Ex. No.: 6a)
Date: 21|2|25

FIRST COME FIRST SERVE

Aim:

To unplement First-come First- serve (FCFS) scheduling technique

Algorithm:

1. Get the number of processes from the user.
2. Read the process name and burst time.
3. Calculate the total process time.
4. Calculate the total waiting time and total turnaround time for each process 5. Display the process name & burst time for each process. 6. Display the total waiting time, average waiting time, turnaround time

Program Code:

```
#include <stdio.h>
int main(){
    int n,i;
    float total=0, float totall =0
    printf("Enter the no of processes");
    scanf("%d",&n);
    int bt[n],wt[n],tat[n];
    char process[n][10];
    printf("Enter process:");
    for(i=0; i<n; i++){
        scanf("%s", process[i]);
    }
    printf(" Enter burst time ")
    for(i=0; i<n; i++){
        Scanf("%d", &bt[i]);

    }
```

```
wt[0]=0;
for(i=1; i<n; i++){
    wt[i]=wt[i-1]+bt[i-1]
    total += tat[i];
printf("Process\t Bursttime\t Waiting \t Turnaround time\n");
for(i=0; i<n; i++){
    printf("%s\t %d\t %d\t\n", process[i], bt[i], wt[i], tat[i
    }
  printf("\n Average Waiting time " %.2f ", total);
  printf("\n Average Turn around time %.2f ", totalt );

  return 0;

}
```

Sample Output:

```
Enter the number of process:
3
Enter the burst time of the processes:
24 3 3

Process     Burst Time     Waiting Time   Turn Around Time

   0           24              0               24

   1            3             24               27

   2            3             27               30

Average waiting time is:  17.0
Average Turn around Time is:  19.0
```

```
Enter the number of processes: 4
Enter process names: p1 p2 p3 p4
Enter the burst time of the processes : 5 3 8 6

Process     Burst Time     Waiting time     turn around time
  P1           5              0                 5
  P2           3              5                 8
  P3           8              8                16
  P4           6             11                22

Average waiting time: 7.25
Average turn around time: 12.75
```

[illegible]

Result:

Thus the code to implement First Come First Serve (FCFS) has been executed successfully

Ex. No.: 6b)
Date: 21|2|25

SHORTEST JOB FIRST

Aim:

To implement the Shortest Job First (SJF) scheduling technique

Algorithm:

1. Declare the structure and its elements.
2. Get number of processes as input from the user.
3. Read the process name, arrival time and burst time
4. Initialize waiting time, turnaround time & flag of read processes to zero. 5. Sort based on burst time of all processes in ascending order 6. Calculate the waiting time and turnaround time for each process. 7. Calculate the average waiting time and average turnaround time. 8. Display the results.

Program Code:

```
#include <stdio.h>
int main() {
    int n,i,j,temp;
    printf(" Enter the number of processes:");
    scanf ("%d", &n);
    int p[n], bt[n],wt[n], tat[n]
    float t_wt =0 ,ttat=0;
    printf(" Enter the burst times: \n");
    for(i=0; i<n; i++ ){
        ~~p[i]=~~
        scanf("%d", &bt[i]);
    }

    for(i=0;i<n-1;i++)
        for(j=0 ;j<n-i -1 ;j++){
            if(bt[j] > bt[j+1]){
                temp = bt[j ];
                bt [j] = bt [j+1 ];
                bt [j+1 ]= temp ;
            }
        }
    }
```

```
wt[0]=0;
for (i=1; i<n; i++){
        wt[i]=wt[i-1]+bt[i-1];
}

for (i=0; i<n; i++){
      tat[i]=wt[i]+bt[i];
      totalwt += wt[i];
      total-tat= tat[i];
}

printf (" \n Process   \t Burst Time \t Waiting Time \t  Turn Around Time\n");
for (i=0; i<n; i++){
      printf("%d\t\t %d\t\t %d \t \t %d\n", pid[i], bt[i],
   wt[i], tat[i]);
    }
    prinf("\n Average waiting time is : %.1f \n", total-wt /n);
    printf("Average Turn Around Time is  %.1f\n", total-tat /n);

    return0;
}
```

Sample Output:

```
Enter the number of process:
4
Enter the burst time of the processes:
8 4 9 5

Process      Burst Time      Waiting Time    Turn Around Time

   2             4                0                4
   4             5                4                9
   1             8                9               17
   3             9               17               26

Average waiting time is: 7.5
Average Turn Around Time is: 13.0
```

```
Enter the number of processes:
5
Enter the burst times:
6 2 8 3 4

Process      Burst Time      Waiting Time      Turn Around Time

  2              2                0                  2
  4              3                2                  5
  5              4                5                  9
  1              6                9                 15
  3              8               15                 23

Average waiting time is:6.2
Average turn around time is: 10.8
```

