

**NAME :** MUHAMMAD AAMIR

**ROLL NO:** DT-22047

# **LAB NO 2:**


## **OPERATING SYSTEM**

1) Implement the First Come First Serve code and paste the output below.

```

1  #include <stdio.h>
2
3  int main() {
4      int bt[20], wt[20], tat[20], i, n;
5      float wtavg, tatavg;
6
7      printf("Enter the number of processes: ");
8      scanf("%d", &n);
9
10     for (i = 0; i < n; i++) {
11         printf("Enter Burst Time for Process %d: ", i);
12         scanf("%d", &bt[i]);
13     }
14
15     wt[0] = wtavg = 0;
16     tat[0] = tatavg = bt[0];
17     for (i = 1; i < n; i++) {
18         wt[i] = wt[i - 1] + bt[i - 1];
19         tat[i] = wt[i] + bt[i];
20         wtavg += wt[i];
21         tatavg += tat[i];
22     }
23
24     printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND TIME\n");
25     for (i = 0; i < n; i++)
26         printf("P%d\t\t%d\t\t%d\t\t%d\n", i, bt[i], wt[i], tat[i]);
27
28     printf("\nAverage Waiting Time: %.2f", wtavg / n);
29     printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);
30
31     return 0;
32 }
33

```



```

Enter the number of processes: 3
Enter Burst Time for Process 0: 2
Enter Burst Time for Process 1: 6
Enter Burst Time for Process 2: 4

PROCESS BURST TIME      WAITING TIME      TURNAROUND TIME
P0          2              0                2
P1          6              2                8
P2          4              8               12

Average Waiting Time: 3.33
Average Turnaround Time: 7.33

```

#include <stdio.h>

```

int main() {
    int bt[20], wt[20], tat[20], i, n;
    float wtavg, tatavg;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Enter Burst Time for Process %d: ", i);
        scanf("%d", &bt[i]);
    }

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

    printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND\nTIME\n");
    for (i = 0; i < n; i++)
        printf("P%d\t\t%d\t\t%d\t\t%d\n", i, bt[i], wt[i], tat[i]);

    printf("\nAverage Waiting Time: %.2f", wtavg / n);
    printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);

    return 0;
}

```

**2) Implement the Shortest Job First code and paste the output below.**

```

1  #include <stdio.h>
2
3  int main() {
4      int bt[20], p[20], wt[20], tat[20], i, k, n, temp;
5      float wtavg, tatavg;
6
7      printf("Enter the number of processes: ");
8      scanf("%d", &n);
9
10     for (i = 0; i < n; i++) {
11         p[i] = i;
12         printf("Enter Burst Time for Process %d: ", i);
13         scanf("%d", &bt[i]);
14     }
15
16     for (i = 0; i < n - 1; i++) {
17         for (k = i + 1; k < n; k++) {
18             if (bt[i] > bt[k]) {
19                 temp = bt[i];
20                 bt[i] = bt[k];
21                 bt[k] = temp;
22                 temp = p[i];
23                 p[i] = p[k];
24                 p[k] = temp;
25             }
26         }
27     }
28
29     wt[0] = wtavg = 0;
30     tat[0] = tatavg = bt[0];
31     for (i = 1; i < n; i++) {
32         wt[i] = wt[i - 1] + bt[i - 1];
33         tat[i] = wt[i] + bt[i];
34         wtavg += wt[i];
35         tatavg += tat[i];
36     }
37
38     printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND TIME\n");
39     for (i = 0; i < n; i++)
40         printf("P%d\t\t%d\t\t%d\t\t%d\n", p[i], bt[i], wt[i], tat[i]);
41
42     printf("\nAverage Waiting Time: %.2f", wtavg / n);
43     printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);
44
45     return 0;
46 }
47

```

```

Enter the number of processes: 3
Enter Burst Time for Process 0: 2
Enter Burst Time for Process 1: 6
Enter Burst Time for Process 2: 4

PROCESS BURST TIME      WAITING TIME      TURNAROUND TIME
P0           2           0           2
P2           4           2           6
P1           6           6          12

Average Waiting Time: 2.67
Average Turnaround Time: 6.67

...Program finished with exit code 0
Press ENTER to exit console.

