Name : Ashutosh Ardu

Regno : 20BRS1262

OS LAB 10

Code

(All the three Algorithms in one code)

```
#include "stdio.h"
#include "stdlib.h"
#define pr printf
#define sc scanf

void push(int frame[],int *front,int *back,int n,int item){
  if(*front==(*back+1)%n){
    pr("Full\n");
    return;
  }
  else{
    *back=(*back+1)%n;
    frame[*back]=item;
    if(*front==-1) *front+=1;
  }
}

void pop(int frame[],int *front,int *back,int n){
  if(*front==-1){
    pr("Empty\n");
    return;
  }
  else{
    if(*front==*back){
      *front=*back=-1;
    }
    else{
      *front=(*front+1)%n;
    }
  }
}
```

```c
int in(int frame[],int n,int item){
  for(int i=0;i<n;++i){
    if(frame[i]==item) return 1;
  }return 0;
}


void print(int ar[],int front,int end,int n){
  if(front==-1){
    pr("Empty");
  }
  else if(front<=end){
    for(int i=front;i<=end;++i)
      pr("%d ",ar[i]);
  }
  else{
    for(int i=front;i<n;++i)
      pr("%d ",ar[i]);
    for(int i=0;i<=end;++i)
      pr("%d ",ar[i]);
  }pr("\n");
}

void fifo(int ar[],int n,int f){
  int frame[f];
  for(int i=0;i<f;++i)
    frame[i]=-1;
  int front,back,hit,miss;
  hit=miss=0;
  front=back=-1;
  for(int i=0;i<n;++i){
    if(in(frame,f,ar[i])){
      hit+=1;
    }
    else if(front==(back+1)%f){
      miss+=1;
      pop(frame,&front,&back,f);
      push(frame,&front,&back,f,ar[i]);
    }
    else{
      miss+=1;
      push(frame,&front,&back,f,ar[i]);
    }print(frame,front,back,f);
  }
  pr("Page Faults: %d\n",miss);
```

```c
}

int findLRU(int time[], int n){
 int i, minimum = time[0], pos = 0;
 for (i = 1; i < n; ++i){
  if (time[i] < minimum){
   minimum = time[i];
   pos = i;
  }
 }
 return pos;
}

void lru(int pages[],int n,int f){
  int frames[f];
  for(int i=0;i<f;++i)
    frames[i]=-1;
  int i,j,counter=0,flag1,flag2,faults=0,pos,time[30];
  for (i = 0; i < f; ++i){
  frames[i] = -1;
  }
 for (i = 0; i < n; ++i){
  flag1 = flag2 = 0;
  for (j = 0; j < f; ++j){
   if (frames[j] == pages[i]){
    counter++;
    time[j] = counter;
    flag1 = flag2 = 1;
    break;
   }
  }
  if (flag1 == 0){
   for (j = 0; j < f; ++j){
    if (frames[j] == -1){
     counter++;
     faults++;
     frames[j] = pages[i];
     time[j] = counter;
     flag2 = 1;
     break;
    }
   }
  }
  if (flag2 == 0){
   pos = findLRU(time, f);
```

```
      counter++;
      faults++;
      frames[pos] = pages[i];
      time[pos] = counter;
     }
     pr("\n");
     for (j = 0; j < f; ++j){
      pr("%d ", frames[j]);
     }
    }pr("\n");
    pr("Page Faults = %d\n", faults);
   }

   void optimal(int pages[],int n,int f){
     int frames[f];
     for(int i=0;i<n;++i)
       frames[i]=-1;
     int i,j,counter=0,flag1,flag2,faults=0,pos,time[30],max;
     int flag3,k;
     for (i = 0; i < f; ++i){
       frames[i] = -1;
     }
     for (i = 0; i < n; ++i){
       flag1 = flag2 = 0;
       for (j = 0; j < f; ++j){
         if (frames[j] == pages[i]){
           flag1 = flag2 = 1;
           break;
         }
       }
       if (flag1 == 0){
         for (j = 0; j < f; ++j){
           if (frames[j] == -1){
             faults++;
             frames[j] = pages[i];
             flag2 = 1;
             break;
           }
         }
       }
       if (flag2 == 0){
         flag3 = 0;
         for (j = 0; j < f; ++j){
           time[j] = -1;
           for (k = i + 1; k < n; ++k){
```

