FCFS

```
#include<stdio.h>
int i,n,p[10],at[10],bt[10],ct[10],tat[10],wt[10],tot_tat=0,tot_wt=0;
float avg_tat,avg_wt;
int swap(int *a, int *b)
{
        int t;
        t = *a;
        *a = *b;
        *b = t;
        return 0;
}

int sort()
{
        int i,j;
        for(i=0;i<n;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(at[i]>at[j])
                        {
                                swap(&at[j],&at[i]);
                                swap(&bt[j],&bt[i]);
                                swap(&p[j],&p[i]);
                        }
                }
        }
        return 0;
}

void calculate()
{
        sort();
        //calculating completion time
        ct[0]=at[0]+bt[0]; //initial values of ct[0]

        for(i=1;i<n;i++)
        {
                if(at[i]<=ct[i-1])
                {
                        ct[i] = ct[i-1] + bt[i];
                }
                else
                {
                        ct[i] = at[i] + bt[i];
                }
        }

        //calculating the turn around time and waiting time
        for(i=0;i<n;i++)
        {
                tat[i] = ct[i]-at[i];
                wt[i] = tat[i]-bt[i];

                tot_tat += tat[i];
                tot_wt += wt[i];
        }
        avg_tat = (float)tot_tat/(float)n;
        avg_wt= (float)tot_wt/(float)n;
}
```

```
void main()
{
        //scanning the number of process
        printf("\nEnter the no of process : ");
        scanf("%d",&n);

        //Scanning the arrival time and burst time of each process
        printf("\nEnter the AT and BT of each process respectively \n");
        for(i=0;i<n;i++)
        {
                scanf("%d%d",&at[i],&bt[i]);
                p[i] = i+1;
        }

        calculate();

        // Printing the result
        printf("\nPID\tAT\tBT\tCT\tTAT\tWT\n");
        for(i=0;i<n;i++)
        {

        printf("\n%d\t%d\t%d\t%d\t%d\t%d",p[i],at[i],bt[i],ct[i],tat[i],wt[
i]);
        }

        printf("\nAverage TAT = %.2f \nAverage WT =%.2f
\n",avg_tat,avg_wt);
}
```

OUTPUT

```
Enter the no of process : 3

Enter the AT and BT of each process respectively
0       2
1       4
2       3

PID     AT      BT      CT      TAT     WT

1       0       2       2       2       0
2       1       4       6       5       1
3       2       3       9       7       4
Average TAT = 4.67
Average WT =1.67
```

SJF

```
#include<stdio.h>
#define size 10
int
n,p[size],at[size],bt[size],ct[size],tat[size],wt[size],tot_tat=0,tot_wt=0;
float avg_tat,avg_wt;
void sort()
{
        int i,j,t;
        for(i=0;i<n;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(bt[i]>bt[j])
                        {
                                //swap(&at[j],&at[i]);
                                t = at[j];
                                at[j] = at[i];
                                at[i] = t;
                                //swap(&bt[j],&bt[i]);
                                t = bt[j];
                                bt[j] = bt[i];
                                bt[i] = t;
                                //swap(&p[j],&p[i]);
                                t = p[j];
                                p[j] = p[i];
                                p[i] = t;
                        }
                        else if(bt[i]>bt[j])
                        {
                                if(at[i]>at[j])
                                {
                                        //swap(&at[j],&at[i]);
                                        t = at[j];
                                        at[j] = at[i];
                                        at[i] = t;
                                        //swap(&bt[j],&bt[i]);
                                        t = bt[j];
                                        bt[j] = bt[i];
                                        bt[i] = t;
                                        //swap(&p[j],&p[i]);
                                        t = p[j];
                                        p[j] = p[i];
                                        p[i] = t;
                                }
                        }
                }
        }
}
void calculate()
{
        int i;
        //calculating completion time
        ct[0]=at[0]+bt[0]; //initial values of ct[0]
        for(i=1;i<n;i++)
        {
                if(at[i]<=ct[i-1])
                {
                        ct[i] = ct[i-1] + bt[i];
                }
```

```
                else
                {
                        ct[i] = at[i] + bt[i];
                }
        }
        //calculating the turn around time and waiting time
        for(i=0;i<n;i++)
        {
                tat[i] = ct[i]-at[i];
                wt[i] = tat[i]-bt[i];

                tot_tat += tat[i];
                tot_wt += wt[i];
        }
        avg_tat = (float)tot_tat/(float)n;
        avg_wt= (float)tot_wt/(float)n;
        // Printing the result
        printf("\nPID\tAT\tBT\tCT\tTAT\tWT\n");
        for(i=0;i<n;i++)
        {
        printf("\n%d\t%d\t%d\t%d\t%d\t%d",p[i]+1,at[i],bt[i],ct[i],tat[i],w
t[i]);
        }
}
void main()
{
        int i;
        printf("\nEnter the no.of processes :");
        scanf("%d",&n);
        printf("Enter arrival time and burst time for each process\n");
        for(i=0;i<n;i++)
        {
                scanf("%d%d",&at[i],&bt[i]);
                p[i] = i;
        }
        sort();
        calculate();
        printf("\nAverage Turn around Time : %.2f ", avg_tat);
        printf("\nAverage Waiting Time : %.2f\n",avg_wt);
}
```

