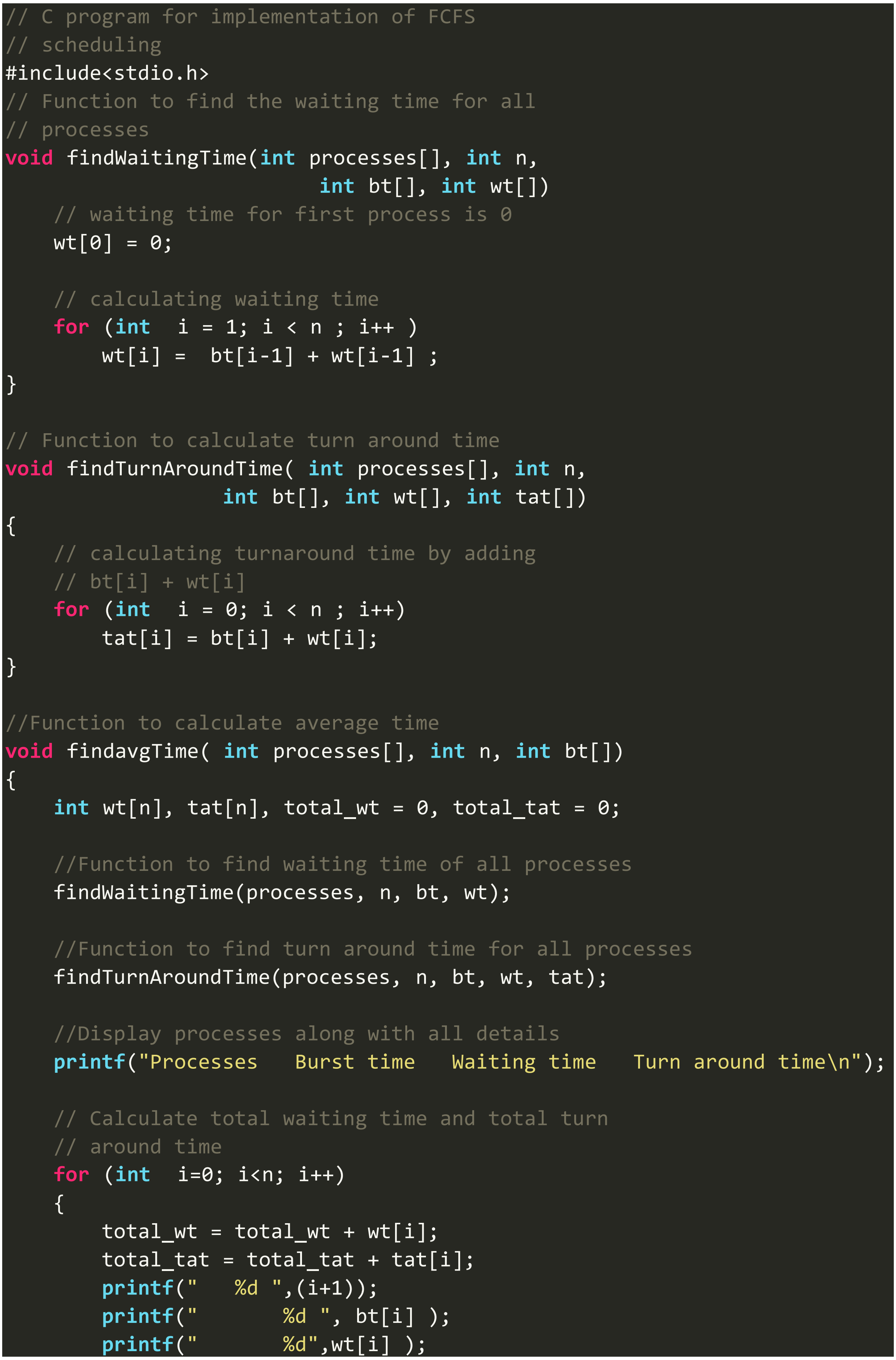

```c
// C program for implementation of FCFS
// scheduling
#include<stdio.h>
// Function to find the waiting time for all
// processes
void findWaitingTime(int processes[], int n,
                          int bt[], int wt[])
    // waiting time for first process is 0
    wt[0] = 0;

    // calculating waiting time
    for (int  i = 1; i < n ; i++ )
        wt[i] =  bt[i-1] + wt[i-1] ;
}

// Function to calculate turn around time
void findTurnAroundTime( int processes[], int n,
                  int bt[], int wt[], int tat[])
{
    // calculating turnaround time by adding
    // bt[i] + wt[i]
    for (int  i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}

//Function to calculate average time
void findavgTime( int processes[], int n, int bt[])
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    //Function to find waiting time of all processes
    findWaitingTime(processes, n, bt, wt);

    //Function to find turn around time for all processes
    findTurnAroundTime(processes, n, bt, wt, tat);

    //Display processes along with all details
    printf("Processes   Burst time   Waiting time   Turn around time\n");

    // Calculate total waiting time and total turn
    // around time
    for (int  i=0; i<n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        printf("   %d ",(i+1));
        printf("       %d ", bt[i] );
        printf("       %d",wt[i] );
        printf("       %d\n",tat[i] );
    }
    float s=(float)total_wt / (float)n;
    float t=(float)total_tat / (float)n;
    printf("Average waiting time = %f",s);
    printf("\n");
    printf("Average turn around time = %f ",t);
}

// Driver code
int main()
{
    //process id's
    int processes[] = { 1, 2, 3};
    int n = sizeof processes / sizeof processes[0];

    //Burst time of all processes
    int  burst_time[] = {10, 5, 8};

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

\\ fcfs with arrival time