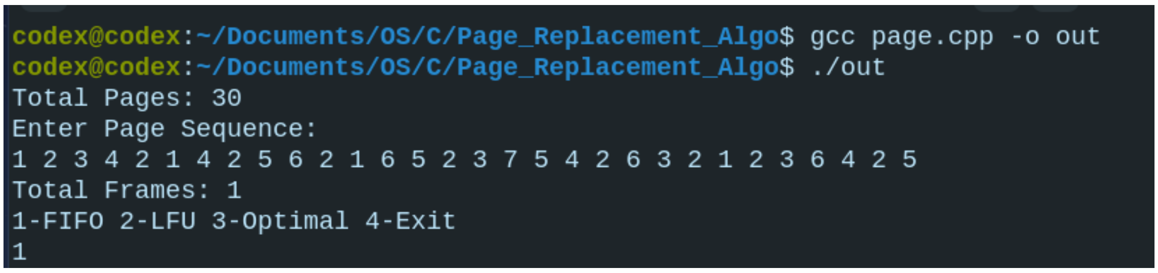
```
          if (frames[j] == pages[k]){
            time[j] = k;
            break;
          }
        }
      }
      for (j = 0; j < f; ++j){
        if (time[j] == -1){
          pos = j;
          flag3 = 1;
          break;
        }
      }
      if (flag3 == 0){
        max = time[0];
        pos = 0;
        for (j = 1; j < f; ++j){
          if (time[j] > max){
            max = time[j];
            pos = j;
          }
        }
      }
      frames[pos] = pages[i];
      faults++;
    }
    pr("\n");
    for (j = 0; j < f; ++j){
      pr("%d\t", frames[j]);
    }
  }
  pr("\n\nTotal Page Faults = %d\n", faults);
}

int main(){
  int n,ch,f;
  pr("Total Pages: ");
  sc("%d",&n);
  int pages[n];
  pr("Enter Page Sequence:\n");
  for(int i=0;i<n;++i)
    sc("%d",&pages[i]);
  do{
    pr("Total Frames: ");
    sc("%d",&f);
```

```
        pr("1-FIFO 2-LFU 3-Optimal 4-Exit\n");
        sc("%d",&ch);
        switch(ch){
          case 1:
            fifo(pages,n,f);
            break;
          case 2:
            lru(pages,n,f);
            break;
          case 3:
            optimal(pages,n,f);
            break;
        }
      }while(ch<=3);
    }
```

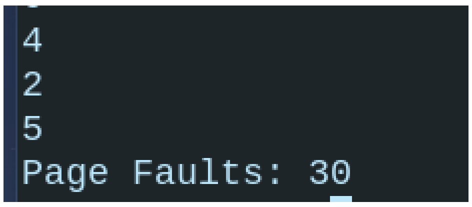
Output

Frame Size : 1

FIFO

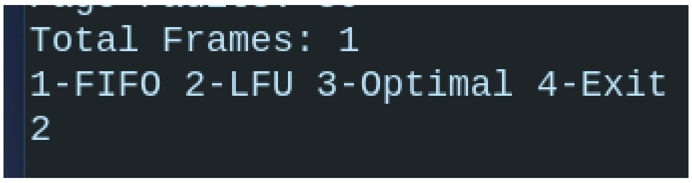


```
codex@codex:~/Documents/OS/C/Page_Replacement_Algo$ gcc page.cpp -o out
codex@codex:~/Documents/OS/C/Page_Replacement_Algo$ ./out
Total Pages: 30
Enter Page Sequence:
1 2 3 4 2 1 4 2 5 6 2 1 6 5 2 3 7 5 4 2 6 3 2 1 2 3 6 4 2 5
Total Frames: 1
1-FIFO 2-LFU 3-Optimal 4-Exit
1
```

