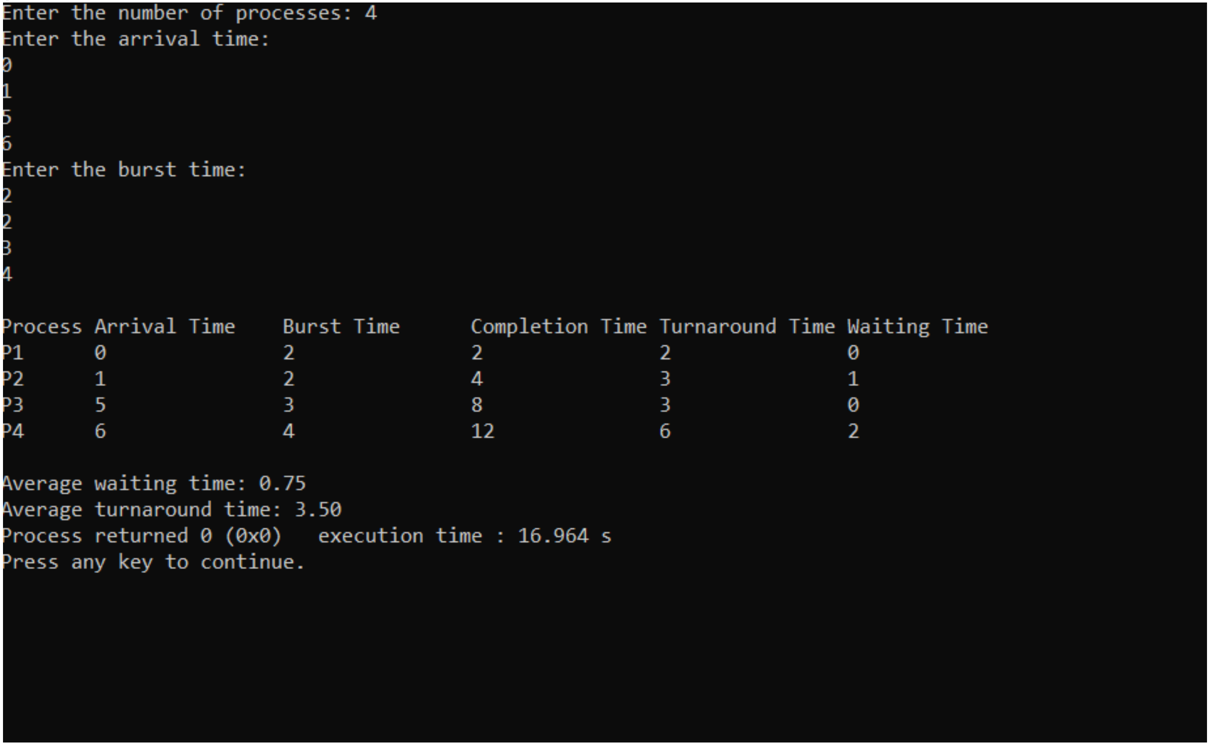
1.FCFS scheduling using array.

```
#include <stdio.h>

#define MAX_PROCESS 10

void fcfs(int n, int at[], int bt[]) {
    int ct[MAX_PROCESS];
    int tat[MAX_PROCESS];
    int wt[MAX_PROCESS];

    int total_wt = 0;
    int total_tat = 0;

    int current_time = 0;

    // Initialize completion time array with -1
    for (int i = 0; i < n; i++) {
        ct[i] = -1;
    }

    // Find completion time for each process
    for (int i = 0; i < n; i++) {
        if (current_time < at[i]) {
            current_time = at[i];
        }

        ct[i] = current_time + bt[i];
        current_time = ct[i];
    }

    // Find turnaround time for each process
    for (int i = 0; i < n; i++) {
        tat[i] = ct[i] - at[i];
        total_tat += tat[i];
    }

    // Find waiting time for each process
    for (int i = 0; i < n; i++) {
        wt[i] = tat[i] - bt[i];
        total_wt += wt[i];
    }

    // Print the results
    printf("\nProcess\tArrival Time\tBurst Time\tCompletion Time\tTurnaround 
Time\tWaiting Time\n");
    for (int i = 0; i < n; i++) {
        printf("P%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i+1, at[i], bt[i], ct[i], tat[i], wt[i]);
    }
    printf("\nAverage waiting time: %.2f", (float)total_wt / n);
    printf("\nAverage turnaround time: %.2f", (float)total_tat / n);
}

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

