```
Script started on 2020-03-29 21:07:50+0530
 ]0;GAYU@GAYU: ~/Desktop/replacem $ gcc replacement.c -o r
 ]0;GAYU@GAYU: ~/Desktop/replacem $ cat replacement.c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int frame_size;
int pagefaults;
typedef struct node
{
 int data;
 struct node* next;
}n;
n* create()
{
 n* head = (n*)malloc(sizeof(n));
 head->next = NULL;
 return head;
}
n* createNode(int data)
{
 n* newNode = malloc(sizeof(n));
 newNode->data = data;
 newNode->next = NULL;
 return newNode;
}
void insertLast(n* head, n* newNode)
{
 n* temp = head;
 while(temp->next!=NULL)
  temp = temp->next;
 newNode->next = temp->next;
 temp->next = newNode;
}
n* removeFront(n* head)
{
 n* temp = head->next;
 if(head == NULL)
  return NULL;
 printf("removed %d\n", temp->data);
 head = head->next->next;
 return temp;
}
int search(n* head, int ref)
{
 n* temp = head->next;
 int status;
 while(temp!= NULL)
 {
  if(temp->data == ref)
  {
   status = 0;
   return status;
```

```
  }
  temp = temp->next;
 }
 return -1;
}

void display(n* head)
{
 n* temp = head->next;
 while(temp!=NULL)
 {
  printf("%d ",temp->data);
  temp = temp->next;
 }
 printf("\n");
}

void fifo(n* head, int ref_string[20],int req_frame_size)
{
 pagefaults = 0;
 n* newNode;
 for(int i = 0; i < req_frame_size; i++)
 {
  newNode = createNode(ref_string[i]);
  insertLast(head,newNode);
  pagefaults++;
  display(head);
 }
 for(int i = req_frame_size; i< 20; i++)
 {
  int status = search(head,ref_string[i]);
  if(status == -1) // string not found in the alocated list
  {
   head = removeFront(head);
   newNode = createNode(ref_string[i]);
   insertLast(head,newNode);
   pagefaults++;
   display(head);
  }
  else
  { display(head);
   continue;
  }
 }
 printf("Total number of pagefaults : %d \n",pagefaults);
}
void replace(n* head, int rep_no,int replace_with)
{
 n* temp = head->next;
 while(temp!=NULL)
 {
  if(temp->data == rep_no)
  {
   temp->data = replace_with;
```

```
   return;
  }
  temp = temp->next;
 }
}
int longest_time(int ref_string[20],int index,int data)
{
 //printf("inside longest time ## \n");
 int count = 0;
 for(int i = index ; i < 20; i++)
 {
  if(ref_string[i] == data)
  {
   return count;
  }
  else
  { count++;
   continue;
  }
 }
 return count;
}
void optimal(n* head, int ref_string[20],int req_frame_size)
{
 pagefaults = 0;
 n* newNode;
 n* temp;
 int max_count = 0;
 int max_count_for_no = 0;
 int count = 0;
 for(int i = 0; i < req_frame_size; i++)
 {
  newNode = createNode(ref_string[i]);
  insertLast(head,newNode);
  pagefaults++;
  display(head);
 }
 // 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
 for(int i = req_frame_size; i< 20; i++)
 {
  int status = search(head,ref_string[i]);
  if(status == -1) // string not found in the alocated list
  {
   max_count =0;
   max_count_for_no = 0;
   temp = head->next;
   while(temp!=NULL)
   {
    count = longest_time(ref_string, i, temp->data);
    if(count > max_count)
    {
     max_count = count;
     max_count_for_no = temp->data;
    }
```

```
   temp = temp->next;
   }
   replace(head,max_count_for_no,ref_string[i]);
   pagefaults++;
   display(head);
  }
  else
  { display(head);
   continue;
  }
 }
 printf("Total number of pagefaults : %d \n",pagefaults);
}
int longest_time_lru(int ref_string[20],int index,int data)
{
 for(int i = index ; i >= 0; i--)
 {
  if(ref_string[i] == data)
  {
   return i;
  }
  else
   continue;
 }
 return 100;
}
void lru(n* head, int ref_string[20],int req_frame_size)
{
 pagefaults = 0;
 n* newNode;
 n* temp;
 int max_age = 100;
 int max_age_for_no = 100;
 int count = 0;

 for(int i = 0; i < req_frame_size; i++)
 {
  newNode = createNode(ref_string[i]);
  insertLast(head,newNode);
  pagefaults++;
  display(head);
 }
 // 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
 for(int i = req_frame_size; i< 20; i++)
 {
  int status = search(head,ref_string[i]);
  if(status == -1) // string not found in the alocated list
  {
   max_age = 100;
   max_age_for_no = 100;
   temp = head->next;
   while(temp!=NULL)
   {
    count = longest_time_lru(ref_string, i, temp->data);
```

```
    if(count < max_age)
    {
     max_age = count;
     max_age_for_no = temp->data;
    }
    temp = temp->next;
   }
   replace(head,max_age_for_no,ref_string[i]);
   pagefaults++;
   display(head);
  }
  else
  { display(head);
   continue;
  }
 }
 printf("Total number of pagefaults : %d \n",pagefaults);
}
int longest_time_lfu(int ref_string[20],int index,int data)
{
 int count = 0;
 for(int i = index ; i >= 0; i--)
 {
  if(ref_string[i] == data)
  {
   count++;
  }
 }
 return count;
}
void lfu(n* head, int ref_string[20],int req_frame_size)
{
 pagefaults = 0;
 n* newNode;
 n* temp;
 int min_freq = 100;
 int min_freq_for_no = 100;
 int count = 0;

 for(int i = 0; i < req_frame_size; i++)
 {
  newNode = createNode(ref_string[i]);
  insertLast(head,newNode);
  pagefaults++;
  display(head);
 }
 // 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
 for(int i = req_frame_size; i< 20; i++)
 {
  int status = search(head,ref_string[i]);
  if(status == -1) // string not found in the alocated list
  {
   min_freq = 100;
   min_freq_for_no = 100;
```

```
  temp = head->next;
  while(temp!=NULL)
  {
   count = longest_time_lfu(ref_string, i, temp->data);
   if(count < min_freq)
   {
    min_freq = count;
    min_freq_for_no = temp->data;
   }
   temp = temp->next;
  }
  replace(head,min_freq_for_no,ref_string[i]);
  pagefaults++;
  display(head);
 }
 else
 { display(head);
  continue;
 }
}
printf("Total number of pagefaults : %d \n",pagefaults);
}


void main()
{
 int req_frame_size;
 int ref_string[20];
 int choice;
 n* head = create();
 do
 {
 printf("1. READ_INPUT \n");
 printf("2. FIFO\n");
 printf("3. OPTIMAL \n");
 printf("4. LRU ( Least recently used)\n");
 printf("5. LFU (Least Frequently used\n");
 printf("6. Exit\n");
 printf("Enter choice : ");
 scanf("%d",&choice);
 switch(choice)
 {
  case 1: printf("\nPrint Total Frame available : ");
   scanf("%d",&frame_size);
   printf("Frames required by the process : ");
   scanf("%d",&req_frame_size);
   printf("Enter reference string - (size 20) : ");
   for(int i = 0; i < 20 ; i++)
   scanf("%d", &ref_string[i]);
   break;
  case 2:
   if(req_frame_size < frame_size)
   fifo(head, ref_string,req_frame_size);
   else
```

```
    printf("Required frames exceeding available frames - Exiting\n");
   break;
  case 3:
   if(req_frame_size < frame_size)
    optimal(head, ref_string,req_frame_size);
   else
    printf("Required frames exceeding available frames - Exiting\n");
   break;
  case 4:
   if(req_frame_size < frame_size)
    lru(head, ref_string,req_frame_size);
   else
    printf("Required frames exceeding available frames - Exiting\n");
   break;
  case 5:
   if(req_frame_size < frame_size)
    lfu(head, ref_string,req_frame_size);
   else
    printf("Required frames exceeding available frames - Exiting\n");
   break;
 }
}
while(choice!=6);
}
```

```
]0;GAYU@GAYU: ~/Desktop/replacem  [01;32mGAYU@GAYU [00m: [01;34m~/Desktop/replacem [00m$ ./r
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU ( Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 1

Print Total Frame available : 10
Frames required by the process : 4
Enter reference string - (size 20) : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU ( Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 2
7
7 0
7 0 1
```

```
7 0 1 2
7 0 1 2
removed 7
0 1 2 3
0 1 2 3
removed 0
1 2 3 4
1 2 3 4
1 2 3 4
removed 1
2 3 4 0
2 3 4 0
2 3 4 0
removed 2
3 4 0 1
removed 3
4 0 1 2
4 0 1 2
4 0 1 2
removed 4
0 1 2 7
0 1 2 7
0 1 2 7
Total number of pagefaults : 10
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU ( Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 6
 ]0;GAYU@GAYU: ~/Desktop/replacem  [01;32mGAYU@GAYU [00m: [01;34m~/Desktop/replacem [00m$ ./r
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU ( Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 1

Print Total Frame available : 10
Frames required by the process : 4
Enter reference string - (size 20) : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU ( Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 3
7
7 0
7 0 1
7 0 1 2
```