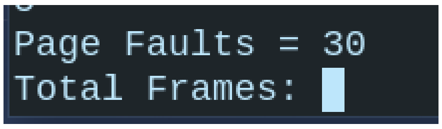


```
4
2
5
Page Faults: 30
```

LFU

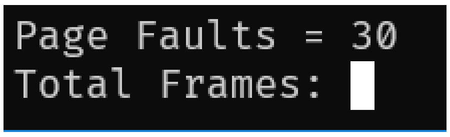


```
Total Frames: 1
1-FIFO 2-LFU 3-Optimal 4-Exit
2
```



```
Page Faults = 30
Total Frames: 
```

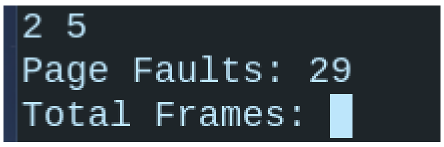
Optimal



```
Page Faults = 30
Total Frames: 
```

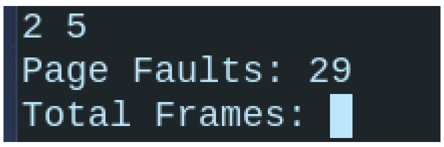
Frame size : 2

FIFO



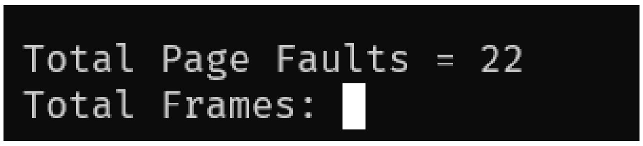
```
2 5
Page Faults: 29
Total Frames: 
```

LFU



```
2 5
Page Faults: 29
Total Frames: 
```

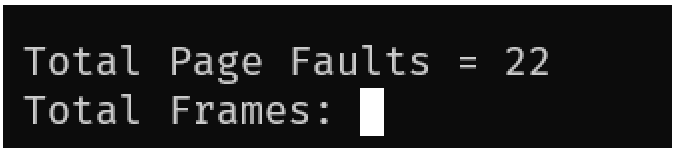
Optimal



```
Total Page Faults = 22
Total Frames:
```

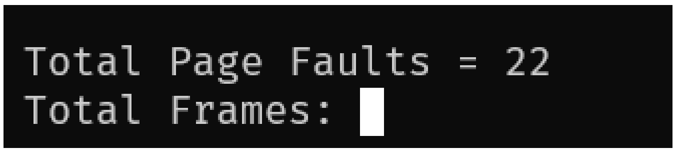
Frame size : 3

FIFO



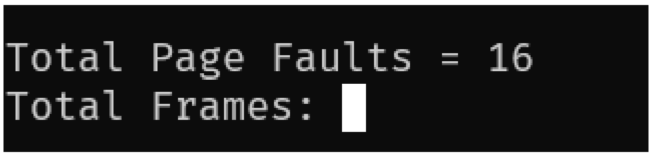
```
Total Page Faults = 22
Total Frames:
```

LFU



```
Total Page Faults = 22
Total Frames:
```

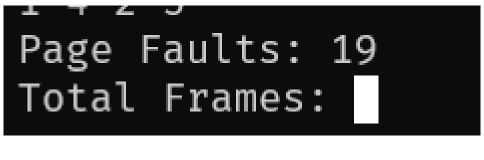
Optimal



```
Total Page Faults = 16
Total Frames:
```

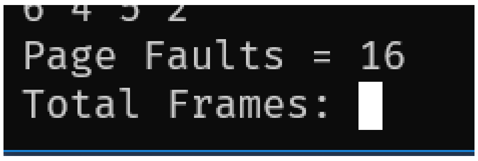
Frame size : 4

FIFO



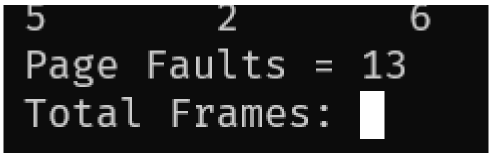
```
1 4 2 3
Page Faults: 19
Total Frames:
```

