1) TO SHOW THE IMPLEMENTATION OF FORK

PROGRAM:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
//make two process which run some
//program after this instruction
fork();
printf("Hello world!\n");
return 0;
}
```

2) TO SHOW THE IMPLEMENTATION OF FIRST COME FIRST SERVED (FCFS) ALGORITHM

PROGRAM:

// A program to implement First Come First Served (FCFS) algorithm.

```
#include<stdio.h>
int main(){
int n,bt[20], wt[20],tat[20], avwt=0, avtat=0,i,j;
printf("Enter total number of processes: ");
scanf("%d",&n);
printf("\nEnter Process Burst Time\n");
for(i=0;i<n;i++)
{
printf("P[%d]:",i+1);
scanf("%d",&bt[i]);
}
wt[0]=0;
for(i=1;i<n;i++)
{
wt[i]=0;
for(j=0;j<i;j++)
wt[i]+=bt[j];
}
printf("\nProcess\t\tBurst Time\t Waiting Time\t Turnaround Time");
for(i=0;i<n;i++)
{
tat[i]=bt[i]+wt[i];
avtat+=tat[i];
avtat+=tat[i];
printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i], wt[i], tat[i]);
}
avwt/=i;
avtat/=i;
printf("\n\nAverage Waiting Time:%d",avwt);
printf("\nAverage Turnaround Time:%d",avtat);
return 0;
}
```

3) TO SHOW THE IMPLEMENTATION OF SHORTEST JOB FIRST (SJF) ALGORITHM

PROGRAM:

```
//A program to implement Shortest Job First (SJF) algorithm.

#include<stdio.h>
int main()
{
int bt[20],p[20], wt[20], tat[20],i,j,n,total=0,pos,temp;
float avg_wt,avg_tat;
printf("Enter number of process:");
scanf("%d",&n);
printf("\nEnter Burst Time for Process: \n");
for(i=0;i<n;i++)
{
printf("p%d:",i+1);
scanf("%d",&bt[i]);
p[i]=i+1;
}
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(bt[j]<bt[pos])
pos=j;
}
temp=bt[i];
bt[i]=bt[pos];
bt[pos]=temp;
temp=p[i];
p[i]=p[pos];
p[pos]=temp;
}
wt[0]=0;
for(i=1;i<n;i++)
{
wt[i]=0;
for(j=0;j<i;j++)
wt[i]+=bt[j];
total+=wt[i];
}
avg_wt=(float)total/n;
total=0;
printf("\nProcess \t Burst Time \t Waiting Time \t Turnaround Time");
for(i=0;i<n;i++)
{
tat[i]=bt[i]+wt[i];
total+=tat[i];
printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i], wt[i],tat[i]);
}
avg_tat=(float)total/n;
printf("\n\nAverage Waiting Time=%f",avg_wt);
printf("\n\nAverage Turnaround Time=%f",avg_tat);
}
```

4) TO SHOW THE IMPLEMENTATION OF ROUND ROBIN (RR) ALGORITHM.

PROGRAM:

```
//Write a program to show the implementation of Round Robin (RR) algorithm.

#include<stdio.h>
int main()
{
int i, limit, total = 0, x, counter = 0, time_quantum;
int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10], temp[10];
float average_wait_time, average_turnaround_time;
printf("\nEnter Total Number of Processes:");
scanf("%d", &limit);
x= limit;
for(i=0; i<limit; i++)
{
printf("\nEnter Details of Process[%d]\n",i+1);
printf("Arrival Time:\t");
scanf("%d", &arrival_time[i]);
printf("Burst Time:\t");
scanf("%d", &burst_time[i]);
temp[i]=burst_time[i];
}
printf("\nEnter Time Quantum:\t");
scanf("%d", &time_quantum);
printf("\nProcess ID\t\tBurst Time\t Turnaround Time \tWaiting Time\n");
for(total=0,i=0;x!=0;)
{
if(temp[i] <=time_quantum && temp[i] >0)
{
total= total +temp[i];
temp[i]=0;
counter=1;
}
else if(temp[i]>0)
{
temp[i] = temp[i]-time_quantum;
total = total +time_quantum;
}
if(temp[i] == 0 && counter == 1)
{
   x--;
printf("\nProcess[%d]\t\t%d\t\t%d\t\t\t%d", i + 1, burst_time[i],total-arrival_time[i], total-arrival_time[i]-
burst_time[i]);
wait_time = wait_time + total-arrival_time[i]-burst_time[i];
turnaround_time = turnaround_time+total-arrival_time[i];
counter = 0;
}
if(i== limit-1)
{
i=0;
}
else if(arrival_time[i + 1] <= total)
{
i++;
}
else
{
i=0;
}
}
average_wait_time = wait_time* 1.0/limit;
average_turnaround_time = turnaround_time* 1.0/limit;
printf("\n\nAverage Waiting Time:\t%f", average_wait_time);
printf("\nAvg Turnaround Time:\t%f\n", average_turnaround_time);
return 0;
}
```

5) TO SHOW THE IMPLEMENTATION OF FIFO PAGE REPLACEMENT ALGORITHM.

PROGRAM:

