
  while (tmp->next) {
    if (tmp->next->d == d)
      return tmp;
    tmp = tmp->next;
  }

  return NULL;
}

int frequency(int *seq, int d, int start, int end) {
  int itr = 0;
  if (start == end)
    return -1;
  for (int i = 0; i < start; i++)
  {
    if (seq[i] == d)
      itr++;
  }
  return itr;
}
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
int length(node *head) {
  node *tmp = head->next;
  if (tmp == NULL)
    return 0;

  int count = 0;
  while (tmp) {
    tmp = tmp->next;
    count++;
  }
  return count;
}

void putTable(int table[10][20], int n_frames, int n_updates) {
  printf("\n ");
  for (int i = 0; i < n_updates; i++)
    printf("+----");
  printf("+\n ");

  for (int i = 0; i < n_frames; i++) {
    for (int j = 0; j < n_updates; j++) {
      if (table[i][j] == -1)
        printf("| -  ");
      else
        printf("| %-2d ", table[i][j]);
    }
    printf("|\n ");
  }
  for (int i = 0; i < n_updates; i++)
    printf("+----");
  printf("+\n ");
}

void insertTable(node *tmp, int table[10][20], int n_frames, int faults) {
  for (int i = 0; i < n_frames; i++) {
    if (tmp) {
      table[i][faults] = tmp->d;
      tmp = tmp->next;
    } else
      table[i][faults] = -1;
  }
}

void FIFO(int seq[30], int len, int n_frames) {
  int faults = 0;
  int size = 0;
  int table[10][20];
  node *pg = createlist();
  node *oldest;
  printf("\n");
  printf("   Frame ->        In Memory      -> Faults \n\n");
  for (int i = 0; i < len; i++) {
    printf("     %-2d  ->", seq[i]);
    node *found = search(pg, seq[i]);
    node *tmp;
    if (!found) {
      if (size < n_frames) {
        insertlast(pg, seq[i]);
        size++;
        if (size == 1) {
          oldest = pg->next;
        }
      }
      else {
        oldest->d = seq[i];
        if (oldest->next) {
          oldest = oldest->next;
        } else {
          oldest = pg->next;
        }
      }
      insertTable(pg->next, table, n_frames, faults);
      faults++;
    }
    display(pg);
    // check formatting
    for (int i = length(pg) * 3; i <= 22; i++)
      printf(" ");
    printf("->    %-2d   \n", faults);
  }
  putTable(table, n_frames, faults);
}

void optimal(int seq[30], int len, int n_frames) {
  int size = 0;
  int faults = 0;
  int table[10][20];
  int nextocc[n_frames];
  node *pg = createlist();
  printf("   Frame ->        In Memory      -> Faults \n\n");
  for (int i = 0; i < len; i++) {
    printf("     %-2d  ->", seq[i]);
    node *found = search(pg, seq[i]);
    node *tmp;
    if (!found) {
      if (size < n_frames) {
        insertlast(pg, seq[i]);
        size++;
      }
      else {
        int distance = 0, maxd = 0, replace;
        int flag = 0;
        tmp = pg->next;
        node *change;
        while (tmp){
          flag = 0;
          for (int j = i; j<len && flag == 0; j++){
            if (seq[j] == tmp->d){
              flag = 1;
              distance = j-i;
              if (distance > maxd){
                maxd = distance;
                replace = seq[j];
              }
            }

            //printf("replace: %d ", replace);
          }
          if (flag == 0)
          {
            maxd = 99999;
            replace = tmp->d;
            break;
          }
          tmp = tmp->next;
        }
        change = search(pg, replace);
        change->next->d = seq[i];
      }
      insertTable(pg->next, table, n_frames, faults);
      faults++;
    }
    display(pg);
    // check formatting
    for (int i = length(pg) * 3; i <= 22; i++)
      printf(" ");
    printf("->    %-2d   \n", faults);
  }
  putTable(table, n_frames, faults);