```

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

```

int main() {
    int bt[20], p[20], wt[20], tat[20], i, k, n, temp;
    float wtavg, tatavg;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        p[i] = i;
        printf("Enter Burst Time for Process %d: ", i);
        scanf("%d", &bt[i]);
    }

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

    // Calculation of waiting and turnaround times
    wt[0] = 0;
    for (i = 1; i < n; i++) {
        wt[i] = wt[i-1] + bt[i-1];
    }

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

    // Average values
    wtavg = (wt[0] + wt[1] + ... + wt[n-1]) / n;
    tatavg = (tat[0] + tat[1] + ... + tat[n-1]) / n;

    printf("Average Waiting Time: %.2f\n", wtavg);
    printf("Average Turnaround Time: %.2f\n", tatavg);

    return 0;
}

```

```

    }
}

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

printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND
TIME\n");
for (i = 0; i < n; i++)
    printf("P%d\t\t%d\t\t%d\t\t%d\n", p[i], bt[i], wt[i], tat[i]);

printf("\nAverage Waiting Time: %.2f", wtavg / n);
printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);

return 0;
}

```

**3) Implement the Round Robin code and paste the output below.**

```

1  #include <stdio.h>
2
3  int main() {
4      int i, j, n, bu[10], wa[10], tat[10], t, ct[10], max;
5      float awt = 0, att = 0, temp = 0;
6
7      printf("Enter the number of processes: ");
8      scanf("%d", &n);
9
10     for (i = 0; i < n; i++) {
11         printf("Enter Burst Time for Process %d: ", i + 1);
12         scanf("%d", &bu[i]);
13         ct[i] = bu[i];
14     }
15
16     printf("Enter Time Quantum: ");
17     scanf("%d", &t);
18
19     max = bu[0];
20     for (i = 1; i < n; i++)
21         if (max < bu[i])
22             max = bu[i];
23
24     for (j = 0; j < (max / t) + 1; j++) {
25         for (i = 0; i < n; i++) {
26             if (bu[i] != 0) {
27                 if (bu[i] <= t) {
28                     tat[i] = temp + bu[i];
29                     temp += bu[i];
30                     bu[i] = 0;
31                 } else {
32                     bu[i] -= t;
33                     temp += t;
34                 }
35             }
36         }
37     }
38
39     for (i = 0; i < n; i++) {
40         wa[i] = tat[i] - ct[i];
41         att += tat[i];
42         awt += wa[i];
43     }
44
45     printf("\nAverage Turnaround Time: %.2f", att / n);
46     printf("\nAverage Waiting Time: %.2f\n", awt / n);
47     printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND TIME\n");
48     for (i = 0; i < n; i++)
49         printf("P%d\t\t%d\t\t%d\t\t%d\n", i + 1, ct[i], wa[i], tat[i]);
50
51     return 0;
52 }
53

```

```

Enter the number of processes: 3
Enter Burst Time for Process 1: 2
Enter Burst Time for Process 2: 6
Enter Burst Time for Process 3: 4
Enter Time Quantum: 3

Average Turnaround Time: 8.33
Average Waiting Time: 4.33

PROCESS BURST TIME      WAITING TIME      TURNAROUND TIME
P1          2           0           2
P2          6           5          11
P3          4           8          12

...Program finished with exit code 0
Press ENTER to exit console.

```

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

```

int main() {
    int i, j, n, bu[10], wa[10], tat[10], t, ct[10], max;
    float awt = 0, att = 0, temp = 0;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Enter Burst Time for Process %d: ", i + 1);
        scanf("%d", &bu[i]);
        ct[i] = bu[i];
    }

    printf("Enter Time Quantum: ");
    scanf("%d", &t);

    max = bu[0]; for (i =
1; i < n; i++) if (max
< bu[i]) max = bu[i];

```



```

for (j = 0; j < (max / t) + 1; j++) {
    for (i = 0; i < n; i++) {
        if (bu[i] != 0) {
            if (bu[i] <= t) {
                tat[i] = temp + bu[i];
                temp += bu[i];
                bu[i] = 0;
            } else {
                bu[i] -= t;
                temp += t;
            }
        }
    }
}

for (i = 0; i < n; i++) {
    wa[i] = tat[i] - ct[i];
    att += tat[i];
    awt += wa[i];
}

printf("\nAverage Turnaround Time: %.2f", att / n);
printf("\nAverage Waiting Time: %.2f\n", awt / n);
printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND\nTIME\n");
for (i = 0; i < n; i++)
    printf("P%d\t\t%d\t\t%d\t\t%d\n", i + 1, ct[i], wa[i], tat[i]);

return 0;
}