Result:

Thus the code to implement shortest job first has been executed successfully

[illegible]

Ex. No.: 6c)
Date: 8|3|25

PRIORITY SCHEDULING

Aim:

To implement priority scheduling technique

Algorithm:

1. Get the number of processes from the user.
2. Read the process name, burst time and priority of process.
3. Sort based on burst time of all processes in ascending order based priority 4. Calculate the total waiting time and total turnaround time for each process 5. Display the process name & burst time for each process.
6. Display the total waiting time, average waiting time, turnaround time

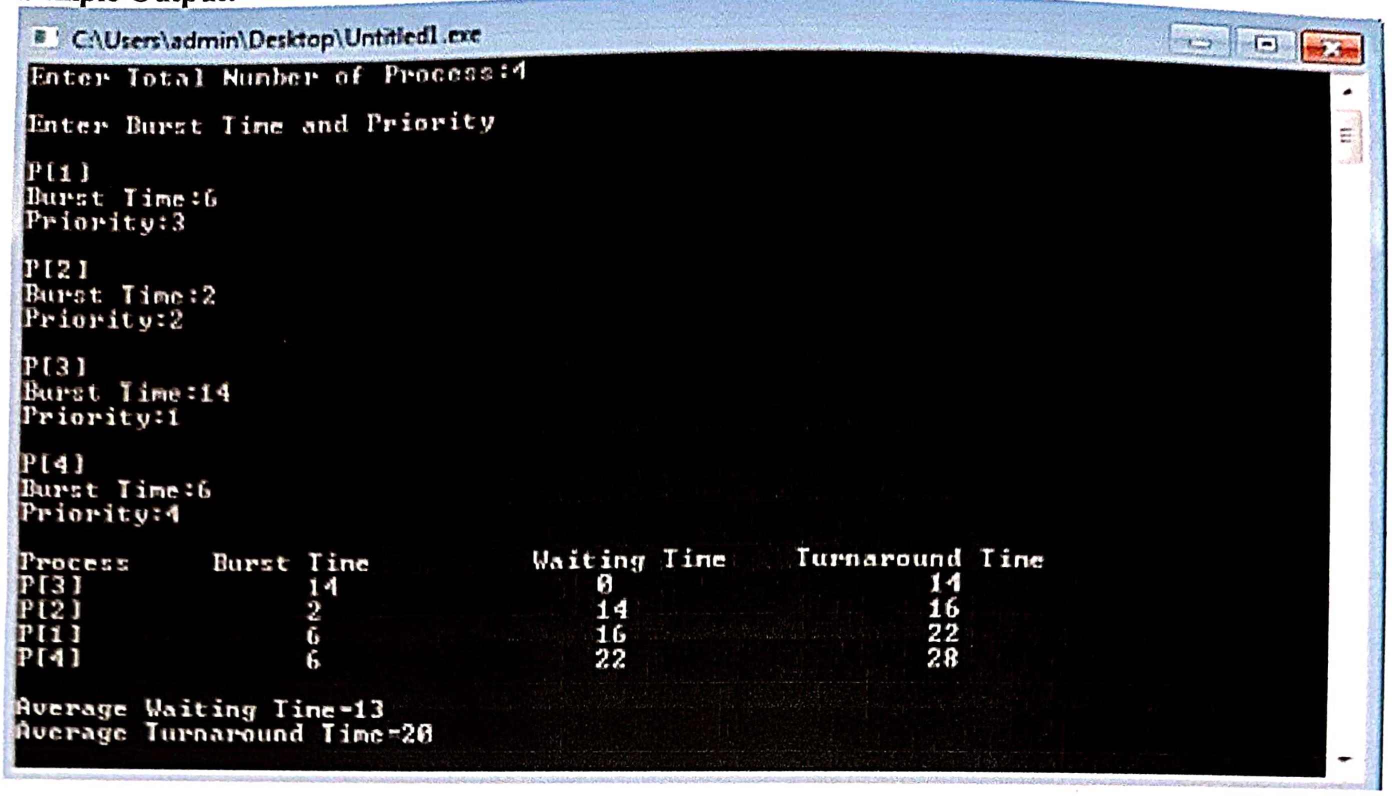
Program Code:

```
#include <stdio.h>
int main(){
    int n;
    printf("Enter total number of process");
    scanf("%d", &n);
    int bt[n], p[n], w[n], t[n];

    printf("\n Enter Burst Time & Priority\n");
    for(int i=0; i<n; i++){
        printf("\n P[%d]\n Burst Time", i+1);
        scanf("%d", &b[i]);
        printf("Priority :");
        scanf("%d", &p[i]);


    for(int i=0; i<n-1; i++){
        for(int j=i+1; j<n; j++){
            if (p[i]>p[j]){
                int temp=bt[i];
                bt[i]=bt[j];
                bt[j]=temp;
            }
        }
    }
w[0]=0
float total-w=0
float total-t=0;
for(int i=1;i<n;i++){
    w[i]=w[i-1]+ b[i-1];
 }
 printf("\n Process \t Burst Time \t waiting Time \t Turn Around Time\n")
 for(int i=0; i<n;i++){
  t[i] =w[i]+b[i];
   total_w +=w[i];
   total-t+= t[i];
    printf("p[%d]\t %d \t\t%d\t\t %d\n", i+1, b[i],w[i],t[i]);
 }
 printf("\n Average waiting time= %d \n", total_w/n));
 printf(" Average turn around time =%d\n", total-t/n);
 return0;
}
```