LFU



```
6 4 5 2
Page Faults = 16
Total Frames: 
```

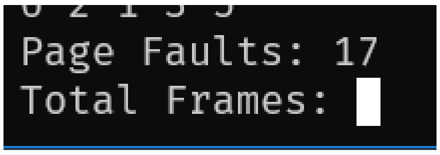
Optimal



```
5         2         6
Page Faults = 13
Total Frames: 
```

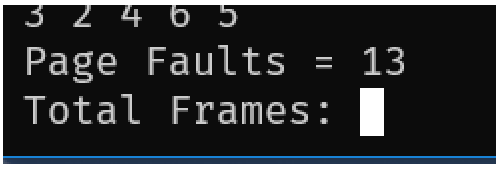
Frame size : 5

FIFO



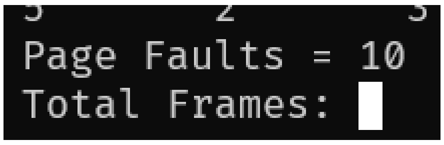
```
6 2 1 3 5
Page Faults: 17
Total Frames: 
```

LFU



```
3 2 4 6 5
Page Faults = 13
Total Frames: 
```

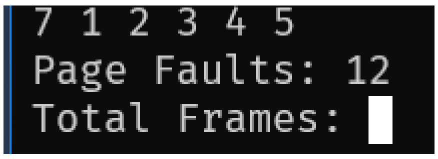
Optimal



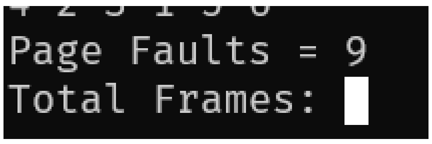
```
5         2         3
Page Faults = 10
Total Frames: 
```

Frame size : 6

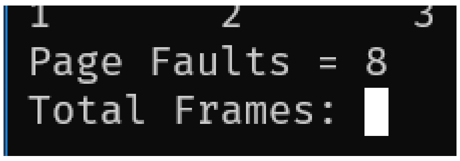
FIFO


```
 7 1 2 3 4 5
 Page Faults: 12
 Total Frames: 
```

LFU

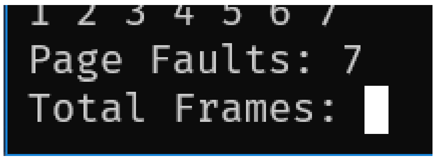

```
4 2 3 1 5 6
Page Faults = 9
Total Frames: 
```

Optimal

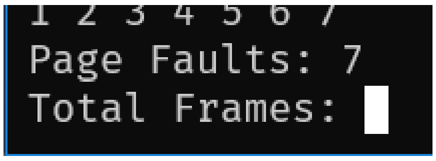

```
1        2         3
Page Faults = 8
Total Frames: 
```

Frame size : 7

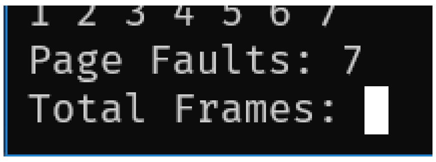
FIFO


```
 1 2 3 4 5 6 7
 Page Faults: 7
 Total Frames: 
```

LFU


```
 1 2 3 4 5 6 7
 Page Faults: 7
 Total Frames: 
```

Optimal



```
1 2 3 4 5 6 7
Page Faults: 7
Total Frames: 
```

Conclusion

Page Faults for each Algorithm:

Frame size	FIFO	LFU	Optimal
1	30	30	30
2	29	29	22
3	22	22	16
4	19	16	13
5	17	13	10
6	12	9	8
7	7	7	7

Beyond Frame size : 7 its meaning less as 7 is the Minimum possible page fault as pages are numbered from 1-7.