}

void LRU(int seq[30], int len, int n_frames)
{
  int size = 0;
  int faults = 0;
```

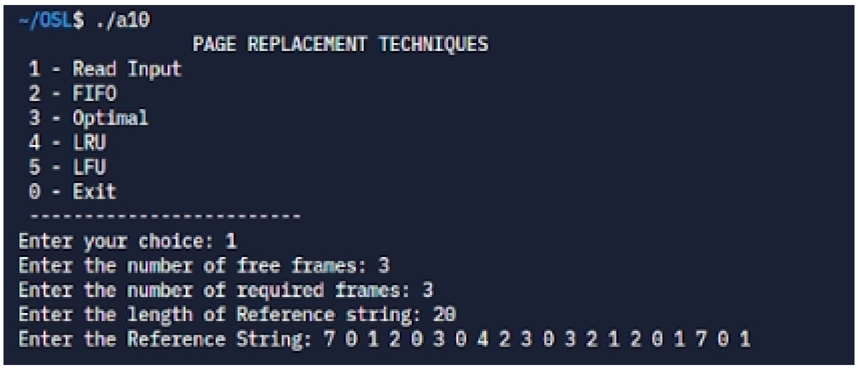
Name: Krithika Swaminathan
Roll No.: 205001057

```
int table[10][20];
int nextocc[n_frames];
node *pg = createlist();
printf("   Frame ->        In Memory      -> Faults \n\n");
for (int i = 0; i < len; i++) {
  printf("     %-2d  ->", seq[i]);
  node *found = search(pg, seq[i]);
  node *tmp;
  if (!found) {
    if (size < n_frames) {
      insertlast(pg, seq[i]);
      size++;
    }
    else {
      int distance = 0, maxd = 0, replace;
      int flag = 0;
      tmp = pg->next;
      node *change;
      while (tmp){
        flag = 0;
        for (int j = i; j>0 && flag == 0; j--){
          if (seq[j] == tmp->d){
            flag = 1;
            distance = i-j;
            if (distance > maxd){
              maxd = distance;
              replace = seq[j];
            }
          }
        }
        if (flag == 0)
        {
          maxd = 99999;
          replace = tmp->d;
          break;
        }
        tmp = tmp->next;
      }
      change = search(pg, replace);
      change->next->d = seq[i];
    }
    insertTable(pg->next, table, n_frames, faults);
    faults++;
  }
  display(pg);
  // check formatting
  for (int i = length(pg) * 3; i <= 22; i++)
    printf(" ");
  printf("->    %-2d   \n", faults);
  }
  putTable(table, n_frames, faults);
}

void LFU(int seq[30], int len, int n_frames)
{
  int size = 0;
  int faults = 0;
  int table[10][20];
  int freq[n_frames];
  node *pg = createlist();
  printf("   Frame ->        In Memory      -> Faults \n\n");
  for (int i = 0; i < len; i++) {
    printf("     %-2d  ->", seq[i]);
    node *found = search(pg, seq[i]);
    node *tmp;
    if (!found) {
      if (size < n_frames) {
        insertlast(pg, seq[i]);
        size++;
        tmp = pg->next;
        while (tmp->next){
          tmp = tmp->next;
        }
        tmp->order = faults+1;
      }
      else {
        tmp = pg->next;
        int repl;
        int leastfreq = 999, eqno = 0;
        for( int j = 0; j < n_frames; j++)
          freq[j]=0;
        node *change;
        int k = 0;
        tmp = pg->next;
        while (tmp){
          freq[k] = frequency(seq, tmp->d, i, len);
          //printf("freq %d = %d\n", tmp->d, freq[k])
          if (freq[k]<leastfreq){
            leastfreq = freq[k];
            repl = tmp->d;
          }
          //leastfreq = tmp->d;
          k++;
          tmp = tmp->next;
        }
        tmp = pg->next;
        eqno = 0;
        for (int in = 0; in < k; in++){
          //printf("freq %d ", freq[in]);
          if (freq[in] == leastfreq)
            eqno++;
        }

        int minorder = 999;
        if (eqno > 1)
        {
          int in = 0;
          while (tmp){
            if (freq[in] == leastfreq)
              if (minorder>tmp->order){
                minorder = tmp->order;
                repl = tmp->d;
              }
            in++;
            tmp = tmp->next;
          }
          tmp = tmp->next;
        }
        change = search(pg, repl);
        change->next->d = seq[i];
        change->next->order = faults+1;
      }
      insertTable(pg->next, table, n_frames, faults);
      faults++;
    }
    display(pg);
    // check formatting
    for (int i = length(pg) * 3; i <= 22; i++)
      printf(" ");
    printf("->    %-2d   \n", faults);
  }
  putTable(table, n_frames, faults);
}


