Script started on 2019-03-10 17:11:09+0530
praveen@praveen$ cat replacement.c

```
#include<stdio.h>
#include<stdlib.h>
#define MAXSIZE 20
struct node {
   int data;
   struct node *next;
};
typedef struct node Node;
Node *current = NULL;
Node *prev = NULL;
Node* enq(Node *head, int data)
{
   Node *link = (Node*)malloc(sizeof(Node));
   link->data = data;
   link->next = head;
   head = link;
   return head;
}
Node* deq(Node* head)
{
   Node *temp = head;
   if(head == NULL) return head;
   else if(head->next == NULL) {
      free(head);
      head = NULL;
      return head;
   }
   for(temp = head; temp->next != NULL; temp = temp->next)
      prev = temp;
   free(temp);
   prev->next = NULL;
   return head;
}
//deletes val if found in ll
Node* delete(Node *head, int data)
{
   if(head == NULL)
      return head;
   else if(head->data == data) {
      Node *t = head;
      head = head->next;
      free(t);
      return head;
   }
   else {
      Node *temp = NULL, *t = NULL;
      for(temp = head;temp->next != NULL;temp = temp->next) {
         if(temp->next->data == data) {
```

```
            t = temp->next;
            temp->next = temp->next->next;
            free(t);
         }
      }
      return head;
   }
}
//returns 1 if val found in ll
int search(Node *head, int val)
{
   current = head;
   if(head == NULL)
      return 0;
   while(current != NULL) {
      if(current->data == val) {
         return 1;
      }
      current = current->next;
   }
   return 0;
}
//finds size of ll
int findSize(Node *head)
{
   int size = 0;
   if(head == NULL)
   {
      return 0;
   }
   current = head;
   size = 1;
   while(current->next != NULL)
   {
      current = current->next;
      size++;
   }
   return size;
}
void printList(Node *head)
{
   Node *ptr = head;
   while(ptr != NULL) {
      printf("%d ",ptr->data);
      ptr = ptr->next;
   }
   printf("\n");
}
//stores values of linked list in arr
void store(Node *head, int arr[10])
```

```
{
    Node *ptr = head;
    int i = 0;
    while(ptr != NULL) {
        arr[i] = ptr->data;
        ptr = ptr->next;
        i++;
    }
}
//returns max index
int max(int arr[], int len)
{
    int max = 0;
    for(int i = 1; i < len; i++)
    {
        if(arr[i] > arr[max])
            max = i;
    }
    return max;
}
//returns min index
int min(int arr[], int len)
{
    int min = 0;
    for(int i = 0; i < len; i++)
    {
        if(arr[i] <= arr[min])
            min = i;
    }
    return min;
}
//performs linsearch from start to len
int linsearch(int arr[], int start, int len, int t)
{
    int ind = 100;
    for(int i = start; i < len; i++)
    {
        if(arr[i] == t)
            ind = i;
    }
    return ind;
}
void prntarr(int arr[], int len)
{
    for(int i = 0; i < len; i++)
        printf("%d  ",arr[i]);
    printf("\n");
}
int main()
{
```

```
int refstr[MAXSIZE] = {1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6};
int arrlen = 20, choice = 0;
int fsize = 3, pfault = 0;
do {
   printf("\nEnter your choice:\n1.Input 2.View 3.FIFO 4.LRU 5.OPT 6.Exit\nChoice: ");
   scanf("%d",&choice);
   //INPUT
   if(choice == 1) {
      printf("Enter the length of reference string (MAX 20): ");
      scanf("%d",&arrlen);
      printf("Enter the reference string: ");
      for(int i = 0; i < arrlen; i++)
         scanf("%d",&refstr[i]);
      printf("Enter frame size: ");
      scanf("%d",&fsize);
   }
   if(choice == 2) {
      printf("Reference String:\n");
      prntarr(refstr,arrlen);
      printf("Length: %d\n", arrlen);
      printf("Frames: %d\n", fsize);
   }
   //FIFO
   if(choice == 3) {
      Node *head=NULL;
      printf("FIFO:\n");
      for(int i = 0; i < arrlen; i++) {
         int size = findSize(head);
         if(size < fsize) {
            int found = search(head,refstr[i]);
            if(found == 0) {
               pfault++;
               head = enq(head,refstr[i]);
               printList(head);
            }
            else printList(head);
         }
         else {
            int found = search(head,refstr[i]);
            if(found == 0) {
               pfault++;
               head = deq(head);
               head = enq(head,refstr[i]);
               printList(head);
            }
            else printList(head);
         }
      }
      printf("\nPage Faults: %d\n",pfault);
      pfault = 0;
```

```
}
//LRU
if(choice == 4) {
   Node *head = NULL;
   int curlist[MAXSIZE];
   int curfoundind[fsize];
   printf("LRU:\n");
   for(int i = 0; i < arrlen; i++) {
      int size = findSize(head);
      if(size < fsize) {
         int found = search(head,refstr[i]);
         if(found == 0) {
            pfault++;
            head = enq(head,refstr[i]);
            printList(head);
         }
         else printList(head);
      }
      else {
         int found = search(head,refstr[i]);
         if(found == 0) {
            pfault++;
            //LRU
            store(head,curlist);
            for(int j = 0; j < fsize; j++)
               curfoundind[j] = linsearch(refstr,0,i,curlist[j]);
            int minn = min(curfoundind,fsize);
            if(minn == fsize-1){
               head = deq(head);
               head = enq(head,refstr[i]);
            }
            else {
               head = delete(head,curlist[minn]);
               head = enq(head,refstr[i]);
            }
            printList(head);
         }
         else printList(head);
      }
   }
   printf("\nPage Faults: %d\n",pfault);
   pfault = 0;
}
//OPT
if(choice == 5) {
   Node *head=NULL;
   int curlist[MAXSIZE];
   int curfoundind[fsize];
   printf("OPT:\n");
   for(int i = 0; i < arrlen; i++) {
```

```
            int size = findSize(head);
            if(size < fsize) {
               int found = search(head,refstr[i]);
               if(found == 0) {
                  pfault++;
                  head=enq(head,refstr[i]);
                  printList(head);
               }
               else printList(head);
            }
            else {
               int found = search(head,refstr[i]);
               if(found==0) {
                  pfault++;
                  //OPTIMAL
                  store(head,curlist);
                  for (int j = 0; j < fsize; j++)
                     curfoundind[j] = linsearch(refstr,i+1,arrlen,curlist[j]);
                  int maxx = max(curfoundind,fsize);
                  head = delete(head,curlist[maxx]);
                  head = enq(head,refstr[i]);
                  printList(head);
               }
               else printList(head);
            }
         }
         printf("\nPage Faults: %d\n",pfault);
         pfault = 0;
      }
   }while(choice != 6);
   return 0;
}
praveen@praveen$ gcc replacement.c
praveen@praveen$ ./a.out

Enter your choice:
1.Input 2.View 3.FIFO 4.LRU 5.OPT 6.Exit
Choice: 1
Enter the length of reference string (MAX 20): 10
Enter the reference string: 1 2 3 4 2 3 5 1 4 3 5 1 2 3 # ## ## ## ## ## ## ## #
Enter frame size: 3

Enter your choice:
1.Input 2.View 3.FIFO 4.LRU 5.OPT 6.Exit
Choice: 3
FIFO:
1
2 1
3 2 1
4 3 2
```