OUTPUT

```
Enter the no.of processes :4
Enter arrival time and burst time for each process
0       6
0       8
0       7
0       3

PID     AT      BT      CT      TAT     WT

4       0       3       3       3       0
1       0       6       9       9       3
3       0       7       16      16      9
2       0       8       24      24      16
Average Turn around Time : 13.00
Average Waiting Time : 7.00
```

ROUND ROBIN

```
#include<stdio.h>
#define size 10
int TRUE = 0;
int FALSE = -1;
int
n=0,qt=0,bt[size],tbt[size],tat[size],wt[size],tqt=0,time=0,lmore,tot_tat=0
,tot_wt=0;
float avg_tat,avg_wt;
void calculate()
{
        int i;
        lmore = TRUE;
        while(lmore == TRUE)
        {
                lmore = FALSE;
                for(i=0;i<n;i++)
                {
                        if(bt[i] != 0)
                                wt[i] = wt[i] + (time - tat[i]);
                        tqt = 1;
                        while(tqt <= qt && bt[i] !=0)
                        {
                                lmore = TRUE;
                                bt[i] = bt[i] -1;
                                tqt++;
                                time++;
                                tat[i] = time;
                        }
                }
        }

        printf("\nProcessor ID\tBurstTime\tTurnAroundTime\tWaitingTime\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t\t%d\t\t%d\t\t%d\n",i+1,tbt[i],tat[i],wt[i]);
                tot_tat = tot_tat + tat[i];
                tot_wt = tot_wt + wt[i];
        }
        avg_tat = (float)tot_tat/(float)n;
        avg_wt= (float)tot_wt/(float)n;

        //printf("\nTotal Turn Around Time:%d",tot_tat);
        //printf("\nTotal Waiting Time:%d",tot_wt);
}
void main()
{
        int i,j;
        printf("\nEnter no. of processors : ");
        scanf("%d",&n);
        printf("\nEnter Quantum Time : ");
        scanf("%d",&qt);
        printf("Enter burst time for each process\n");
        for(i=0;i<n;i++)
        {
                //printf("\nProcessor[%d]:",i+1);
                scanf("%d",&bt[i]);
                tbt[i] = bt[i];
                wt[i] = tat[i] = 0;
        }
```

```
        calculate();

        printf("\nAverage Turn around Time : %.2f ", avg_tat);
        printf("\nAverage Waiting Time : %.2f\n",avg_wt);
}
```

OUTPUT

```
Enter no. of processors : 4

Enter Quantum Time : 2
Enter burst time for each process
4
3
2
1

Processor ID   BurstTime      TurnAroundTime WaitingTime
1              4              9              5
2              3              10             7
3              2              6              4
4              1              7              6

Average Turn around Time : 8.00
Average Waiting Time : 5.50
```

