Name: Om Jadhav **Roll No.:** I3275 **Div:** 2

Statement: Implement the C program for Page Replacement Algorithms: FCFS, LRU, andOptimal for frame size as minimum three.

Code:

```
#include<stdio.h>

#include<stdlib.h>

typedef struct

{

 char data[20][2];          //2nd column stores distance or time

 int end;

}queue;


void enqueue(queue *q,char data,int position);

char dequeue(queue *q,int position);


void fifo(char string[],int frameSize,int count);

void optimal(char string[],int frameSize,int count);

void lru(char string[],int frameSize,int count);



void main()

{

 int frameSize,count,cnt,ch;

 char string[50];


 printf("Enter the string: ");

 count=0;

 do
```

```
        {
         scanf("%c",&string[count]);
         count++;
        }while(string[count-1]!='\n');
 count--;        //This is the no. of data available
 printf("\nEnter the size of the frame: ");
 scanf("%d",&frameSize);
do
{
 printf("\nMENU\n====\n1.FIFO\n2.Least Recently Used (LRU)\n3.Optimal\n4.Exit\n\nYour Choice:");
 scanf("%d",&ch);
 switch(ch)
        {
         case 1:fifo(string,frameSize,count);
                break;
         case 2:lru(string,frameSize,count);
                break;
         case 3:optimal(string,frameSize,count);
                break;
         case 4:exit(0);
                break;
         default:printf("\nInvalid choice! Please try again!");

                 continue;
        }
}while(ch!=4);

}
```

```
void enqueue(queue *q,char data,int position)

{

 q->data[position][0]=data;

}




char dequeue(queue *q,int position)

{

 char value;

 value=q->data[position][0];

 return(value);

}


void fifo(char string[],int frameSize,int count)

{

 int cnt,cnt2,flag,faults=0;

 queue q;

 int firstin=-1;

 q.end=0;

 printf("\nData Requested\tFrame contents\t Page
Fault\n=============================================");

 for(cnt=0;cnt<count;cnt+=2)      //String[] includes spaces

              {

               printf("\n\n\t%c",string[cnt]);

               flag=0;

               for(cnt2=0;cnt2<q.end;cnt2++)

                            {

                             if(string[cnt]==q.data[cnt2][0])
```

```
                {
                 flag=1;
                 break;
                }
            }
        if(flag==0)
            {
             faults++;
             if(q.end<frameSize)
                {       //Frame has empty slots
                 enqueue(&q,string[cnt],q.end);
                 q.end++;
                }
             else
                {
                 //printf("\n\n\tPage containing %c was replaced!"),
                 dequeue(&q,firstin);
                 firstin=(firstin+1)%(q.end);
                 enqueue(&q,string[cnt],firstin);
                }
             printf("\t ");
             for(cnt2=0;cnt2<q.end;cnt2++)
                {
                 printf("%c  ",q.data[cnt2][0]);
                }
             printf("\t\tY");
          }

    else
```

```
                  {
                        printf("\t ");
                        for(cnt2=0;cnt2<q.end;cnt2++)
                                    {
                                     printf("%c  ",q.data[cnt2][0]);
                                    }
                        printf("\t\tN");
                  }

            }
 printf("\n\n================================================\n");
 printf("\nTotal no. of Page Faults: %d\n\n",faults);
}




void optimal(char string[],int frameSize,int count)
{
 int cnt,cnt2,selector,flag,max,faults=0;
 int distance[20];
 queue q;
 q.end=0;
 printf("\nData Requested\tFrame contents\t Page
Fault\n================================================");
 for(cnt=0;cnt<count;cnt+=2)      //String[] includes spaces
            {
             printf("\n\n\t%c",string[cnt]);
             flag=0;
             for(cnt2=0;cnt2<q.end;cnt2++)
```

```
    {   //check for existing data in pages
     if(string[cnt]==q.data[cnt2][0])
        {
         flag=1;
         break;
        }
    }
if(flag==0)
    {
     faults++;
     if(q.end<frameSize)
        {   //Frame has empty slots
         enqueue(&q,string[cnt],q.end);
         q.data[q.end][1]=cnt;  //Update time
         q.end++;
        }
     else
        {
         for(cnt2=0;cnt2<q.end;cnt2++)
            {   //Reset reference distances
             distance[cnt2]=0;
            }
         for(selector=0;selector<q.end;selector++)
            {   //Calculate distance of next reference from current position
              for(cnt2=cnt;cnt2<count;cnt2+=2)      //String[] includes spaces
                {
                 if(string[cnt2]==q.data[selector][0])
                    {
                     distance[selector]=cnt2/2;
                     break;
                    }
                if(distance[selector]==0)
                    { //No further reference
                     distance[selector]=99-q.data[selector][1];
                    }
                }
            }
         max=0;
         /*Select farthest referenced page for replacement*/
         for(cnt2=0;cnt2<q.end;cnt2++)
            {
             if(distance[cnt2]>max)
                {
                 max=distance[cnt2];
                 selector=cnt2;
                }
            }
         dequeue(&q,selector);
         enqueue(&q,string[cnt],selector);
         q.data[selector][1]=cnt;//Update time
        }
    printf("\t ");
    for(cnt2=0;cnt2<q.end;cnt2++)
        {
         printf("%c  ",q.data[cnt2][0]);
        }
    printf("\t\tY");
}
```

```
                else
                      {     //Data exists in page frame
                             printf("\t ");
                             for(cnt2=0;cnt2<q.end;cnt2++)
                                         {
                                          printf("%c  ",q.data[cnt2][0]);
                                         }
                             printf("\t\tN");
                      }


                }