int main() {
  /*node* t1=createlist();
  insertfirst(t1,1);
  insertfirst(t1,2);
  insertlast(t1,3);
  deletefirst(t1);
  deletelast(t1);
  display(t1);  */
  int n_free_frames = -1;
  int n_reqd_frames = -1;
  char buffer[20] = {0};
  int sequence[30];
```

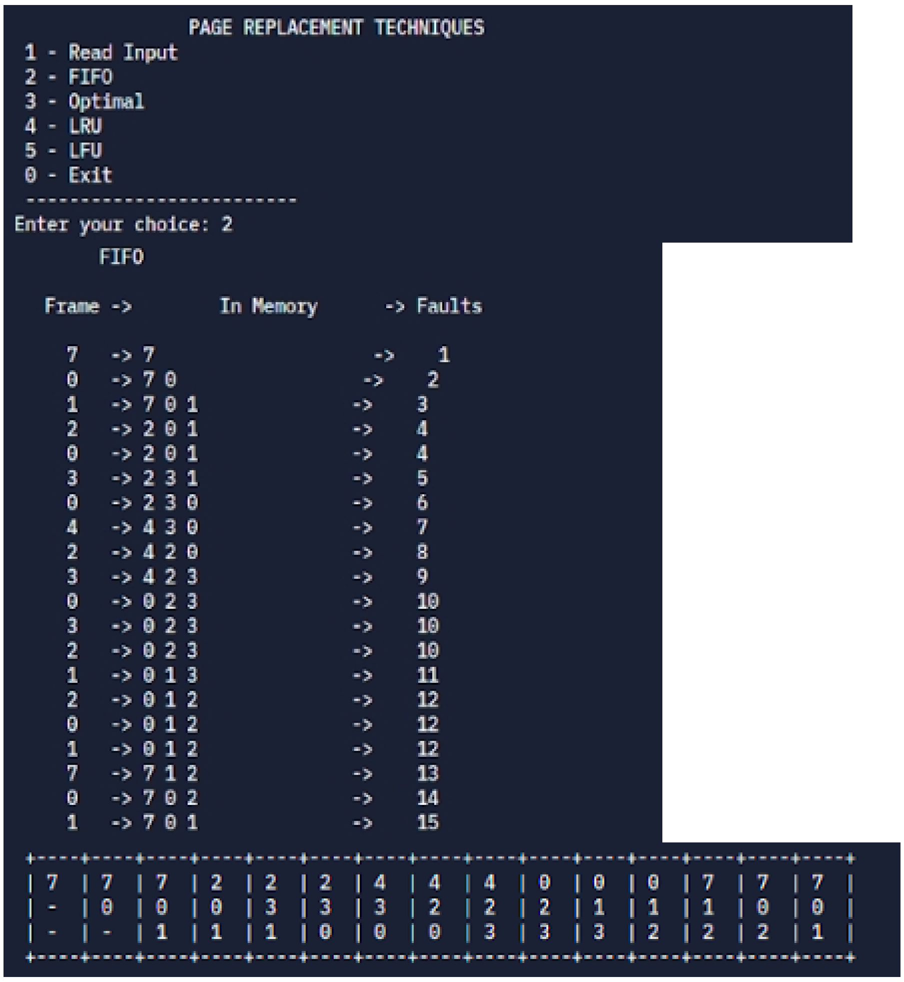
```
 int choice = -1;
 int len = 0;

 while (1) {
   printf("\t\t\t\tPAGE REPLACEMENT TECHNIQUES\n");
   printf(" 1 - Read Input\n");
   printf(" 2 - FIFO\n");
   printf(" 3 - Optimal\n");
   printf(" 4 - LRU\n");
   printf(" 5 - LFU\n");
   printf(" 0 - Exit\n");
   printf(" -------------------------\n");
   printf("Enter your choice: ");
   scanf("%d", &choice);