PRIORITY

```
#define size 10
int
n,p[size],pr[size],at[size],bt[size],ct[size],tat[size],wt[size],tot_tat=0,
tot_wt=0;
float avg_tat,avg_wt;

void sort()
{
        int i,j,t;
        for(i=0;i<n;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(pr[i]>pr[j])
                        {
                                //swap(&pr[j],&pr[i]);
                                t = pr[j];
                                pr[j] = pr[i];
                                pr[i] = t;
                                //swap(&bt[j],&bt[i]);
                                t = bt[j];
                                bt[j] = bt[i];
                                bt[i] = t;
                                //swap(&p[j],&p[i]);
                                t = p[j];
                                p[j] = p[i];
                                p[i] = t;
                        }
                }
        }
}

void calculate()
{
        sort();
        int i;
        //calculating completion time
        ct[0]=at[0]+bt[0]; //initial values of ct[0]

        for(i=1;i<n;i++)
        {
                if(at[i]<=ct[i-1])
                {
                        ct[i] = ct[i-1] + bt[i];
                }
                else
                {
                        ct[i] = at[i] + bt[i];
                }
        }

        //calculating the turn around time and waiting time
        for(i=0;i<n;i++)
        {
                tat[i] = ct[i]-at[i];
                wt[i] = tat[i]-bt[i];

                tot_tat += tat[i];
                tot_wt += wt[i];
```

```
        }
        avg_tat = (float)tot_tat/(float)n;
        avg_wt= (float)tot_wt/(float)n;

        // Printing the result
        printf("\nPID\tAT\tBT\tCT\tTAT\tWT\n");
        for(i=0;i<n;i++)
        {

        printf("\n%d\t%d\t%d\t%d\t%d\t%d",p[i],at[i],bt[i],ct[i],tat[i],wt[
i]);
        }
}

void main()
{
        int i;
        printf("\nEnter the no.of processes :");
        scanf("%d",&n);
        printf("Enter priority and burst time for each process\n");
        for(i=0;i<n;i++)
        {
                scanf("%d%d",&pr[i],&bt[i]);
                p[i] = i+1;
                at[i]=0;
        }
        //sort();
        calculate();
        printf("\nAverage Turn around Time : %.2f ", avg_tat);
        printf("\nAverage Waiting Time : %.2f\n",avg_wt);
}
```

OUTPUT

```
Enter the no.of processes :3
Enter priority and burst time for each process
3       15
1       3
2       3

PID     AT      BT      CT      TAT     WT

2       0       3       3       3       0
3       0       3       6       6       3
1       0       15      21      21      6
Average Turn around Time : 10.00
Average Waiting Time : 3.00
```

BANKERS ALGORITHM

```
#include <stdio.h>

int current[5][5], maximum_claim[5][5], need[5][5], available[5];
int allocation[5] = {0, 0, 0, 0, 0};
int maxres[5], running[5], safe = 0;
int counter = 0, i, j, exec, resources, processes, k = 1;

int main()
{
        printf("\nEnter number of processes: ");
        scanf("%d", &processes);

        for (i = 0; i < processes; i++)
        {
                running[i] = 1;
                counter++;
        }

        printf("\nEnter number of resources: ");
        scanf("%d", &resources);

        printf("\nEnter Max Available Vector:");
        for (i = 0; i < resources; i++)
        {
                scanf("%d", &maxres[i]);
        }

        printf("\nEnter Allocated Resource Table:\n");
        for (i = 0; i < processes; i++)
        {
                for(j = 0; j < resources; j++)
                {
                        scanf("%d", &current[i][j]);
                }
        }

        printf("\nEnter Maximum Claim Table:\n");
        for (i = 0; i < processes; i++)
        {
                for(j = 0; j < resources; j++)
                {
                        scanf("%d", &maximum_claim[i][j]);
                }
        }

        for (i = 0; i < processes; i++)
        {
                for(j = 0; j < resources; j++)
                {
                        need[i][j] = maximum_claim[i][j] - current[i][j];
                }
        }

        printf("\nThe Max Available Vector is: ");
        for (i = 0; i < resources; i++)
        {
                printf("\t%d", maxres[i]);
        }
```

```
printf("\nThe Allocated Resource Table:\n");
for (i = 0; i < processes; i++)
{
        for (j = 0; j < resources; j++)
        {
                printf("\t%d", current[i][j]);
        }
        printf("\n");
}

printf("\nThe Maximum Claim Table:\n");
for (i = 0; i < processes; i++)
{
        for (j = 0; j < resources; j++)
        {
                printf("\t%d", maximum_claim[i][j]);
        }
        printf("\n");
}

printf("\nThe Need Table:\n");
for (i = 0; i < processes; i++)
{
        for (j = 0; j < resources; j++)
        {
                printf("\t%d", need[i][j]);
        }
        printf("\n");
}

for (i = 0; i < processes; i++)
{
        for (j = 0; j < resources; j++)
        {
                allocation[j] += current[i][j];
        }
}

printf("\nAllocated resources:");
for (i = 0; i < resources; i++)
{
        printf("\t%d", allocation[i]);
}

for (i = 0; i < resources; i++)
{
        available[i] = maxres[i] - allocation[i];
}

printf("\nAvailable resources:");
for (i = 0; i < resources; i++)
{
        printf("\t%d", available[i]);
}
printf("\n");

while (counter != 0)
{
        safe = 0;
        for (i = 0; i < processes; i++)
        {
```

```
                        if (running[i])
                        {
                                exec = 1;
                                for (j = 0; j < resources; j++)
                                {
                                        if (maximum_claim[i][j] -
current[i][j] > available[j])
                                        {
                                                exec = 0;
                                                break;
                                        }
                                }
                                if (exec)
                                {
                                        printf("\nProcess%d is executing\n",
i + 1);
                                        running[i] = 0;
                                        counter--;
                                        safe = 1;

                                        for (j = 0; j < resources; j++)
                                        {
                                                available[j] +=
current[i][j];
                                        }
                                        break;
                                }
                        }
                }
                if (!safe)
                {
                        printf("\nThe processes are in unsafe state.\n");
                        break;
                }
                else
                {
                        printf("\nThe process is in safe state");
                        printf("\nAvailable vector:");

                        for (i = 0; i < resources; i++)
                        {
                                printf("\t%d", available[i]);
                        }

                        printf("\n");
                }
        }
        return 0;
}
```

