Name: Lakhan Kumawat

Roll No: 1906055

Branch: CSE-1

Course: CSL4404

Operating System Lab / **Assignment**

1. Write a C program for finding the number of page faults and number of hits using FIFO, OPTIMAL, LRU page replacement algorithm.

- Memory capacity / No. of pages = 5
- Page reference string = 0,2,1,6,4,0,1,0,3,1,2,1

2. Write a C program for finding the number of page faults and number of hits using FIFO, OPTIMAL, LRU page replacement algorithm.

- Memory capacity / No. of pages = 3
- Page reference string = 0,2,1,6,4,0,1,0,3,1,2,1

Solution:

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

void getData()
{
   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);
```

```
}

void initialize()
{
   pgfaultcnt=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 fifo()
{
    initialize();
    for(i=0; i<n; i++)
    {
        printf("\nFor %d :",in[i]);

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

            for(k=0; k<nf-1; k++)
                p[k]=p[k+1];

            p[k]=in[i];
            pgfaultcnt++;
            dispPages();
        }
        else
            printf("No page fault");
    }
    dispPgFaultCnt();
}


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");
  }
  dispPgFaultCnt();
}

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!");
        }
        dispPgFaultCnt();
}



int main()
{
    int choice;
    while(1)
    {
        printf("\nPage Replacement Algorithms\n1.Enter
data\n2.FIFO\n3.Optimal\n4.LRU\n7.Exit\nEnter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
            getData();
            break;
        case 2:
            fifo();
            break;
        case 3:
            optimal();
            break;
        case 4:
            lru();
            break;
        default:
            return 0;
            break;
        }
    }
}
```

Output:

Que 1.
FIFO

```
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
7.Exit
Enter your choice:1

Enter length of page reference sequence:12

Enter the page reference sequence:0 2 1 6 4 0 1 0 3 1 2 1

Enter no of frames:5

Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
7.Exit
Enter your choice:2

For 0 : 0
For 2 : 0 2
For 1 : 0 2 1
For 6 : 0 2 1 6
For 4 : 0 2 1 6 4
For 0 :No page fault
For 1 :No page fault
For 0 :No page fault
For 3 : 2 1 6 4 3
For 1 :No page fault
For 2 :No page fault
For 1 :No page fault
Total no of page faults:6
```

LRU

```
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
7.Exit
Enter your choice:4

For 0 : 0
For 2 : 0 2
For 1 : 0 2 1
For 6 : 0 2 1 6
For 4 : 0 2 1 6 4
For 0 :No page fault!
For 1 :No page fault!
For 0 :No page fault!
For 3 : 0 3 1 6 4
For 1 :No page fault!
For 2 : 0 3 1 2 4
For 1 :No page fault!
Total no of page faults:7
```

OPTIMAL

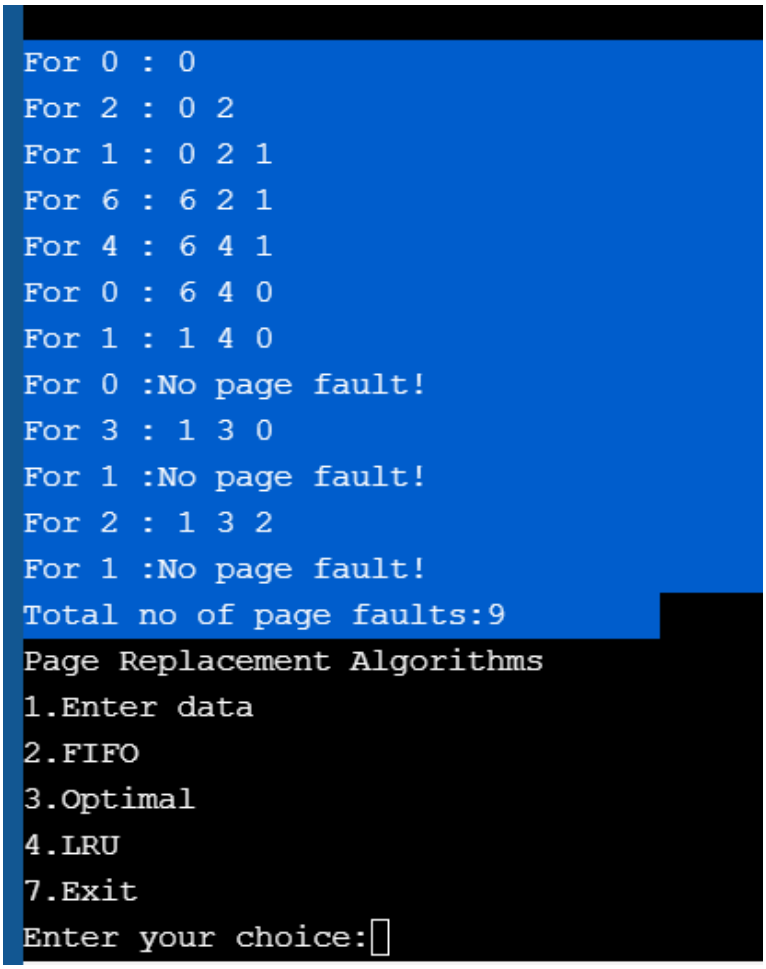
```
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
7.Exit
Enter your choice:3

For 0 : 0
For 2 : 0 2
For 1 : 0 2 1
For 6 : 0 2 1 6
For 4 : 0 2 1 4
For 0 :No page fault
For 1 :No page fault
For 0 :No page fault
For 3 : 3 2 1 4
For 1 :No page fault
For 2 :No page fault
For 1 :No page fault
Total no of page faults:6
```

Que 2.

FIFO

