

### 3. Design a CPU scheduling program with C using First Come First Served technique with the following considerations.

- All processes are activated at time 0.
- Assume that no process waits on I/O devices.

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

```
int main()
```

```
    int n, bt[20], wt[20], tat[20];
```

```
    int i, total_wt = 0, total_tat = 0;
```

```
    printf("Enter number of processes: ");
```

```
    scanf("%d", &n);
```

```
    printf("Enter burst time of each process:\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] = wt[i - 1] + bt[i - 1];
```

```
    for (i = 0; i < n; i++)
```

```
        tat[i] = wt[i] + bt[i];
```

```
    for (i = 0; i < n; i++) {
```

```
        total_wt += wt[i];
```

```
        total_tat += tat[i];
```

```
}
```

```
    printf("\nProcess\tBurst\tWaiting\tTurnaround\n");
```

```
    for (i = 0; i < n; i++)
```

```
        printf("P%d\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);
```

```
    printf("\nAverage Waiting Time = %.2f", (float)total_wt / n);
```

```
    printf("\nAverage Turnaround Time = %.2f\n", (float)total_tat / n);
```

```
    return 0;
```

```
}
```

### Output

```
C:\Users\ROJAYADAV\OneDrive> os lab 1.cpp
1  #include <stdio.h>
2  int main()
3  {
4      int n, bt[20], wt[20], tat[20];
5      int i, total_wt = 0, total_tat = 0;
6      printf("Enter number of processes: ");
7      scanf("%d", &n);
8      printf("Enter burst time of each process:\n");
9      for (i = 0; i < n; i++)
10         printf("P%d: ", i + 1);
11         scanf("%d", &bt[i]);
12     }
13     wt[0] = 0;
14     for (i = 1; i < n; i++)
15         wt[i] = wt[i - 1] + bt[i - 1];
16     for (i = 0; i < n; i++)
17         tat[i] = wt[i] + bt[i];
18     for (i = 0; i < n; i++)
19         total_wt += wt[i];
20     total_tat += tat[i];
21
22     printf("\nProcess\tBurst\tWaiting\tTurnaround\n");
23     for (i = 0; i < n; i++)
24         printf("P%d\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);
25
26     printf("\nAverage Waiting Time = %.2f", (float)total_wt / n);
27     printf("\nAverage Turnaround Time = %.2f\n", (float)total_tat / n);
28
29     return 0;
30 }
```

```
Enter number of processes: 3
Enter burst time of each process:
P1: 5
P2: 3
P3: 2

Process    Burst    Waiting    Turnaround
P1        5          0          5
P2        3          5          8
P3        2          8         10

Average Waiting Time = 4.33
Average Turnaround Time = 7.67

-----
Process exited after 18.42 seconds with return value 0
Press any key to continue . . .
```