```
#include <stdlib.h>
#include <stdio.h>

int pagefault(int a[], int frame[], int n, int no)
{
   int i, j, avail, count = 0, k;
   for (i = 0; i < no; i++)
```

```
    {
        frame[i] = -1;
    }
    j = 0;
    for (i = 0; i < n; i++)
    {
        avail = 0;
        for (k = 0; k < no; k++)
            if (frame[k] == a[i])
                avail = 1;
        if (avail == 0)
        {
            frame[j] = a[i];
            j = (j + 1) % no;
            count++;
        }
    }
    return count;
}
int main()
{
    int n, i, *a, *frame, no, fault;
    printf("\nENTER THE NUMBER OF PAGES:\n");
    scanf("%d", &n);

    a = (int *)malloc(n * sizeof(int));
    printf("ENTER THE PAGE NUMBER :\n");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

    printf("ENTER THE NUMBER OF FRAMES :");
    scanf("%d", &no);

    frame = (int *)malloc(no * sizeof(int));
    fault = pagefault(a, frame, n, no);
    printf("Page Fault Is %d", fault);
}
```

6) TO SHOW THE IMPLEMENTATION OF LRU PAGE REPLACEMENT ALGORITHM.

PROGRAM:

```
// LRU Page Replacement Algorithm

#include <stdio.h>
int n, ref[100], fs, frame[100], count = 0;
void input();
void show();
void cal();
int main() {
 printf("********** LRU Page Replacement Algo ************** \n");
 input();
 cal();
 show();
}

void input() {
 int i;
 printf("Enter no of pages in Refrence String\t");
 scanf("%d", &n);
 printf("Enter the reference string:");
 for (i = 0; i < n; i++)
  scanf("%d", &ref[i]);
 printf("Enter the Frame Size\t");
 scanf("%d", &fs);
}
void cal() {
 int i, j, k = 0, c1, c2[100], r, temp[100], t;
 frame[k] = ref[k];
 count++;
 k++;
 for (i = 1; i < n; i++) {
  c1 = 0;
  for (j = 0; j < fs; j++) {
   if (ref[i] != frame[j])
    c1++;
  }
  if (c1 == fs) {
   count++;
   if (k < fs) {
    frame[k] = ref[i];
    k++;
   } else {
    for (r = 0; r < fs; r++) {
     c2[r] = 0;
     for (j = i - 1; j < n; j--) {
      if (frame[r] != ref[j])
       c2[r]++;
      else
       break;
     }
    }
    for (r = 0; r < fs; r++)
     temp[r] = c2[r];
    for (r = 0; r < fs; r++) {
     for (j = r; j < fs; j++) {
      if (temp[r] < temp[j]) {
            t = temp[r];
            temp[r] = temp[j];
            temp[j] = t;
          }
        }
      }
      for (r = 0; r < fs; r++) {
        if (c2[r] == temp[0])
          frame[r] = ref[i];
      }
    }
  }
 }
}
void show() { printf("Page Faults = %d", count);
}
```

7) TO SHOW THE IMPLEMENTATION OF FCFS DISK SCHEDULING ALGORITHM.

PROGRAM:

```
//FCFS Disk Scheduling Algorithm

#include <stdio.h>
#include <stdlib.h>
int main()
{
int queue [100],n, head,i,j,seek=0,diff;
float avg;
printf("*** FCFS Disk Scheduling Algorithm***\n");
printf("Enter the size of Queue\t");
scanf("%d",&n);
printf("Enter the Queue\t");
for(i=1;i<=n;i++)
{
scanf("%d",&queue[i]);
}
printf("Enter the initial head position\t");
scanf("%d",&head);
queue[0]=head;
printf("\n");
for(j=0;j<=n-1;j++)
{
diff=abs(queue[j+1]-queue[j]);
seek+=diff;
printf("Move from %d to %d with Seek %d\n", queue[j], queue[j+1], diff);
}
printf("\nTotal Seek Time is %d\t",seek);
}
```