```

**4) Implement the Priority Based Scheduling code and paste the output below.**

```

1  #include <stdio.h>
2
3  int main() {
4      int p[20], bt[20], pri[20], wt[20], tat[20], i, k, n, temp;
5      float wtavg, tatavg;
6
7      printf("Enter the number of processes: ");
8      scanf("%d", &n);
9
10     for (i = 0; i < n; i++) {
11         p[i] = i;
12         printf("Enter Burst Time and Priority for Process %d: ", i);
13         scanf("%d%d", &bt[i], &pri[i]);
14     }
15
16     for (i = 0; i < n - 1; i++) {
17         for (k = i + 1; k < n; k++) {
18             if (pri[i] > pri[k]) {
19                 temp = pri[i];
20                 pri[i] = pri[k];
21                 pri[k] = temp;
22
23                 temp = bt[i];
24                 bt[i] = bt[k];
25                 bt[k] = temp;
26
27                 temp = p[i];
28                 p[i] = p[k];
29                 p[k] = temp;
30             }
31         }
32     }
33
34     wtavg = wt[0] = 0;
35     tatavg = tat[0] = bt[0];
36     for (i = 1; i < n; i++) {
37         wt[i] = wt[i - 1] + bt[i - 1];
38         tat[i] = wt[i] + bt[i];
39         wtavg += wt[i];
40         tatavg += tat[i];
41     }
42
43     printf("\nPROCESS\tPRIORITY\tBURST TIME\tWAITING TIME\tTURNAROUND TIME\n");
44     for (i = 0; i < n; i++)
45         printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i], pri[i], bt[i], wt[i], tat[i]);
46
47     printf("\nAverage Waiting Time: %.2f", wtavg / n);
48     printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);
49
50     return 0;
51 }
52

```

```
Enter the number of processes: 3
Enter Burst Time and Priority for Process 0: 2
3
Enter Burst Time and Priority for Process 1: 6
1
Enter Burst Time and Priority for Process 2: 4
2

PROCESS PRIORITY      BURST TIME      WAITING TIME      TURNAROUND TIME
P1          1          6          0          6
P2          2          4          6          10
P0          3          2          10          12

Average Waiting Time: 5.33
Average Turnaround Time: 9.33

...Program finished with exit code 0
Press ENTER to exit console.
```

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

```
int main() {
    int p[20], bt[20], pri[20], wt[20], tat[20], i, k, n, temp;
    float wtavg, tatavg;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        p[i] = i;
        printf("Enter Burst Time and Priority for Process %d: ", i);
        scanf("%d%d", &bt[i], &pri[i]);
    }

    for (i = 0; i < n - 1; i++) {
        for (k = i + 1; k < n; k++) {
            if (pri[i] > pri[k]) {
                temp = pri[i];
                pri[i] = pri[k];
                pri[k] = temp;

                temp = bt[i];
                bt[i] = bt[k];
            }
        }
    }

    // ... (rest of the code for calculating waiting and turnaround times)
}
```

```

        bt[k] = temp;

        temp = p[i];
        p[i] = p[k];
        p[k] = temp;
    }
}

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

printf("\nPROCESS\tPRIORITY\tBURST TIME\tWAITING
TIME\tTURNAROUND TIME\n");
for (i = 0; i < n; i++)
    printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i], pri[i], bt[i], wt[i], tat[i]);

printf("\nAverage Waiting Time: %.2f", wtavg / n);
printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);

return 0;
}

```

Execute all scheduling algorithms on following data and find out the Average Waiting Time and Average Turnaround Time of all scheduling algorithms and discuss your results.  
(Quantum Value is 3)

## 1. First Come First Serve (FCFS)

Order of execution: P0 -> P1 -> P2

Process	Burst Time	Waiting Time	Turnaround Time
P0	2	0	2
P1	6	2	8
P2	4	8	12

$$AWT = (0 + 2 + 8) / 3 = 3.33$$

$$TAT = (2 + 8 + 12) / 3 = 7.33$$

## 2. Shortest Job First (SJF)

Order of execution: P0 -> P2 -> P1

Process	Burst Time	Waiting Time	Turnaround Time
P0	2	0	2
P2	4	2	6
P1	6	6	12

$$AWT = (0 + 2 + 6) / 3 = 2.67$$

$$TAT = (2 + 6 + 12) / 3 = 6.67$$

## 3. Round Robin (Quantum = 3)

Execution order: P0 -> P1 -> P2 -> P1

Process	Burst Time	Waiting Time	Turnaround Time
P0	2	0	2
P1	6	7	13
P2	4	3	7

$$AWT = (0 + 7 + 3) / 3 = 3.33$$

$$TAT = (2 + 13 + 7) / 3 = 7.33$$

#### 4. Priority Scheduling

Order of execution (based on priority) P1 -> P2 -> P0

Process	Burst Time	Priority	Waiting Time	Turnaround Time
P1	6	1	0	6
P2	4	2	6	10
P0	2	3	10	12

$$AWT = (0 + 6 + 10) / 3 = 5.33$$

$$TAT = (6 + 10 + 12) / 3 = 9.33$$

Algorithm	Average Waiting Time	Average Turnaround Time
-----------	----------------------	-------------------------

FCFS	3.33	7.33
------	------	------

SJF	2.67	6.67
-----	------	------

Round Robin	3.33	7.33
-------------	------	------

Priority	5.33	9.33
----------	------	------