 printf("\n\n====================================================\n");
 printf("\nTotal no. of Page Faults: %d\n\n",faults);
}


void lru(char string[],int frameSize,int count)
{
 int cnt,cnt2,selector,flag,min,faults=0;
 queue q;
 q.end=0;
 printf("\nData Requested\tFrame contents\t Page
Fault\n====================================================");
 for(cnt=0;cnt<count;cnt+=2)      //String[] includes spaces
                {
                 printf("\n\n\t%c",string[cnt]);
                 flag=0;
                 for(cnt2=0;cnt2<q.end;cnt2++)
                             {            //check for existing data in pages
                              if(string[cnt]==q.data[cnt2][0])
```

```
        {
         q.data[cnt2][1]=(cnt/2)+1;         //Update time
         flag=1;
         break;
        }
    }
if(flag==0)
    {
     faults++;
     if(q.end<frameSize)
        {            //Frame has empty slots
         enqueue(&q,string[cnt],q.end);
         q.data[q.end][1]=(cnt/2)+1;       //Update time
         q.end++;
        }
     else
        {
         min=99;
         /*Select farthest referenced page for replacement*/
         for(cnt2=0;cnt2<q.end;cnt2++)
            {
             if(q.data[cnt2][1]<min)
                {
                 min=q.data[cnt2][1];
                 selector=cnt2;
                }
            }
         dequeue(&q,selector);
         enqueue(&q,string[cnt],selector);
```

```
                                        q.data[selector][1]=(cnt/2)+1;  //Update time
                                        }
                            printf("\t ");
                            for(cnt2=0;cnt2<q.end;cnt2++)
                                        {
                                        printf("%c  ",q.data[cnt2][0]);
                                        }
                            printf("\t\tY");
                    }
            else
                    {     //Data exists in page frame
                          printf("\t ");
                          for(cnt2=0;cnt2<q.end;cnt2++)
                                        {
                                        printf("%c  ",q.data[cnt2][0]);
                                        }
                          printf("\t\tN");
                    }

            }
 printf("\n\n==================================================\n");
 printf("\nTotal no. of Page Faults: %d\n\n",faults);
}
```

```
/*OUTPUT:
student@student-OptiPlex-390:~/38$ gcc pract6.c
student@student-OptiPlex-390:~/38$ ./a.out
Enter the string: 1 2 3 4 5 3 4 1 6 7 8 7 8 9 5 4 2 4 9
Enter the size of the frame: 3

MENU
====
1.FIFO
2.Least Recently Used (LRU)
3.Optimal
4.Exit

Your Choice:1

Data Requested Frame contents   Page Fault
==============================================

        1        1              Y

        2        1  2           Y

        3        1  2  3                Y

        4        4  2  3                Y

        5        4  5  3                Y

        3        4  5  3                N

        4        4  5  3                N
```

```
      1        4  5  1            Y

      6        6  5  1            Y

      7        6  7  1            Y

      8        6  7  8            Y

      7        6  7  8            N

      8        6  7  8            N

      9        9  7  8            Y

      5        9  5  8            Y

      4        9  5  4            Y

      2        2  5  4            Y

      4        2  5  4            N

      9        2  9  4            Y

=============================================

Total no. of Page Faults: 14
```

```
MENU

====

1.FIFO

2.Least Recently Used (LRU)

3.Optimal

4.Exit


Your Choice:2


Data Requested Frame contents      Page Fault

==============================================


        1         1              Y


        2         1  2           Y


        3         1  2  3                Y


        4         4  2  3                Y


        5         4  5  3                Y


        3         4  5  3                N


        4         4  5  3                N


        1         4  1  3                Y


        6         4  1  6                Y
```

```
       7       7  1  6              Y

       8       7  8  6              Y

       7       7  8  6              N

       8       7  8  6              N

       9       7  8  9              Y

       5       5  8  9              Y

       4       5  4  9              Y

       2       5  4  2              Y

       4       5  4  2              N

       9       9  4  2              Y

=============================================

Total no. of Page Faults: 14

MENU
====
1.FIFO
2.Least Recently Used (LRU)
```

```
3.Optimal

4.Exit


Your Choice:3


Data Requested Frame contents     Page Fault

=============================================


        1          1                Y


        2          1  2             Y


        3          1  2  3                  Y


        4          1  4  3                  Y


        5          5  4  3                  Y


        3          5  4  3                  N


        4          5  4  3                  N


        1          5  4  1                  Y


        6          5  4  6                  Y


        7          5  4  7                  Y


        8          5  8  7                  Y
```

```
          7          5   8   7                 N

          8          5   8   7                 N

          9          5   8   9                 Y

          5          5   8   9                 N

          4          4   8   9                 Y

          2          4   2   9                 Y

          4          4   2   9                 N

          9          4   2   9                 N

============================================

Total no. of Page Faults: 12
MENU
====
1.FIFO
2.Least Recently Used (LRU)
3.Optimal
4.Exit

Your Choice:4
student@student-OptiPlex-390:~/38$ */
```