    int at[n], bt[n];

    printf("Enter the arrival time:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &at[i]);
    }

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

    fcfs(n, at, bt);

    return 0;
}
```

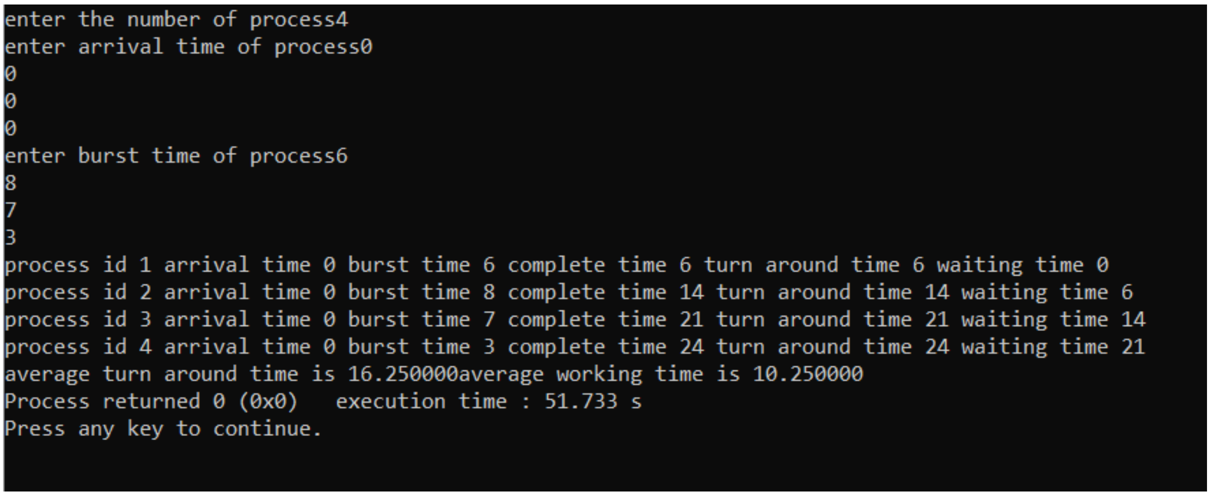


```
Enter the number of processes: 4
Enter the arrival time:
0
1
5
6
Enter the burst time:
2
2
3
4

Process Arrival Time    Burst Time      Completion Time Turnaround Time Waiting Time
P1      0               2               2               2               0
P2      1               2               4               3               1
P3      5               3               8               3               0
P4      6               4               12              6               2

Average waiting time: 0.75
Average turnaround time: 3.50
Process returned 0 (0x0)   execution time : 16.964 s
Press any key to continue.
```

2.SJF(non-preemptive)scheduling using array

```
#include<stdio.h>

int main(){
```

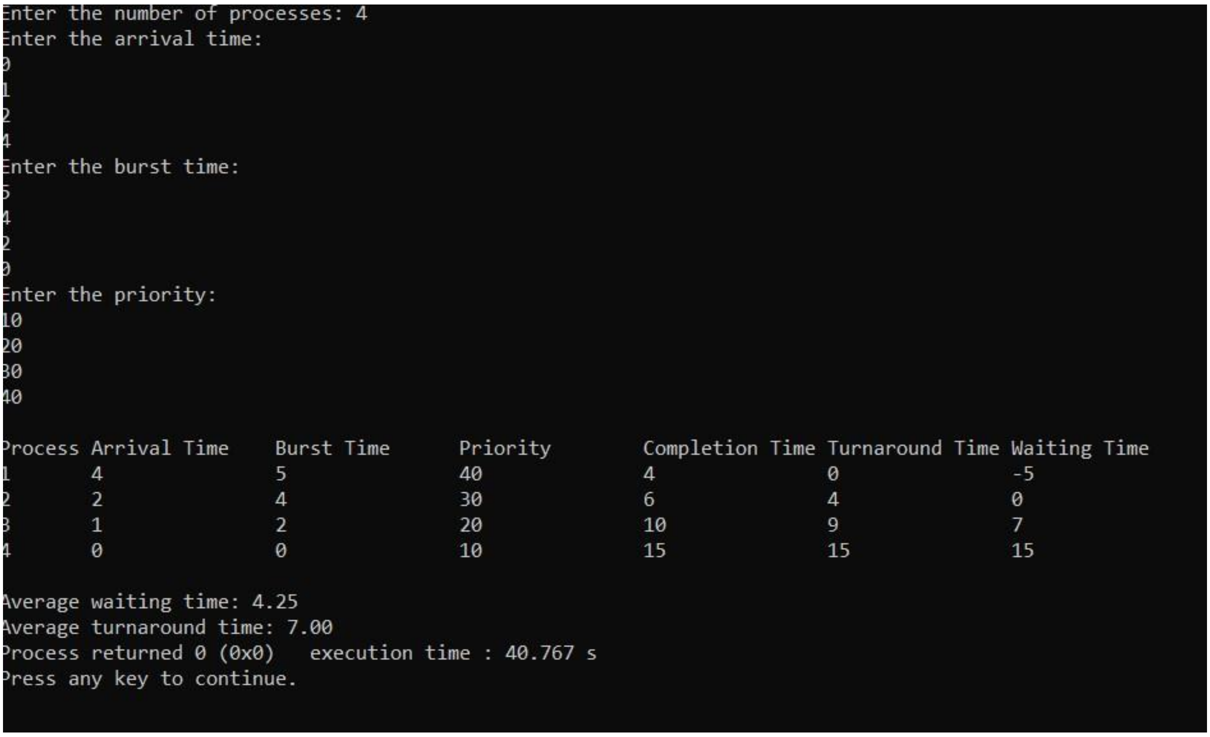
```
int n,i;
float atat=0,awt=0;
printf("enter the number of process");
scanf("%d",&n);
int atime1[n],btime2[n],ctime3[n],tattime4[n],wtime5[n];
printf("enter arrival time of process");
for(i=0;i<n;i++){
   scanf("%d",&atime1[i]);
}
printf("enter burst time of process");
for(i=0;i<n;i++){
   scanf("%d",&btime2[i]);
}
for(i=0;i<n;i++){
   if(i==0){
      ctime3[i]=atime1[i]+btime2[i];