7 0 1 2
3 0 1 2
3 0 1 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2
1 0 4 2
1 0 4 2
1 0 4 2
1 0 4 2
1 0 7 2
1 0 7 2
1 0 7 2
Total number of pagefaults : 8
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU (Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 6
]0;GAYU@GAYU: ~/Desktop/replacem [01;32mGAYU@GAYU [00m: [01;34m~/Desktop/replacem [00m$ ./r
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU (Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 1

Print Total Frame available : 10
Frames required by the process : 3
Enter reference string - (size 20) : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU (Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 4
7
7 0
7 0 1
2 0 1
2 0 1
2 0 3
2 0 3
4 0 3
4 0 2
4 3 2
0 3 2

0 3 2
0 3 2
1 3 2
1 3 2
1 0 2
1 0 2
1 0 7
1 0 7
1 0 7
Total number of pagefaults : 12
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU (Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 6
]0;GAYU@GAYU: ~/Desktop/replacem [01;32mGAYU@GAYU [00m: [01;34m~/Desktop/replacem [00m$ ./r
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU (Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 1

Print Total Frame available : 10
Frames required by the process : 3
Enter reference string - (size 20) : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU (Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 5
7
7 0
7 0 1
2 0 1
2 0 1
3 0 1
3 0 1
4 0 1
2 0 1
2 0 3
2 0 3
2 0 3
2 0 3
1 0 3
2 0 3
2 0 3
2 0 1
2 0 7

2 0 7
2 0 1
Total number of pagefaults : 13
1. READ_INPUT
2. FIFO
3. OPTIMAL
4. LRU (Least recently used)
5. LFU (Least Frequently used
6. Exit
Enter choice : 5 6
 ]0;GAYU@GAYU: ~/Desktop/replacem [01;32mGAYU@GAYU [00m: [01;34m~/Desktop/replacem [00m$ exit
exit

Script done on 2020-03-29 21:08:46+0530