VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SWE3001 – Operating Systems Laboratory Manual

Lab - 09

Student Name	Sam Prince Franklin
Reg Number	20MIS1115
Subject Code	SWE3001
Slot	L37,38
Faculty	Dr.M.Braveen

SWE3001 – Operating Systems

Lab – 09 – Page Replacement Algorithm

Implement LRU and Optimal Page replacement algorithm if the number is even then use LRU and if the number is odd use optimal technique.

```
#include<stdio.h>
int n,nf;
int in[100];
int p[50];
int hit=0;
int i,j,k;
int pgfaultcnt=0;
int pgmisscnt=0;


void initialize()
{
    pgfaultcnt=0;
    pgmisscnt=0;
    for(i=0; i<nf; i++)
        p[i]=9999;
}

int isHit(int data)
{
    hit=0;
    for(j=0; j<nf; j++)
    {
        if(p[j]==data)
        {
            hit=1;
            break;
        }

    }

    return hit;
```

```
}

int getHitIndex(int data)
{
    int hitind;
    for(k=0; k<nf; k++)
    {
        if(p[k]==data)
        {
            hitind=k;
            break;
        }
    }
    return hitind;
}

void dispPages()
{
    for (k=0; k<nf; k++)
    {
        if(p[k]!=9999)
            printf(" %d",p[k]);
    }

}

void dispPgFaultCnt()
{
    printf("\nTotal no of page faults:%d",pgfaultcnt);
}

void dispPgMissCnt()
{
    printf("\nTotal no of page misses:%d",pgmisscnt);
}



void optimal()
{
    initialize();
```

```
int near[50];
for(i=0; i<n; i++)
{

   printf("\nFor %d :",in[i]);

   if(isHit(in[i])==0)
   {

      for(j=0; j<nf; j++)
      {
         int pg=p[j];
         int found=0;
         for(k=i; k<n; k++)
         {
            if(pg==in[k])
            {
               near[j]=k;
               found=1;
               break;
            }
            else
               found=0;
         }
         if(!found)
            near[j]=9999;
      }
      int max=-9999;
      int repindex;
      for(j=0; j<nf; j++)
      {
         if(near[j]>max)
         {
            max=near[j];
            repindex=j;
         }
      }
      p[repindex]=in[i];
      pgfaultcnt++;
```

```
            dispPages();
        }
        else
            printf("No page fault");
            pgmisscnt++;
    }
    dispPgFaultCnt();
    dispPgMissCnt();
}

void lru()
{
    initialize();

    int least[50];
    for(i=0; i<n; i++)
    {

        printf("\nFor %d :",in[i]);

        if(isHit(in[i])==0)
        {

            for(j=0; j<nf; j++)
            {
                int pg=p[j];
                int found=0;
                for(k=i-1; k>=0; k--)
                {
                    if(pg==in[k])
                    {
                        least[j]=k;
                        found=1;
                        break;
                    }
                    else
                        found=0;
                }
                if(!found)
                    least[j]=-9999;
```

```
            }
            int min=9999;
            int repindex;
            for(j=0; j<nf; j++)
            {
                if(least[j]<min)
                {
                    min=least[j];
                    repindex=j;
                }
            }
            p[repindex]=in[i];
            pgfaultcnt++;

            dispPages();
        }
        else
            printf("No page fault!");
            pgmisscnt++;

    }
    dispPgFaultCnt();
    dispPgMissCnt();
}


void secondchance()
{
    int usedbit[50];
    int victimptr=0;
    initialize();
    for(i=0; i<nf; i++)
        usedbit[i]=0;
    for(i=0; i<n; i++)
    {
        printf("\nFor %d:",in[i]);
        if(isHit(in[i]))
        {
            printf("No page fault!");
            int hitindex=getHitIndex(in[i]);
```

```
            if(usedbit[hitindex]==0)
                usedbit[hitindex]=1;
        }
        else
        {
            pgfaultcnt++;
            if(usedbit[victimptr]==1)
            {
                do
                {
                    usedbit[victimptr]=0;
                    victimptr++;
                    if(victimptr==nf)
                        victimptr=0;
                }
                while(usedbit[victimptr]!=0);
            }
            if(usedbit[victimptr]==0)
            {
                p[victimptr]=in[i];
                usedbit[victimptr]=1;
                victimptr++;
            }
            dispPages();

        }
        if(victimptr==nf)
            victimptr=0;
    }
    dispPgFaultCnt();
    dispPgMissCnt();
}

int main()
{
    printf("\nEnter length of page reference sequence:");
    scanf("%d",&n);
    printf("\nEnter the page reference sequence:");
    for(i=0; i<n; i++)
        scanf("%d",&in[i]);
```

```
    printf("\nEnter no of frames:");
    scanf("%d",&nf);
    for(i=0;i<n;i++){
        if(in[i] % 2 == 0)
        {
            optimal();
        }
        else{
            lru();
        }
    }
}
```

Output :

Enter length of page reference sequence:15

Enter the page reference sequence:5 2 1 5 1 7 9 6 7 9 3 8 7 3 8

Enter no of frames:3

For 5 : 5
For 2 : 5 2
For 1 : 5 2 1
For 5 :
For 1 :
For 7 : 5 7 1
For 9 : 9 7 1
For 6 : 9 7 6
For 7 :
For 9 :
For 3 : 9 7 3
For 8 : 9 8 3
For 7 : 7 8 3
For 3 :
For 8 :
Total no of page faults:9
Total no of page misses:6%