   }
   else{
      if(ctime3[i-1]<atime1[i]){
         ctime3[i]=(atime1[i]-ctime3[i-1])+ctime3[i-1]+btime2[i];
      }
      else{
         ctime3[i]=ctime3[i-1]+btime2[i];
      }
   }
}
for(i=0;i<n;i++){
   tattime4[i]=ctime3[i]-atime1[i];
}
for(i=0;i<n;i++){
        wtime5[i]=tattime4[i]-btime2[i];
    }
    for(i=0;i<n;i++){
        atat=atat+tattime4[i];
    }
    atat=(atat/n);
    for(i=0;i<n;i++){
        awt=awt+wtime5[i];
    }
    awt=(awt/n);
    for(i=0;i<n;i++){
        printf("process id %d arrival time %d burst time %d complete time %d turn around time %d waiting time %d\n",i+1,atime1[i],btime2[i],ctime3[i],tattime4[i],wtime5[i]);
    }
    printf("average turn around time is %f",atat);
    printf("average working time is %f",awt);
}
```



```
enter the number of process4
enter arrival time of process0
0
0
0
enter burst time of process6
8
7
3
process id 1 arrival time 0 burst time 6 complete time 6 turn around time 6 waiting time 0
process id 2 arrival time 0 burst time 8 complete time 14 turn around time 14 waiting time 6
process id 3 arrival time 0 burst time 7 complete time 21 turn around time 21 waiting time 14
process id 4 arrival time 0 burst time 3 complete time 24 turn around time 24 waiting time 21
average turn around time is 16.250000average working time is 10.250000
Process returned 0 (0x0)   execution time : 51.733 s
Press any key to continue.
