PRACTICAL:12

AIM:- FCFS (First Come First Serve) Scheduling Algorithm

```
#include <stdio.h>

void findWaitingTime(int processes[], int n, int bt[], int wt[]) {
wt[0] = 0; // First process has no waiting time

   for (int i = 1; i < n; i++) {        wt[i] = bt[i - 1] + wt[i
- 1]; // Waiting time formula
   }
}

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]; // Turnaround time
formula
   }
}

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("Processes  Burst Time  Waiting Time  Turnaround 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]);
   }
```

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

int main() {
    int processes[] = {1, 2, 3};     int n = sizeof
processes / sizeof processes[0];     int
burst_time[] = {10, 5, 8};

    findAvgTime(processes, n, burst_time);
return 0;
}
```

Output Example:

```
Processes  Burst Time  Waiting Time  Turnaround Time
1          10          0             10
2          5           10            15
3          8           15            23

Average Waiting Time = 8.33
Average Turnaround Time = 16.00
```

AIM : Round Robin (RR) Scheduling Algorithm

CODE :

```
#include <stdio.h>

void findWaitingTime(int processes[], int n, int bt[], int wt[], int quantum) {
int rem_bt[n];

   for (int i = 0; i < n; i++) {
rem_bt[i] = bt[i];
   }

   int t = 0; // Current time

   while (1) {        int done =
1;        for (int i = 0; i < n;
i++) {            if (rem_bt[i] >
0) {
            done = 0; // There is a process still running

            if (rem_bt[i] > quantum) {
t += quantum;                    rem_bt[i]
-= quantum;
            } else {
t += rem_bt[i];
wt[i] = t - bt[i];
rem_bt[i] = 0;
            }
         }
```

```
        }       if
(done == 1)
break;
    }
}

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 quantum) {
int wt[n], tat[n], total_wt = 0, total_tat = 0;

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

    printf("Processes  Burst Time  Waiting Time  Turnaround 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]);
    }

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

int main() {
    int processes[] = {1, 2, 3};     int n = sizeof
processes / sizeof processes[0];     int
burst_time[] = {10, 5, 8};     int quantum = 2;
```

```
    findAvgTime(processes, n, burst_time, quantum);
return 0;
}
```

OUTPUT :

```
Processes  Burst Time  Waiting Time  Turnaround Time
1          10          10            20
2          5           12            17
3          8           13            21

Average Waiting Time = 11.67
Average Turnaround Time = 19.33
```