4 3 2
4 3 2
5 4 3
1 5 4
1 5 4
3 1 5

Page Faults: 7

Enter your choice:
1.Input 2.View 3.FIFO 4.LRU 5.OPT 6.Exit
Choice: 4
LRU:
1
2 1
3 2 1
4 3 2
4 3 2
4 3 2
5 3 2
1 5 3
4 1 5
3 4 1

Page Faults: 8

Enter your choice:
1.Input 2.View 3.FIFO 4.LRU 5.OPT 6.Exit
Choice: 5
OPT:
1
2 1
3 2 1
4 2 1
4 2 1
3 4 1
5 4 1
5 4 1
5 4 1
3 4 1

Page Faults: 7

Enter your choice:
1.Input 2.View 3.FIFO 4.LRU 5.OPT 6.Exit
Choice: 1
Enter the length of reference string (MAX 20): 20
Enter the reference string: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
Enter frame size: 3

Enter your choice:
1.Input 2.View 3.FIFO 4.LRU 5.OPT 6.Exit
Choice: 2
Reference String:
1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
Length: 20
Frames: 3

Enter your choice:
1.Input 2.View 3.FIFO 4.LRU 5.OPT 6.Exit
Choice: 3
FIFO:
1
2 1
3 2 1
4 3 2
4 3 2
1 4 3
5 1 4
6 5 1
2 6 5
1 2 6
1 2 6
3 1 2
7 3 1
6 7 3
6 7 3
2 6 7
1 2 6
1 2 6
3 1 2
6 3 1

Page Faults: 16

Enter your choice:
1.Input 2.View 3.FIFO 4.LRU 5.OPT 6.Exit
Choice: 4
LRU:
1
2 1
3 2 1
4 3 2
4 3 2
1 4 2
5 1 2
6 5 1
2 6 5
1 2 6
1 2 6

3 1 2
7 3 2
6 7 3
6 7 3
2 6 3
1 2 3
1 2 3
1 2 3
6 2 3

Page Faults: 15

Enter your choice:
1.Input 2.View 3.FIFO 4.LRU 5.OPT 6.Exit
Choice: 5
OPT:
1
2 1
3 2 1
4 2 1
4 2 1
4 2 1
5 2 1
6 2 1
6 2 1
6 2 1
6 2 1
3 2 1
7 2 1
6 2 1
3 2 1
3 2 1
3 2 1
3 2 1
3 2 1
6 2 1

Page Faults: 11

Enter your choice:
1.Input 2.View 3.FIFO 4.LRU 5.OPT 6.Exit
Choice: 6
praveen@praveen$ exit
exit

Script done on 2019-03-10 17:16:19+0530