```
     #include<stdio.h>

int main()
{
    int  p[10],at[10],bt[10],ct[10],tat[10],wt[10],i,j,temp=0,n;
    float awt=0,atat=0;
    printf("enter no of proccess you want:");
    scanf("%d",&n);
    printf("enter %d process:",n);
    for(i=0;i<n;i++)
    {
    scanf("%d",&p[i]);
    }
    printf("enter %d arrival time:",n);
    for(i=0;i<n;i++)
    {
    scanf("%d",&at[i]);
    }
    printf("enter %d burst time:",n);
    for(i=0;i<n;i++)
    {
```

```
    scanf("%d",&bt[i]);
    }
    // sorting at,bt, and process according to at
    for(i=0;i<n;i++)
    {
     for(j=0;j<(n-i);j++)
    {
      if(at[j]>at[j+1])
     {
        temp=p[j+1];
        p[j+1]=p[j];
        p[j]=temp;
        temp=at[j+1];
        at[j+1]=at[j];
        at[j]=temp;
        temp=bt[j+1];
        bt[j+1]=bt[j];
        bt[j]=temp;
      }
     }
    }
    /* calculating 1st ct */
    ct[0]=at[0]+bt[0];
    /* calculating 2 to n ct */
    for(i=1;i<n;i++)
    {
      //when proess is ideal in between i and i+1
      temp=0;
     if(ct[i-1]<at[i])
     {
        temp=at[i]-ct[i-1];
     }
     ct[i]=ct[i-1]+bt[i]+temp;
    }
    /* calculating tat and wt */
    printf("\np\t A.T\t B.T\t C.T\t TAT\t WT");
    for(i=0;i<n;i++)
    {
    tat[i]=ct[i]-at[i];
    wt[i]=tat[i]-bt[i];
    atat+=tat[i];
    awt+=wt[i];
    }
    atat=atat/n;
    awt=awt/n;
    for(i=0;i<n;i++)
    {
      printf("\nP%d\t %d\t %d\t %d \t %d \t
%d",p[i],at[i],bt[i],ct[i],tat[i],wt[i]);
```

```
    }
    printf("\naverage turnaround time is %f",atat);

    printf("\naverage wating timme is %f",awt);
    return 0;
}
```

```
\\ sjf without arrival time
```

```
    #include <stdio.h>

    int main()

    {

        // Matrix for storing Process Id, Burst

        // Time, Average Waiting Time & Average

        // Turn Around Time.

        int A[100][4];

        int i, j, n, total = 0, index, temp;

        float avg_wt, avg_tat;

        printf("Enter number of process: ");

        scanf("%d", &n);

        printf("Enter Burst Time:\n");

        // User Input Burst Time and alloting Process Id.

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

            printf("P%d: ", i + 1);

            scanf("%d", &A[i][1]);

            A[i][0] = i + 1;

        }

        // Sorting process according to their Burst Time.

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

            index = i;

            for (j = i + 1; j < n; j++)
```

```
                if (A[j][1] < A[index][1])
                        index = j;
        temp = A[i][1];
        A[i][1] = A[index][1];
        A[index][1] = temp;

        temp = A[i][0];
        A[i][0] = A[index][0];
        A[index][0] = temp;
}
A[0][2] = 0;
// Calculation of Waiting Times
for (i = 1; i < n; i++) {
        A[i][2] = 0;
        for (j = 0; j < i; j++)
                A[i][2] += A[j][1];
        total += A[i][2];
}
avg_wt = (float)total / n;
total = 0;
printf("P       BT      WT      TAT\n");
// Calculation of Turn Around Time and printing the
// data.
for (i = 0; i < n; i++) {
        A[i][3] = A[i][1] + A[i][2];
        total += A[i][3];
        printf("P%d     %d      %d      %d\n", A[i][0],
                A[i][1], A[i][2], A[i][3]);
}
avg_tat = (float)total / n;
printf("Average Waiting Time= %f", avg_wt);
```

```
            printf("\nAverage Turnaround Time= %f", avg_tat);
    }
```

```
    struct Process {
    int arrival_time;
    int burst_time;
int waiting_time;
    };

int compare(const void *a, const void *b) {
    struct Process *p1 = (struct Process *)a;
struct Process *p2 = (struct Process *)b;
    return p1->burst_time - p2->burst_time;
    }

    int main() {
    int n, i, j;
float avg_waiting_time = 0, avg_turnaround_time = 0;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    struct Process processes[n];
    for (i = 0; i< n; i++) {
    printf("Enter arrival time and burst time of process %d: ", i+1);
    scanf("%d %d", &processes[i].arrival_time, &processes[i].burst_time);
    }
    qsort(processes, n, sizeof(struct Process), compare);
    processes[0].waiting_time = 0;
    for (i = 1; i< n; i++) {
    processes[i].waiting_time = 0;
    for (j = 0; j <i; j++)
    {

    processes[i].waiting_time += processes[j].burst_time;
```

```
        }

        avg_waiting_time += processes[i].waiting_time;

        }

        avg_waiting_time /= n;

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

        avg_turnaround_time += processes[i].burst_time + processes[i].waiting_time;

        }

avg_turnaround_time /= n;

        printf("\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\"
        );

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

        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i+1, processes[i].arrival_time, processe
        s[i].burst_time, processes[i].waiting_time, processes[i].burst_time+processes[i].
        waiting_time);

        }

        printf("\nAverage Waiting Time: %f\n", avg_waiting_time);

        printf("Average Turnaround Time: %f\n", avg_turnaround_time);

        return 0;

        }
```