

Exp No:6a

First Come First Serve

PROGRAM

```
#include <stdio.h>

void findWaitingTime(int processes[], int n, int bt[], int wt[]) {
    wt[0] = 0;
    for (int i = 1; i < n; i++)
        wt[i] = bt[i - 1] + wt[i - 1];
}

void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];
}

void findavgTime(int processes[], int n, int bt[]) {
    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    findWaitingTime(processes, n, bt, wt);
    findTurnAroundTime(processes, n, bt, wt, tat);

    printf("\nProcesses   Burst time   Waiting time   Turn around time\n");
    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf("    %d         %d           %d           %d\n", processes[i], bt[i], wt[i], tat[i]);
    }

    float avg_wt = (float)total_wt / (float)n;
    float avg_tat = (float)total_tat / (float)n;
    printf("\nAverage waiting time = %.2f", avg_wt);
    printf("\nAverage turn around time = %.2f\n", avg_tat);
}
```

```
int main() {
    int n;

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

    int processes[n], bt[n];

    printf("Enter the burst time for each process:\n");
    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;
        printf("Process %d: ", i + 1);
        scanf("%d", &bt[i]);
    }

    findavgTime(processes, n, bt);
    return 0;
}
```

OUTPUT

```

[cse81@localhost ~]$ ./a.out
Enter the number of processes: 3
Enter the burst time for each process:
Process 1: 24
Process 2: 3
Process 3: 3

Processes    Burst time    Waiting time    Turn around time
  1           24             0             24
  2            3            24             27
  3            3            27             30

Average waiting time = 17.00
Average turn around time = 27.00

```

Exp No:6b Shortest Job First

PROGRAM

```

#include <stdio.h>

void main() {
    int n, i, j, temp;
    float avg_wt = 0, avg_tat = 0;

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

    int bt[n], wt[n], tat[n], p[n];

    printf("Enter the burst time of the processes: \n");
    for (i = 0; i < n; i++) {
        scanf("%d", &bt[i]);
        p[i] = i + 1;
    }

    // Sorting based on burst time (SJF Scheduling)
    for (i = 0; i < n - 1; i++) {
        for (j = i + 1; j < n; j++) {
            if (bt[i] > bt[j]) {
                // Swap burst time
                temp = bt[i];
                bt[i] = bt[j];
                bt[j] = temp;

                // Swap process number
                temp = p[i];
                p[i] = p[j];
                p[j] = temp;
            }
        }
    }

}

```

```

wt[0] = 0; // First process has zero waiting time

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

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

avg_wt /= n;
avg_tat /= n;

printf("\nProcess   Burst Time   Waiting Time   Turnaround Time\n");
for (i = 0; i < n; i++) {
    printf("    %d\t      %d\t          %d\t          %d\n", p[i], bt[i], wt[i], tat[i]);
}

printf("\nAverage Waiting Time: %.2f", avg_wt);
printf("\nAverage Turnaround Time: %.2f\n", avg_tat);
}

```

OUTPUT

```

[cse81@localhost ~]$ ./a.out
Enter the number of processes: 3
Enter the burst time of the processes:
3
7
2

Process   Burst Time   Waiting Time   Turnaround Time
    3           2           0           2
    1           3           2           5
    2           7           5          12

Average Waiting Time: 2.33
Average Turnaround Time: 6.33

```

Exp No:6c

Priority

PROGRAM

```

#include <stdio.h>

struct Process {
    int id;
    int bt; // Burst Time
    int priority;
    int wt; // Waiting Time
    int tat; // Turnaround Time
};

void swap(struct Process *a, struct Process *b) {
    struct Process temp = *a;
    *a = *b;
    *b = temp;
}

// Function to sort processes based on priority (Higher priority first)
void sortProcesses(struct Process proc[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (proc[i].priority > proc[j].priority) {
                swap(&proc[i], &proc[j]);
            }
        }
    }
}

// Function to calculate waiting time and turnaround time
void calculateTimes(struct Process proc[], int n) {
    proc[0].wt = 0; // First process has zero waiting time

    for (int i = 1; i < n; i++) {
        proc[i].wt = proc[i - 1].wt + proc[i - 1].bt;
    }

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