```
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
7.Exit
Enter your choice:1

Enter length of page reference sequence:12

Enter the page reference sequence:0 2 1 6 4 0 1 0 3 1 2 1

Enter no of frames:3

Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
7.Exit
Enter your choice:2

For 0 : 0
For 2 : 0 2
For 1 : 0 2 1
For 6 : 2 1 6
For 4 : 1 6 4
For 0 : 6 4 0
For 1 : 4 0 1
For 0 :No page fault
For 3 : 0 1 3
For 1 :No page fault
For 2 : 1 3 2
For 1 :No page fault
Total no of page faults:9
```

LRU


For 0 : 0
For 2 : 0 2
For 1 : 0 2 1
For 6 : 6 2 1
For 4 : 6 4 1
For 0 : 6 4 0
For 1 : 1 4 0
For 0 :No page fault!
For 3 : 1 3 0
For 1 :No page fault!
For 2 : 1 3 2
For 1 :No page fault!
Total no of page faults:9
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
7.Exit
Enter your choice:

OPTIMAL

```
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
7.Exit
Enter your choice:3

For 0 : 0
For 2 : 0 2
For 1 : 0 2 1
For 6 : 0 6 1
For 4 : 0 4 1
For 0 :No page fault
For 1 :No page fault
For 0 :No page fault
For 3 : 3 4 1
For 1 :No page fault
For 2 : 2 4 1
For 1 :No page fault
Total no of page faults:7
```

Name: Lakhan Kumawat
Roll No: 1906055
Branch: CSE-1.

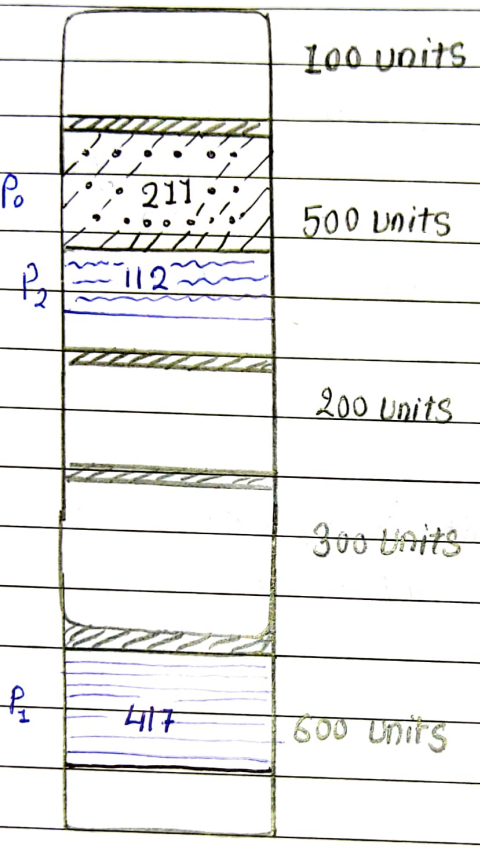
Page No.
Date

Ques. Given 5 partition of 100 units, 500 units, 200 units, 300 units and 600 units (in order), use the 1st fit, best fit, worst fit algorithm to place the process of

211 units, 417 units, 112 units, 426 units (in order)

Say P_0 P_1 P_2 P_3

(i) First-Fit:



In the first fit, the partition is allocated which is first sufficient from the top of main memory.

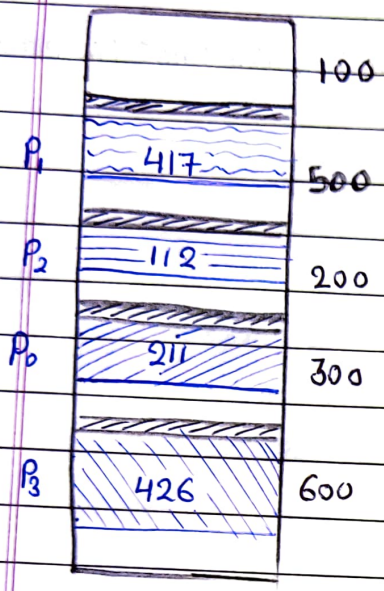
Since there is no partition of size 426 P_3 won't we allocated rest all partions of size 211, 112, & 417 units will be allocated easily.

Lakhan kumawat
1906055
CSE-1

Page No.
Date

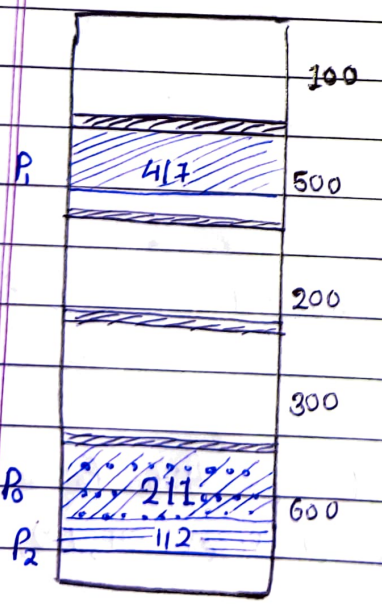
2) Best Fit :

In the best fit, the partition allocated is the smallest which is sufficient for the given process size



Here in best fit all the processes are allocated easily.
And there is No process which is not allocated.

3) Worst Fit: In the beWorst fit, the partition allocated is largest size, irrelevant of process size.



Here Except P_3 i.e. 426 all the processes are allocated.

Conclusion: Hence Best Fit. works best for the given problem Statement