```

3.priority(preemptive)scheduling

```
#include <stdio.h>

#define MAX 10

void priority_non_preemptive(int n, int at[], int bt[], int p[]) {
    int ct[MAX] = {0};
    int tat[MAX] = {0};
    int wt[MAX] = {0};

    int total_wt = 0;
    int total_tat = 0;


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


    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (p[i] < p[j]) {

                int temp = at[i];
                at[i] = at[j];
                at[j] = temp;


                temp = bt[i];
                bt[i] = bt[j];
                bt[j] = temp;


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


    ct[0] = at[0] + bt[0];
    tat[0] = ct[0] - at[0];
    wt[0] = tat[0] - bt_copy[0];
    total_wt += wt[0];
    total_tat += tat[0];

    for (int i = 1; i < n; i++) {
        ct[i] = ct[i - 1] + bt[i];
        tat[i] = ct[i] - at[i];
        wt[i] = tat[i] - bt_copy[i];
        total_wt += wt[i];
        total_tat += tat[i];
    }


    printf("\nProcess\tArrival Time\tBurst Time\tPriority\tCompletion Time\tTurnaround Time\tWaiting Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i+1, at[i], bt_copy[i], p[i], ct[i], tat[i], wt[i]);
    }

    printf("\nAverage waiting time: %.2f", (float)total_wt / n);
    printf("\nAverage turnaround time: %.2f", (float)total_tat / n);
}

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

    int at[MAX], bt[MAX], p[MAX];

    printf("Enter the arrival time:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &at[i]);
    }

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

    printf("Enter the priority:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &p[i]);
    }

    priority_non_preemptive(n, at, bt, p);

    return 0;
}
```



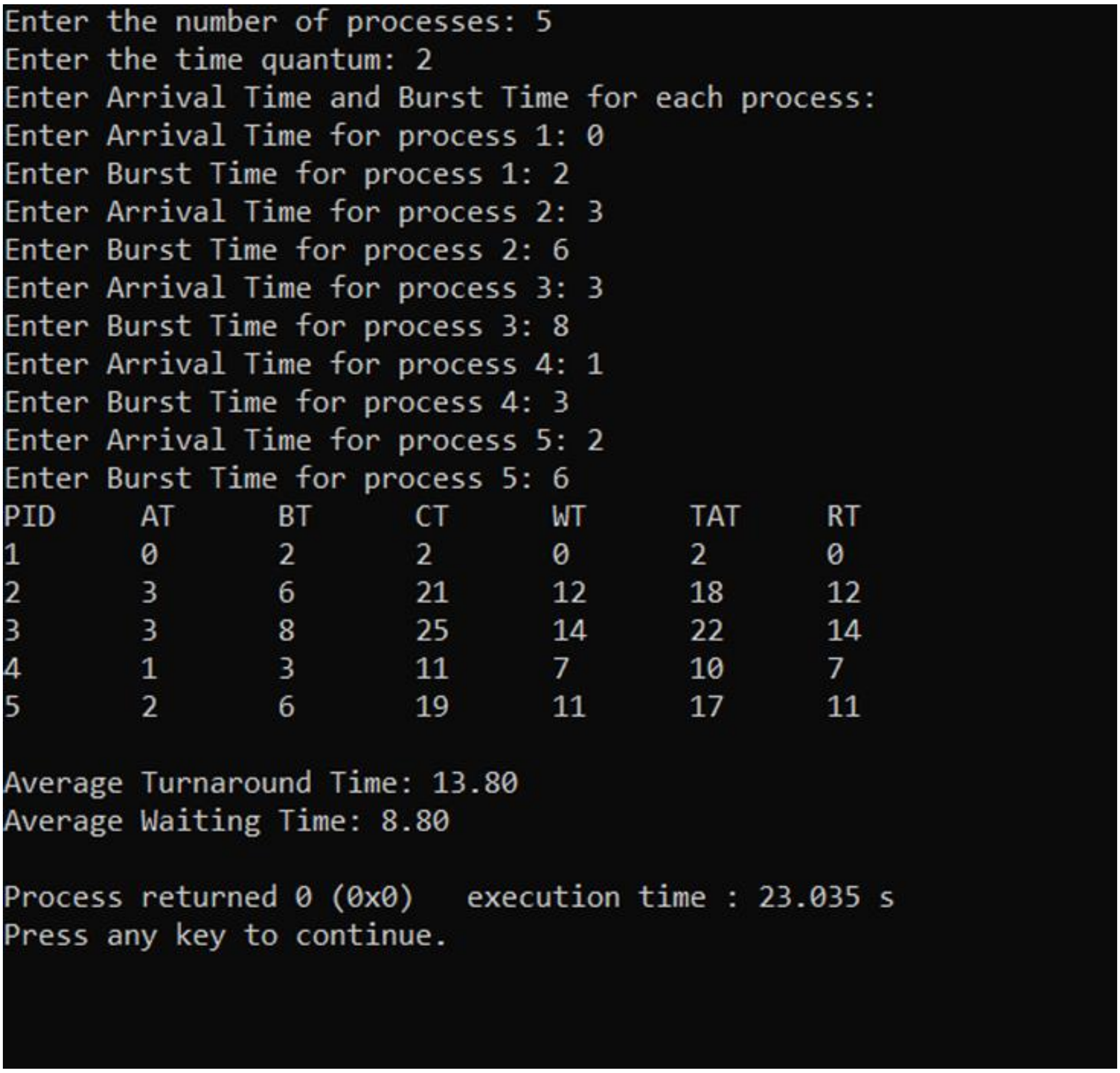
```
Enter the number of processes: 4
Enter the arrival time:
0
1
2
4
Enter the burst time:
5
4
2
0
Enter the priority:
10
20
30
40

Process Arrival Time    Burst Time      Priority        Completion Time Turnaround Time Waiting Time
1       4               5               40              4               0               -5
2       2               4               30              6               4               0
3       1               2               20              10              9               7
4       0               0               10              15              15              15

Average waiting time: 4.25
Average turnaround time: 7.00
Process returned 0 (0x0)   execution time : 40.767 s
Press any key to continue.