OUTPUT

```
Enter number of processes: 5

Enter number of resources: 3

Enter Max Available Vector:10  5       7

Enter Allocated Resource Table:
0       1       0
```

```
2       0       0
3       0       2
2       1       1
0       0       2

Enter Maximum Claim Table:
7       5       3
3       2       2
9       0       2
2       2       2
4       3       3

The Max Available Vector is:   10      5       7
The Allocated Resource Table:
        0       1       0
        2       0       0
        3       0       2
        2       1       1
        0       0       2

The Maximum Claim Table:
        7       5       3
        3       2       2
        9       0       2
        2       2       2
        4       3       3

The Need Table:
        7       4       3
        1       2       2
        6       0       0
        0       1       1
        4       3       1

Allocated resources:   7       2       5
Available resources:   3       3       2

Process2 is executing

The process is in safe state
Available vector:      5       3       2

Process4 is executing

The process is in safe state
Available vector:      7       4       3

Process1 is executing

The process is in safe state
Available vector:      7       5       3

Process3 is executing

The process is in safe state
Available vector:      10      5       5

Process5 is executing

The process is in safe state
Available vector:      10      5       7
```

FCFS DISK SCHEDULING

```
#include<stdio.h>
void main()
{
        int t[20], tohm[20],n, h, i, j, tot=0;
        float avhm;
        printf("\nEnter the no. of tracks : ");
        scanf("%d", &n);
        printf("\nEnter the position of head : ");
        scanf("%d", &h);
        t[0] = 0;
        t[1] = h;
        printf("\nEnter the track to be traversed\n");
        for(i=2; i<n+2; i++)
                scanf("%d",&t[i]);
        for(i=1;i<n+1;i++)
        {
                tohm[i]=t[i+1]-t[i];
                if(tohm[i]<0)
                        tohm[i] *= (-1);
        }
        for(i=1;i<n+1;i++)
                tot += tohm[i];
        avhm = (float)tot/n;
        printf("\n Track traversed\tDifference b/w tracks");
        for(i=1;i<n+1;i++)
                printf("\n%d\t\t\t%d",t[i+1],tohm[i]);
        printf("\nAverage header movement = %.2f",avhm);
        printf("\n");
}
```

OUTPUT

```
Enter the no. of tracks : 8

Enter the position of head : 50

Enter the track to be traversed
95 180 34 119 11 123 62 64

 Track traversed       Difference b/w tracks
95                     45
180                    85
34                     146
119                    85
11                     108
123                    112
62                     61
64                     2
Average header movement = 80.50
```