   switch (choice) {
   case 0:
     exit(0);
   case 1:
     printf("Enter the number of free frames: ");
     scanf("%d", &n_free_frames);
     printf("Enter the number of required frames: ");
     scanf("%d", &n_reqd_frames);
     getchar();
     printf("Enter the length of Reference string: ");
     scanf("%d", &len);
     printf("Enter the Reference String: ");
     for (int i = 0; i < len; i++) {
       scanf("%d", &sequence[i]);
     }
     break;
   case 2:
     printf("\n\t\tFIFO\n");
     FIFO(sequence, len, n_reqd_frames);
     break;
   case 3:
     printf("\n\t\t\tOPTIMAL\n");
     optimal(sequence, len, n_reqd_frames);
     break;
   case 4:
     printf("\n\t\tLRU\n");
     LRU(sequence, len, n_reqd_frames);
     break;
   case 5:
     printf("\n\t\tLFU\n");
     LFU(sequence, len, n_reqd_frames);
     break;
   default:
     printf("Invalid Input!\n");
```

```
    }
    printf("\n");
  }
}
```

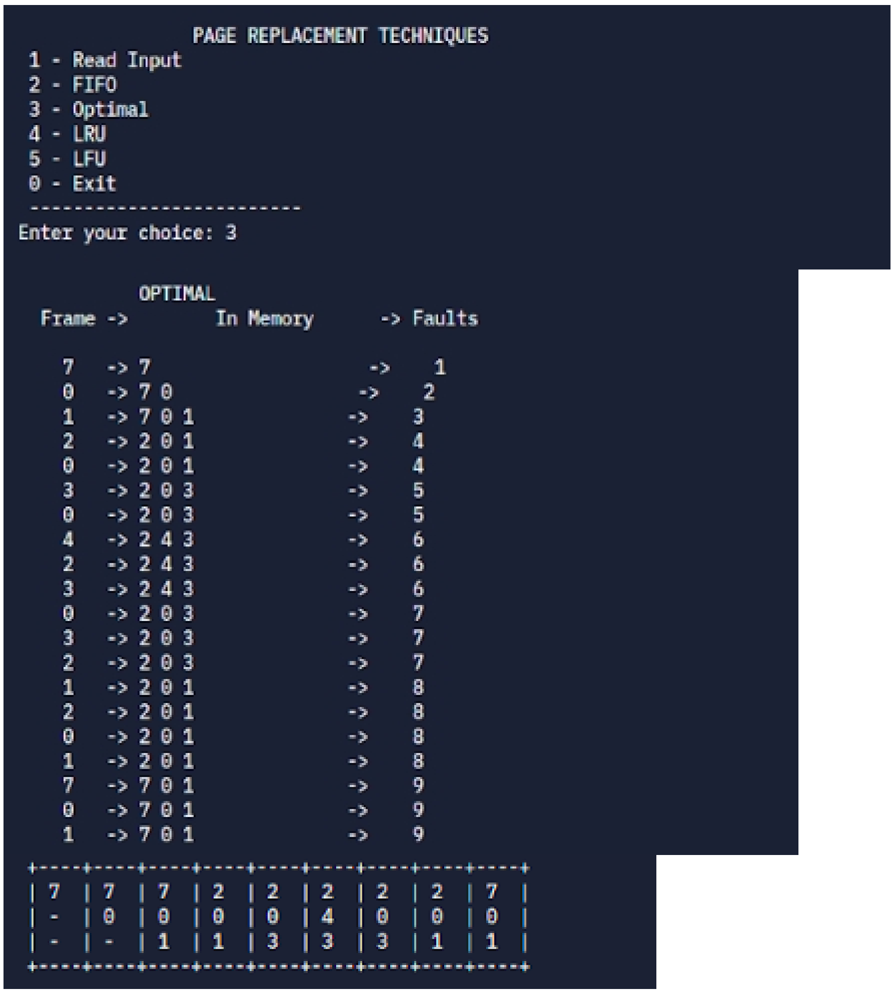
<u>Output:</u>



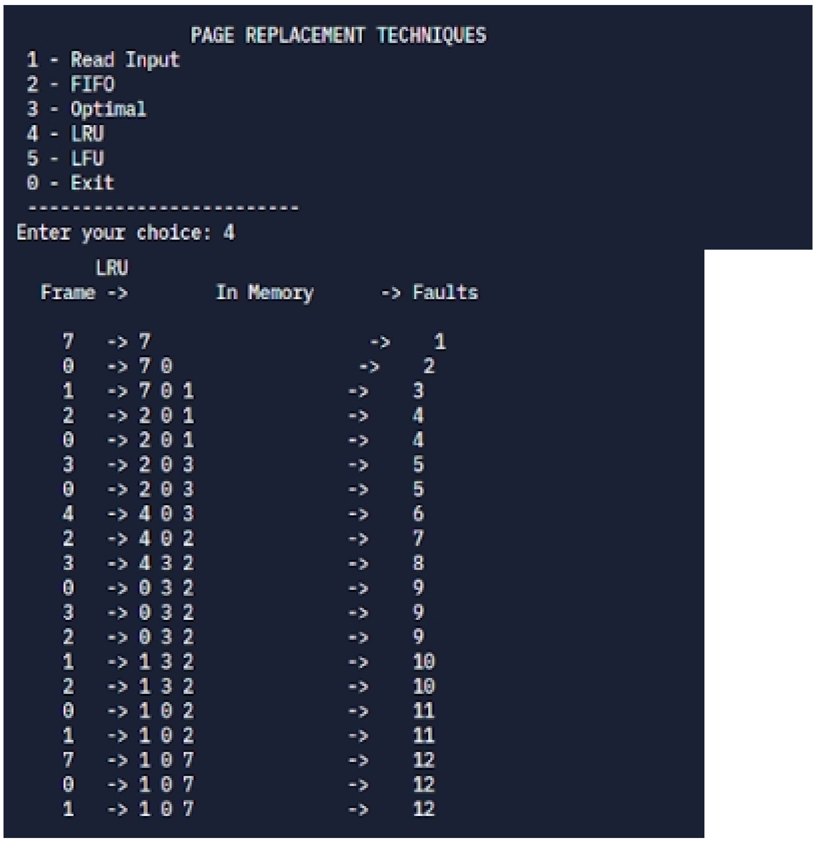
```
~/OSL$ ./a10
                  PAGE REPLACEMENT TECHNIQUES
 1 - Read Input
 2 - FIFO
 3 - Optimal
 4 - LRU
 5 - LFU
 0 - Exit
 -------------------------
Enter your choice: 1
Enter the number of free frames: 3
Enter the number of required frames: 3
Enter the length of Reference string: 20
Enter the Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
```



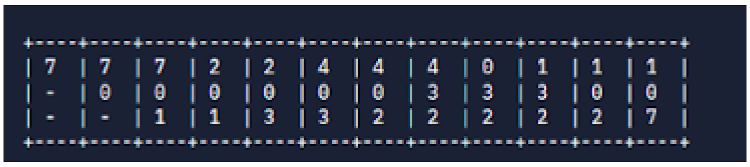
```
                  PAGE REPLACEMENT TECHNIQUES
 1 - Read Input
 2 - FIFO
 3 - Optimal
 4 - LRU
 5 - LFU
 0 - Exit
 -------------------------
Enter your choice: 2
        FIFO