```

```

void displayResults(struct Process proc[], int n) {
    int total_wt = 0, total_tat = 0;

    printf("\nProcess\tBurst Time\tPriority\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("P%d\t%d\t%d\t\t%d\t\t%d\n", proc[i].id, proc[i].bt, proc[i].priority, proc[i].wt, proc[i].tat);
        total_wt += proc[i].wt;
        total_tat += proc[i].tat;
    }

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

int main() {
    int n;

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

    struct Process proc[n];

    for (int i = 0; i < n; i++) {
        proc[i].id = i + 1;
        printf("\nP[%d]\n", i + 1);
        printf("Burst Time: ");
        scanf("%d", &proc[i].bt);
        printf("Priority: ");
        scanf("%d", &proc[i].priority);
    }

    sortProcesses(proc, n);
    calculateTimes(proc, n);
    displayResults(proc, n);

    return 0;
}

```

OUTPUT

```

[cse81@localhost ~]$ ./a.out
Enter the number of processes: 4

P[1]
Burst Time: 6
Priority: 3

P[2]
Burst Time: 2
Priority: 2

P[3]
Burst Time: 4
Priority: 1

P[4]
Burst Time: 6
Priority: 4

Process Burst Time      Priority      Waiting Time      Turnaround Time
P3      4                  1              0                  4
P2       2                  2              4                  6
P1       6                  3              6                 12
P4       6                  4             12                 18

Average Waiting Time = 5.50
Average Turnaround Time = 10.00

```

Exp No:6d

Round Robin

PROGRAM

```
#include <stdio.h>

struct Process {
    int id;
    int at; // Arrival Time
    int bt; // Burst Time
    int wt; // Waiting Time
    int tat; // Turnaround Time
};

// Function to implement Round Robin Scheduling
void roundRobinScheduling(struct Process proc[], int n, int quantum) {
    int rem_bt[n]; // Array to store remaining burst times
    int t = 0; // Current time
    int done;

    // Initialize remaining burst times
    for (int i = 0; i < n; i++) {
        rem_bt[i] = proc[i].bt;
        proc[i].wt = 0; // Initialize waiting time to zero
    }

    // Keep executing processes in a cyclic manner
    do {
        done = 1;
        for (int i = 0; i < n; i++) {
            if (rem_bt[i] > 0) {
                done = 0; // There is a pending process
                if (rem_bt[i] > quantum) {
                    t += quantum;
                    rem_bt[i] -= quantum;
                } else { // Last cycle for this process
                    t += rem_bt[i];
                    proc[i].wt = t - proc[i].bt - proc[i].at;
                    rem_bt[i] = 0;
                }
            }
        }
    } while (!done);
}
```

```

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

void displayResults(struct Process proc[], int n) {
    int total_wt = 0, total_tat = 0;

    printf("\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("P%d\t%d\t%d\t%d\t%d\n", proc[i].id, proc[i].at, proc[i].bt, proc[i].wt, proc[i].tat);
        total_wt += proc[i].wt;
        total_tat += proc[i].tat;
    }

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

int main() {
    int n, quantum;

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

    struct Process proc[n];

    for (int i = 0; i < n; i++) {
        proc[i].id = i + 1;
        printf("\nP[%d]\n", i + 1);
        printf("Arrival Time: ");
        scanf("%d", &proc[i].at);
        printf("Burst Time: ");
        scanf("%d", &proc[i].bt);
    }

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

    roundRobinScheduling(proc, n, quantum);
    displayResults(proc, n);
}

```

OUTPUT

```
[cse81@localhost ~]$ ./a.out
Enter the number of processes: 4
```

```
P[1]
Arrival Time: 0
Burst Time: 3
```

```
P[2]
Arrival Time: 1
Burst Time: 7
```

```
P[3]
Arrival Time: 2
Burst Time: 5
```

```
P[4]
Arrival Time: 3
Burst Time: 6
Enter Time Quantum: 3
```

Process	Arrival Time	Burst Time	Waiting Time	Turnaround Time
P1	0	3	0	3
P2	1	7	13	20
P3	2	5	10	15
P4	3	6	11	17

```
Average Waiting Time = 8.50
Average Turnaround Time = 13.75
```