SCAN DISK SCHEDULING

```
#include<stdio.h>
void main()
{
        int t[20],d[20],h,i,j,n,temp,k,atr[20],tot,p,sum=0;
        float avhm;
        printf("\nEnter the no. of tracks to be traversed : ");
        scanf("%d",&n);

        printf("\nEnter the position of head : ");
        scanf("%d",&h);

        t[0]=0;
        t[1]=h;

        printf("Enter the tracks : ");
        for(i=2;i<n+2;i++)
                scanf("%d",&t[i]);

        for(i=0;i<n+2;i++)
        {
                for(j=0;j<(n+2)-i-1;j++)
                {
                        if(t[j]>t[j+1])
                        {
                                temp=t[j];
                                t[j]=t[j+1];
                                t[j+1]=temp;
                        }
                }
        }
        for(i=0;i<n+2;i++)
                if(t[i]==h)
                {
                        j=i;
                        k=i;
                }
        p=0;
        while(t[j]!=0)
        {
                atr[p]=t[j];
                j--;
                p++;
        }
        atr[p]=t[j];

        for(p=k+1;p<n+2;p++,k++)
                atr[p]=t[k+1];

        for(j=0;j<n+1;j++)
        {
                if(atr[j]>atr[j+1])
                        d[j]=atr[j]-atr[j+1];
                else
                         d[j]=atr[j+1]-atr[j];
                sum+=d[j];
        }
        avhm = (float)sum/n;
        printf("\n Track traversed\tDifference b/w tracks");
        for(i=0;i<n+1;i++)
```

```
                printf("\n%d\t\t\t%d",atr[i+1],d[i]);
        printf("\nTotal header movement = %d",sum);
        printf("\nAverage header movement = %.2f",avhm);
        printf("\n");
}
```

OUTPUT

```
Enter the no. of tracks to be traversed : 8

Enter the position of head : 50
Enter the tracks : 176 79      34      60      92      11      41      114

 Track traversed       Difference b/w tracks
41                     9
34                     7
11                     23
0                      11
60                     60
79                     19
92                     13
114                    22
176                    62
Total header movement = 226
Average header movement = 28.25
```

CSCAN DISK SCHEDULING

```
#include<stdio.h>
void main()
{
        int t[20],d[20],h,i,j,n,temp,k,atr[20],tot,p,sum=0;
        float avhm;
        printf("\nEnter the no. of tracks to be traversed : ");
        scanf("%d",&n);

        printf("\nEnter the position of head : ");
        scanf("%d",&h);

        t[0]=0;
        t[1]=h;

        printf("\nEnter the total tracks : ");
        scanf("%d",&tot);
        t[2]  = tot-1;

        printf("Enter the tracks : ");
        for(i=3;i<n+3;i++)
                scanf("%d",&t[i]);

        for(i=0;i<n+3;i++)
        {
                for(j=0;j<(n+3)-i-1;j++)
                {
                        if(t[j]>t[j+1])
                        {
                                temp=t[j];
                                t[j]=t[j+1];
                                t[j+1]=temp;
                        }
                }
        }
        for(i=0;i<n+3;i++)
                if(t[i]==h)
                {
                        j=i;
                        //k=i;
                        break;
                }
        p=0;
        while(t[j]!=tot-1)
        {
                atr[p]=t[j];
                j++;
                p++;
        }
        atr[p]=t[j];
        p++;
        i=0;

        while(p!=(n+3) && t[i]!=h)
        {
                atr[p]=t[i];
                i++;
                p++;
        }
```

```
        for(j=0;j<n+2;j++)
        {
                if(atr[j]>atr[j+1])
                        d[j]=atr[j]-atr[j+1];
                else
                         d[j]=atr[j+1]-atr[j];
                sum+=d[j];
        }
        avhm = (float)sum/n;
        printf("\n Track traversed\tDifference b/w tracks");
        for(i=0;i<n+2;i++)
                printf("\n%d\t\t\t%d",atr[i+1],d[i]);
        printf("\nTotal header movement = %d",sum);
        printf("\nAverage header movement = %.2f",avhm);
        printf("\n");
}
```

OUTPUT

```
Enter the no. of tracks to be traversed : 8

Enter the position of head : 50

Enter the total tracks : 200
Enter the tracks : 176 79      34      60      92      11      41      114

 Track traversed       Difference b/w tracks
60                     10
79                     19
92                     13
114                    22
176                    62
199                    23
0                      199
11                     11
34                     23
41                     7
Total header movement = 389
Average header movement = 48.62
```

FIFO

```
#include <stdio.h>
int main()
{
	int referenceString[10], pageFaults = 0, m, n, s, pages, frames;
	printf("\nEnter the number of Pages:\t");
	scanf("%d", &pages);
	printf("\nEnter reference string values:\t");
	for(m = 0; m < pages; m++)
	{
		// printf("Value No. [%d]:\t", m + 1);
		 scanf("%d", &referenceString[m]);
	}
	printf("\n What are the total number of frames:\t");
	{
		scanf("%d", &frames);
	}
	int temp[frames];
	for(m = 0; m < frames; m++)
	{
		temp[m] = -1;
	}
	for(m = 0; m < pages; m++)
	{
		s = 0;
		for(n = 0; n < frames; n++)
		{
			if(referenceString[m] == temp[n])
			{
				s++;
				pageFaults--;
			}
		}
		pageFaults++;
		if((pageFaults <= frames) && (s == 0))
		{
			temp[m] = referenceString[m];
		}
		else if(s == 0)
		{
			temp[(pageFaults - 1) % frames] = referenceString[m];
		}
		printf("\n");
		for(n = 0; n < frames; n++)
		{
			printf("%d\t", temp[n]);
		}
	}
	printf("\nTotal Page Faults:\t%d\n", pageFaults);
	return 0;
}
```