   Frame ->        In Memory      -> Faults

     7   -> 7                   ->    1
     0   -> 7 0                ->    2
     1   -> 7 0 1             ->    3
     2   -> 2 0 1             ->    4
     0   -> 2 0 1             ->    4
     3   -> 2 3 1             ->    5
     0   -> 2 3 0             ->    6
     4   -> 4 3 0             ->    7
     2   -> 4 2 0             ->    8
     3   -> 4 2 3             ->    9
     0   -> 0 2 3             ->    10
     3   -> 0 2 3             ->    10
     2   -> 0 2 3             ->    10
     1   -> 0 1 3             ->    11
     2   -> 0 1 2             ->    12
     0   -> 0 1 2             ->    12
     1   -> 0 1 2             ->    12
     7   -> 7 1 2             ->    13
     0   -> 7 0 2             ->    14
     1   -> 7 0 1             ->    15

 +----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+
 | 7  | 7  | 7  | 2  | 2  | 2  | 4  | 4  | 4  | 0  | 0  | 0  | 7  | 7  | 7  |
 | -  | 0  | 0  | 0  | 3  | 3  | 3  | 2  | 2  | 2  | 1  | 1  | 1  | 0  | 0  |
 | -  | -  | 1  | 1  | 1  | 0  | 0  | 0  | 3  | 3  | 3  | 2  | 2  | 2  | 1  |
 +----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+
```



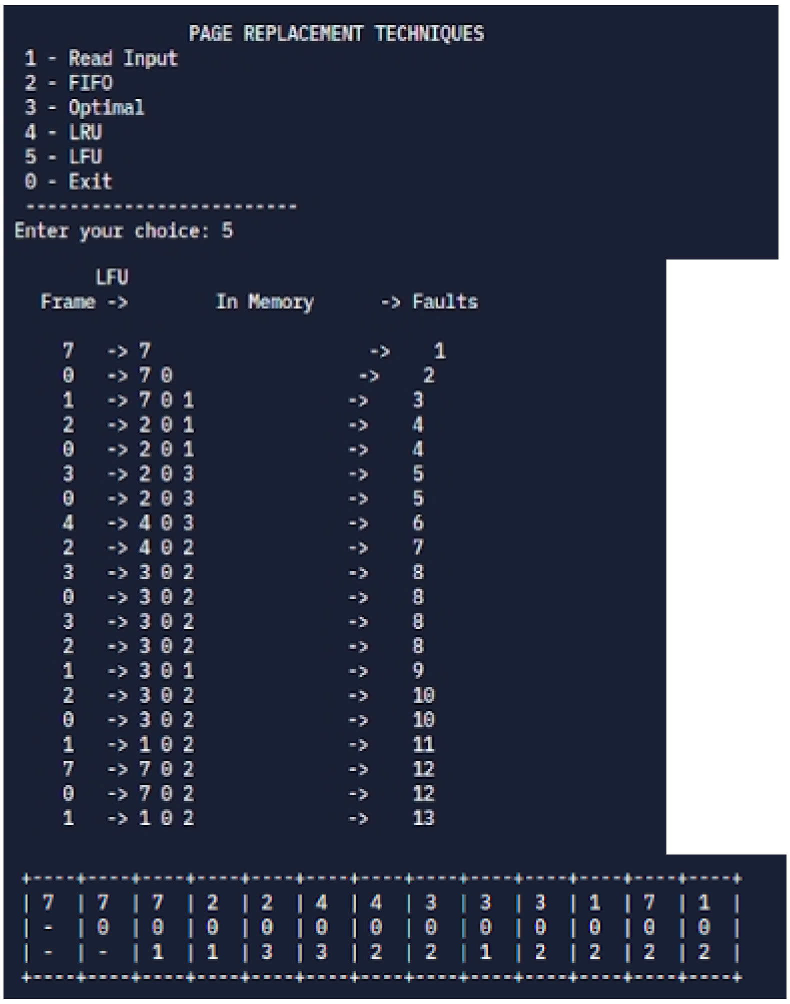
```
                    PAGE REPLACEMENT TECHNIQUES
 1 - Read Input
 2 - FIFO
 3 - Optimal
 4 - LRU
 5 - LFU
 0 - Exit
 -------------------------
Enter your choice: 3

            OPTIMAL
  Frame ->        In Memory      -> Faults

    7   -> 7                    ->    1
    0   -> 7 0                 ->    2
    1   -> 7 0 1              ->    3
    2   -> 2 0 1              ->    4
    0   -> 2 0 1              ->    4
    3   -> 2 0 3              ->    5
    0   -> 2 0 3              ->    5
    4   -> 2 4 3              ->    6
    2   -> 2 4 3              ->    6
    3   -> 2 4 3              ->    6
    0   -> 2 0 3              ->    7
    3   -> 2 0 3              ->    7
    2   -> 2 0 3              ->    7
    1   -> 2 0 1              ->    8
    2   -> 2 0 1              ->    8
    0   -> 2 0 1              ->    8
    1   -> 2 0 1              ->    8
    7   -> 7 0 1              ->    9
    0   -> 7 0 1              ->    9
    1   -> 7 0 1              ->    9