Output

```
Enter Total Number of Process:4
Enter Burst Time & Priority
P[1]
Burst Time:5
Priority:2
P[2]
Burst Time:8
Priority:1
P[3]
Burst Time:3
Priority:4
P[4]
Burst Time:1
Priority:3
```

Sample Output:



```
Enter Total Number of Process:4

Enter Burst Time and Priority

P[1]
Burst Time:6
Priority:3

P[2]
Burst Time:2
Priority:2

P[3]
Burst Time:14
Priority:1

P[4]
Burst Time:6
Priority:4

Process     Burst Time          Waiting Time     Turnaround Time
P[3]              14                 0                  14
P[2]              2                  14                 16
P[1]              6                  16                 22
P[4]              6                  22                 28

Average Waiting Time=13
Average Turnaround Time=20
```

Process	Burst Time	Waiting time	Turn around time
P[2]	8	0	8
P[1]	5	8	13
P[4]	6	13	19
P[3]	3	19	22

Average waiting time=10

Average turn around time=15

Result:

Thus the code to implement priority scheduling has been executed successfully

[illegible]

Ex. No.: 6d)
Date 21|3|25

ROUND ROBIN SCHEDULING

Aim:

To implement the Round Robin (RR) scheduling technique

Algorithm:

```
1. Declare the structure and its elements.
2. Get number of processes and Time quantum as input from the user.
3. Read the process name, arrival time and burst time
4. Create an array rem_bt[] to keep track of remaining burst time of processes which is initially
   copy of bt[] (burst times array)
5. Create another array wt[] to store waiting times of processes. Initialize this array as 0. 6.
Initialize time : t = 0
7. Keep traversing the all processes while all processes are not done. Do following for i'th
process if it is not done yet.
 a- If rem_bt[i] > quantum
 (i) t = t + quantum
 (ii) bt_rem[i] -= quantum;
 b- Else // Last cycle for this process
 (i) t = t + bt_rem[i];
 (ii) wt[i] = t - bt[i]
 (iii) bt_rem[i] = 0; // This process is over
8. Calculate the waiting time and turnaround time for each process.
9. Calculate the average waiting time and average turnaround time.
10. Display the results.
```

