Ex. No.: 6a)
Date: 20.02.2025

FIRST COME FIRST SERVE

Aim:

To implement First-come First- serve (FCFS) scheduling.

Program:

```c
#include <stdio.h>
int main() {
    int n,i,j,bt[10],wt[10],tat[10],total_wt=0,total_tat=0;
    printf("Enter the number of processes: ");
    scanf("%d",&n);
    printf("Enter the burst time of the processes: ");
    for(i=0;i<n;i++) scanf("%d",&bt[i]);
    wt[0]=0;
    for(i=1;i<n;i++) wt[i]=bt[i-1]+wt[i-1];
    for(i=0;i<n;i++) tat[i]=bt[i]+wt[i];
    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for(i=0;i<n;i++) printf("%d\t%d\t\t%d\t\t%d\n",i,bt[i],wt[i],tat[i]);
    for(i=0;i<n;i++) {
        total_wt+=wt[i];
        total_tat+=tat[i];
    }
    printf("Average waiting time is: %.2f\n",(float)total_wt/n);
    printf("Average Turnaround Time is: %.2f\n",(float)total_tat/n);
    return 0;
}
```

Output:

```
Enter the number of processes: 3
Enter the burst time of the processes: 24 3 3
Process Burst Time  Waiting Time    Turnaround Time
   0        24          0                24
   1        3           24               27
   2        3           27               30
Average waiting time is: 17.00
Average Turnaround Time is: 27.00
```

Result:

The program implements the First-Come-First-Serve (FCFS) scheduling technique, calculating the waiting time, turnaround time, and averages and executed successfully.

Ex. No.: 6b)
Date: 26.02.2025

SHORTEST JOB FIRST

Aim:

To implement the Shortest Job First (SJF) scheduling.

Program:

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int n,i,j;
    printf("Enter the number of processes: ");
    scanf("%d",&n);
    int burst_time[n],waiting_time[n],turnaround_time[n],pid[n];
    int total_wt=0,total_tat=0;
    printf("Enter the burst time of the processes: ");
    for(i=0;i<n;i++){
        pid[i]=i;
        scanf("%d",&burst_time[i]);
        waiting_time[i]=0;
        turnaround_time[i]=0;
    }
    for(i=0;i<n-1;i++){
        for(j=i+1;j<n;j++){
            if(burst_time[i]>burst_time[j]){
                int temp=burst_time[i];
                burst_time[i]=burst_time[j];
                burst_time[j]=temp;
                temp=pid[i];
                pid[i]=pid[j];
                pid[j]=temp;
            }
        }
    }
    for(i=1;i<n;i++){
        waiting_time[i]=burst_time[i-1]+waiting_time[i-1];
    }
    for(i=0;i<n;i++){
        turnaround_time[i]=burst_time[i]+waiting_time[i];
    }
    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for(i=0;i<n;i++){
        printf("%d\t%d\t\t%d\t\t%d\n",pid[i],burst_time[i],waiting_time[i],turnaround_time[i]);
    }
    for(i=0;i<n;i++){
        total_wt+=waiting_time[i];
        total_tat+=turnaround_time[i];
    }
    printf("Average waiting time is: %.2f\n",(float)total_wt/n);
    printf("Average Turnaround Time is: %.2f\n",(float)total_tat/n);
    return 0;
}
```

Output:

```
Enter the number of processes: 4
Enter the burst time of the processes: 8 4 9 5
Process Burst Time  Waiting Time    Turnaround Time
   1        4            0                4
   3        5            4                9
   0        8            9                17
   2        9            17               26
Average waiting time is: 7.50
Average Turnaround Time is: 14.00
```

Result:

The program implements the Shortest Job First (SJF) scheduling technique, calculating the waiting time, turnaround time, and averages, and executed successfully.

Ex. No.: 6c)
Date: 27.02.2025

PRIORITY SCHEDULING

Aim:

To implement priority scheduling technique.

Program:

```c
#include <stdio.h>
int main(){
    int n,i,j;
    printf("Enter the number of processes: ");
    scanf("%d",&n);
    int bt[n],wt[n],tat[n],p[n],pri[n],total_wt=0,total_tat=0;
    printf("Enter the burst time of the processes: ");
    for(i=0;i<n;i++){
        scanf("%d",&bt[i]);
        p[i]=i;
    }
    printf("Enter the priority of the processes: ");
    for(i=0;i<n;i++) scanf("%d",&pri[i]);
    for(i=0;i<n-1;i++){
        for(j=i+1;j<n;j++){
            if(pri[i]>pri[j]){
                int temp=pri[i];pri[i]=pri[j];pri[j]=temp;
                temp=bt[i];bt[i]=bt[j];bt[j]=temp;
                temp=p[i];p[i]=p[j];p[j]=temp;
            }
        }
    }
    wt[0]=0;
    for(i=1;i<n;i++) wt[i]=bt[i-1]+wt[i-1];
    for(i=0;i<n;i++) tat[i]=bt[i]+wt[i];
    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for(i=0;i<n;i++){
        printf("%d\t%d\t\t%d\t\t%d\n",p[i],bt[i],wt[i],tat[i]);
        total_wt+=wt[i];
        total_tat+=tat[i];
    }
    printf("Average waiting time is: %.2f\n",(float)total_wt/n);
    printf("Average Turnaround Time is: %.2f\n",(float)total_tat/n);
    return 0;
}
```

Output:

```
Enter the number of processes: 4
Enter the burst time of the processes: 8 4 9 5
Enter the priority of the processes: 3 1 4 2
Process Burst Time  Waiting Time    Turnaround Time
   1        4             0                4
   3        5             4                9
   0        8             9                17
   2        9             17               26
Average waiting time is: 7.50
Average Turnaround Time is: 14.00
```

Result:

The program implements the Priority Scheduling technique, calculating waiting time, turnaround time, and averages, and executed successfully.

Ex. No.: 6d)
Date: 26.03.2025
ROUND ROBIN SCHEDULING

Aim:

To implement the Round Robin (RR) scheduling technique.

Program:

```c
#include <stdio.h>
int main(){
    int n,i,tq;
    printf("Enter the number of processes: ");
    scanf("%d",&n);
    int bt[n],wt[n],tat[n],rem_bt[n],p[n];
    printf("Enter the burst time of the processes: ");
    for(i=0;i<n;i++){
        scanf("%d",&bt[i]);
        rem_bt[i]=bt[i];
        p[i]=i;
    }
    printf("Enter the time quantum: ");
    scanf("%d",&tq);
    int t=0,done;
    while(1){
        done=1;
        for(i=0;i<n;i++){
            if(rem_bt[i]>0){
                done=0;
                if(rem_bt[i]>tq){
                    t+=tq;
                    rem_bt[i]-=tq;
                }else{
                    t+=rem_bt[i];
                    wt[i]=t-bt[i];
                    rem_bt[i]=0;
                }
            }
        }
        if(done==1) break;
    }
    int total_wt=0,total_tat=0;
    for(i=0;i<n;i++){
        tat[i]=bt[i]+wt[i];
        total_wt+=wt[i];
        total_tat+=tat[i];
    }
    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for(i=0;i<n;i++) printf("%d\t%d\t\t%d\t\t%d\n",p[i],bt[i],wt[i],tat[i]);
    printf("Average waiting time is: %.2f\n",(float)total_wt/n);
    printf("Average Turnaround Time is: %.2f\n",(float)total_tat/n);
    return 0;
}
```

Output:

```
Enter the number of processes: 4
Enter the burst time of the processes: 8 4 9 5
Enter the time quantum: 3
Process Burst Time  Waiting Time     Turnaround Time
   0        8            15                 23
   1        4            12                 16
   2        9            17                 26
   3        5            16                 21
Average waiting time is: 15.00
Average Turnaround Time is: 21.50
```

Result:

The program implements the Round Robin Scheduling technique, calculates waiting time, turnaround time, averages, and executed successfully.