 +----+----+----+----+----+----+----+----+----+
 | 7  | 7  | 7  | 2  | 2  | 2  | 2  | 2  | 7  |
 | -  | 0  | 0  | 0  | 0  | 4  | 0  | 0  | 0  |
 | -  | -  | 1  | 1  | 3  | 3  | 3  | 1  | 1  |
 +----+----+----+----+----+----+----+----+----+
```



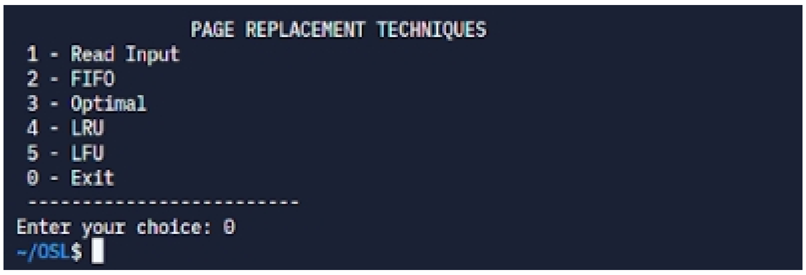
```
                    PAGE REPLACEMENT TECHNIQUES
 1 - Read Input
 2 - FIFO
 3 - Optimal
 4 - LRU
 5 - LFU
 0 - Exit
 -------------------------
Enter your choice: 4
       LRU
  Frame ->        In Memory      -> Faults

    7   -> 7                    ->    1
    0   -> 7 0                 ->    2
    1   -> 7 0 1              ->    3
    2   -> 2 0 1              ->    4
    0   -> 2 0 1              ->    4
    3   -> 2 0 3              ->    5
    0   -> 2 0 3              ->    5
    4   -> 4 0 3              ->    6
    2   -> 4 0 2              ->    7
    3   -> 4 3 2              ->    8
    0   -> 0 3 2              ->    9
    3   -> 0 3 2              ->    9
    2   -> 0 3 2              ->    9
    1   -> 1 3 2              ->    10
    2   -> 1 3 2              ->    10
    0   -> 1 0 2              ->    11
    1   -> 1 0 2              ->    11
    7   -> 1 0 7              ->    12
    0   -> 1 0 7              ->    12
    1   -> 1 0 7              ->    12
```



```
+----+----+----+----+----+----+----+----+----+----+----+----+
| 7  | 7  | 7  | 2  | 2  | 4  | 4  | 4  | 0  | 1  | 1  | 1  |
| -  | 0  | 0  | 0  | 0  | 0  | 0  | 3  | 3  | 3  | 0  | 0  |
| -  | -  | 1  | 1  | 3  | 3  | 2  | 2  | 2  | 2  | 2  | 7  |
+----+----+----+----+----+----+----+----+----+----+----+----+
```



```
                PAGE REPLACEMENT TECHNIQUES
 1 - Read Input
 2 - FIFO
 3 - Optimal
 4 - LRU
 5 - LFU
 0 - Exit
 -------------------------
Enter your choice: 5

       LFU
  Frame ->        In Memory      -> Faults

    7   -> 7                   ->    1
    0   -> 7 0                ->    2
    1   -> 7 0 1             ->    3
    2   -> 2 0 1             ->    4
    0   -> 2 0 1             ->    4
    3   -> 2 0 3             ->    5
    0   -> 2 0 3             ->    5
    4   -> 4 0 3             ->    6
    2   -> 4 0 2             ->    7
    3   -> 3 0 2             ->    8
    0   -> 3 0 2             ->    8
    3   -> 3 0 2             ->    8
    2   -> 3 0 2             ->    8
    1   -> 3 0 1             ->    9
    2   -> 3 0 2             ->    10
    0   -> 3 0 2             ->    10
    1   -> 1 0 2             ->    11
    7   -> 7 0 2             ->    12
    0   -> 7 0 2             ->    12
    1   -> 1 0 2             ->    13

+----+----+----+----+----+----+----+----+----+----+----+----+----+
| 7  | 7  | 7  | 2  | 2  | 4  | 4  | 3  | 3  | 3  | 1  | 7  | 1  |
| -  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| -  | -  | 1  | 1  | 3  | 3  | 2  | 2  | 1  | 2  | 2  | 2  | 2  |
+----+----+----+----+----+----+----+----+----+----+----+----+----+
```



```
                PAGE REPLACEMENT TECHNIQUES
 1 - Read Input
 2 - FIFO
 3 - Optimal
 4 - LRU
 5 - LFU
 0 - Exit
 -------------------------
Enter your choice: 0
~/OSL$ █
```

Name: Krithika Swaminathan
Roll No.: 205001057

Learning outcomes:

- Page replacement techniques were understood and implemented.
- The different page replacement techniques were compared.
- The optimal page replacement technique was found to produce the least number of page faults.