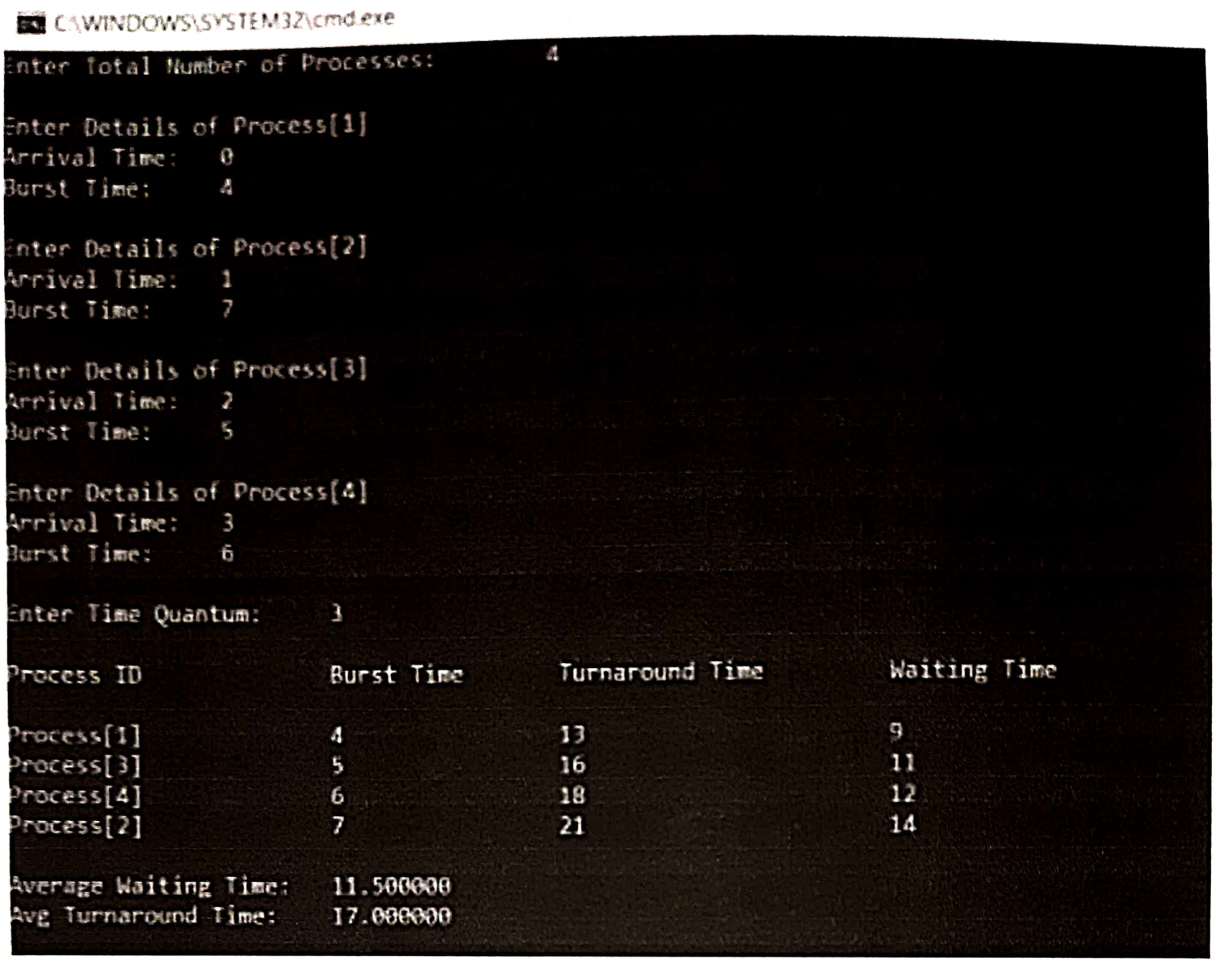
Program Code:

```
#include <Stdio.h>
int main(){
        int n,tq,t=0, done;
        printf("Enter Total number of processes:");
        scanf("%d", &n);
        int bt[n],at[n],rem_bt [n] ,wt[n], tat[n];
        for (int i=0; i<n;i++){
              printf("\n Enter Details of Process[%d]\n", i+1);
              printf("Arrival time :");
              Scanf(" %d", &at[i]);
              printf("Burst Time :");
              scanf(" %d ", &bt[i]);
              rem_bt[i] = bt[i];
        }
   printf("\n Enter Time Quantum");
   Scanf("%d", &tq)
```

```
while(1){
   done=1
   for(int 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]-at[i]
                tat[i]=t-at[i];
                rem_bt[i]=0;
           }
       }
   }
   if(done) break;
}
printf("\n Process ID Burst Time Turnaround Time Waitind Time \n");
float avg_wt=0;
float avg_tat=0;
for(int i=0; i<n; i++){
   avg_wt+=wt[i];
   avg_tat+=tat[i];
   printf("Process[%d]   %d   %d   %d \n", i+1, bt[i], tat[i], wt[i]);
   printf("\n Average waiting Time %f", avg_wt/n);
 printf("\n Average Turnaround time %f", avg_tat/n);
 return 0;
}
```

```
Enter no. of processes : 4
Enter arrival time :0
Enter Burst time :  5 4 6 2
Enter quantum time:2

Process    Burst Time    Arrival time    Waiting Time    Turn Around Time
  1           5              0               10              15
  2           4              0                8              12
  3           6              0               11              17
  4           2              0                6               8

The average waiting time is :8.75
The average turn around time is:13:00
```

Sample Output:



```
C:\WINDOWS\SYSTEM32\cmd.exe
Enter Total Number of Processes:        4

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     4

Enter Details of Process[2]
Arrival Time:   1
Burst Time:     7

Enter Details of Process[3]
Arrival Time:   2
Burst Time:     5

Enter Details of Process[4]
Arrival Time:   3
Burst Time:     6

Enter Time Quantum:     3

Process ID              Burst Time       Turnaround Time         Waiting Time

Process[1]              4                13                      9
Process[3]              5                16                      11
Process[4]              6                18                      12
Process[2]              7                21                      14

Average Waiting Time:   11.500000
Avg Turnaround Time:    17.000000
```

Result:

Thus the code to implement round robin scheduling has been executed successfully

[illegible]