```

4.Round robin scheduling.

```
 #include struct Process
 { int pid;
 int burst_time;
int arrival_time;
int remaining_time;
};
void roundRobin(struct Process processes[], int n, int time_quantum)
{
int remaining_processes = n;
int current_time = 0;
int completed[n];
int ct[n], wt[n], tat[n], rt[n];
 for (int i = 0; i < n; i++)
{ completed[i] = 0;
 }
While
 (remaining_processes > 0)
```

```
{
 for (int i = 0; i < n; i++)
 {
if (completed[i] == 0 && processes[i].arrival_time <= current_time)
 {
if (processes[i].remaining_time > 0)
{
 if (processes[i].remaining_time <= time_quantum)
 {
current_time += processes[i].remaining_time;
processes[i].remaining_time = 0;
completed[i] = 1; remaining_processes--;
 ct[i] = current_time;
 tat[i] = ct[i]- processes[i].arrival_time;
 }
Else
 {
current_time += time_quantum;
processes[i].remaining_time-= time_quantum;
}
 wt[i] = ct[i]- processes[i].arrival_time- processes[i].burst_time; rt[i] = wt[i];
 }
 }
}
}
printf("PID\tAT\tBT\tCT\tWT\tTAT\tRT\n");
float avg_tat = 0, avg_wt = 0; for (int i = 0; i < n; i++)
{
printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\n", processes[i].pid, processes[i].arrival_time,
processes[i].burst_time, ct[i], wt[i], tat[i], rt[i]); avg_tat += tat[i]; avg_wt += wt[i];
 }
```

```
avg_tat /= n; avg_wt /= n;
printf("\nAverage Turnaround Time: %.2f\n", avg_tat);
 printf("Average Waiting Time: %.2f\n", avg_wt);
 }
 int main() { int n, time_quantum; printf("Enter the number of processes: ");
scanf("%d", &n);
 printf("Enter the time quantum: ");
scanf("%d", &time_quantum);
struct Process processes[n];
 printf("Enter Arrival Time and Burst Time for each process:\n");
 for (int i = 0; i < n; i++)
 {
printf("Enter Arrival Time for process %d: ", i+1);
scanf("%d", & processes[i].arrival_time);
 printf("Enter Burst Time for process %d: ", i+1);
scanf("%d", & processes[i].burst_time);
processes[i].pid = i+1; processes[i].remaining_time = processes[i].burst_time;
 }
roundRobin(processes, n, time_quantum);
 return 0;
}
```



```
Enter the number of processes: 5
Enter the time quantum: 2
Enter Arrival Time and Burst Time for each process:
Enter Arrival Time for process 1: 0
Enter Burst Time for process 1: 2
Enter Arrival Time for process 2: 3
Enter Burst Time for process 2: 6
Enter Arrival Time for process 3: 3
Enter Burst Time for process 3: 8
Enter Arrival Time for process 4: 1
Enter Burst Time for process 4: 3
Enter Arrival Time for process 5: 2
Enter Burst Time for process 5: 6
PID     AT      BT      CT      WT      TAT     RT
1       0       2       2       0       2       0
2       3       6       21      12      18      12
3       3       8       25      14      22      14
4       1       3       11      7       10      7
5       2       6       19      11      17      11

Average Turnaround Time: 13.80
Average Waiting Time: 8.80

Process returned 0 (0x0)   execution time : 23.035 s
Press any key to continue.
```