OUTPUT

```
Enter the number of Pages:    5

Enter reference string values: 4      1      2      4      5

 What are the total number of frames: 3
```

```
4       -1      -1
4       1       -1
4       1       2
4       1       2
5       1       2
Total Page Faults:     4
```

LRU

```
#include<stdio.h>

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;
}

int main()
{
    int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0,
time[10], flag1, flag2, i, j, pos, faults = 0;
printf("Enter number of frames: ");
scanf("%d", &no_of_frames);
printf("Enter number of pages: ");
scanf("%d", &no_of_pages);
printf("Enter reference string: ");
    for(i = 0; i < no_of_pages; ++i){
     scanf("%d", &pages[i]);
    }

for(i = 0; i < no_of_frames; ++i){
     frames[i] = -1;
    }

    for(i = 0; i < no_of_pages; ++i){
     flag1 = flag2 = 0;

     for(j = 0; j < no_of_frames; ++j){
     if(frames[j] == pages[i]){
     counter++;
     time[j] = counter;
   flag1 = flag2 = 1;
   break;
   }
     }

     if(flag1 == 0){
for(j = 0; j < no_of_frames; ++j){
     if(frames[j] == -1){
     counter++;
     faults++;
     frames[j] = pages[i];
     time[j] = counter;
     flag2 = 1;
     break;
     }
     }
     }

     if(flag2 == 0){
     pos = findLRU(time, no_of_frames);
     counter++;
```

```
      faults++;
      frames[pos] = pages[i];
      time[pos] = counter;
      }

      printf("\n");

      for(j = 0; j < no_of_frames; ++j){
      printf("%d\t", frames[j]);
      }
}
printf("\n\nTotal Page Faults = %d", faults);

    return 0;
}
```

OUTPUT

```
Enter number of frames: 3
Enter number of pages: 6
Enter reference string: 5      7       5       6       7       3

5       -1      -1
5       7       -1
5       7       -1
5       7       6
5       7       6
3       7       6

Total Page Faults = 4
```

LFU

```
#include<stdio.h>
int main()
{
        int f,p;
        int pages[50],frame[10],hit=0,count[50],time[50];
        int i,j,page,flag,least,minTime,temp;
        printf("Enter no of frames : ");
        scanf("%d",&f);
        printf("Enter no of pages : ");
        scanf("%d",&p);
        for(i=0;i<f;i++)
        {
                frame[i]=-1;
        }
        for(i=0;i<50;i++)
        {
                count[i]=0;
        }
        printf("Enter page no : \n");
        for(i=0;i<p;i++)
        {
                scanf("%d",&pages[i]);
        }
        printf("\n");
        for(i=0;i<p;i++)
        {
                count[pages[i]]++;
                time[pages[i]]=i;
                flag=1;
                least=frame[0];
                for(j=0;j<f;j++)
                {
                        if(frame[j]==-1 || frame[j]==pages[i])
                        {
                                if(frame[j]!=-1)
                                {
                                        hit++;
                                }
                                flag=0;

                                frame[j]=pages[i];
                                break;
                        }
                        if(count[least]>count[frame[j]])
                        {
                                least=frame[j];
                        }
                }
                if(flag)
                {
                        minTime=50;
                        for(j=0;j<f;j++)
                        {
                                if(count[frame[j]]==count[least] &&
time[frame[j]]<minTime)
                                {
                                        temp=j;
                                        minTime=time[frame[j]];
                                }
```

```
                        }
                        count[frame[temp]]=0;
                        frame[temp]=pages[i];
                }
                for(j=0;j<f;j++)
                {
                        printf("%d ",frame[j]);
                }
                printf("\n");
        }
        printf("Page hit = %d",hit);
        return 0;
}
```

OUTPUT

```
Enter no of frames : 3
Enter no of pages : 7
Enter page no :
1       2       3       4       2       1       5

1 -1 -1
1 2 -1
1 2 3
4 2 3
4 2 3
4 2 1
5 2 1
Page hit = 1
```