```
moujanmirjalili@ubuntu:~/Desktop/Lab8$ ./FCFS
Enter total number of processes(maximum 10): 4

Enter Process Burst Time
Process 1: 8
Process 2: 2
Process 3: 5
Process 4: 14

Process         Burst Time      Waiting Time    Turnaround Time
Process 1               8               0               8
Process 2               2               8               10
Process 3               5               10              15
Process 4               14              15              29

Average Waiting Time:8.25
Average Turnaround Time:15.50
moujanmirjalili@ubuntu:~/Desktop/Lab8$
```

در اینجا به دلیل اینکه فرآیندی با زمان کوتاه ممکن است پشت فرآیندی با زمان طولانی منتظر بماند، الگوریتم بهینه نیست و ممکن است convoy effect رخ دهد.

```c
#include<stdio.h>

typedef struct {
    int pid;
    int bt;
    int wt;
    int tt;
}Process;

int main()
{
    Process process[10];
    int n, i, j;
    float avg_wt = 0, avg_tt = 0;
    printf("Enter total number of processes(maximum 10): ");
    scanf("%d",&n);

    printf("\nEnter Process Burst Time\n");
    for(i = 0; i < n;i++)
    {
        process[i].pid = i+1;
        printf("Process %d: ",process[i].pid);
        scanf("%d",&process[i].bt);
    }

    process[0].wt = 0;  //waiting time for first process is 0

    //calculating waiting time
    for(i = 1; i < n;i++) {
        process[i].wt = 0;
        for(j = 0; j < i;j++)
            process[i].wt += process[j].bt;
    }

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

    //calculating turnaround time
    for(i = 0; i < n;i++) {
        process[i].tt = process[i].bt + process[i].wt;
        avg_wt += process[i].wt;
        avg_tt += process[i].tt;
        printf("\nProcess %d\t\t%d\t\t%d\t\t%d", process[i].pid,process[i].bt,
        process[i].wt,process[i].tt);
    }

    avg_wt /= n;
    avg_tt /= n;
    printf("\n\nAverage Waiting Time:%0.2f", avg_wt);
    printf("\nAverage Turnaround Time:%0.2f\n", avg_tt);

    return 0;
}
```

```
moujanmirjalili@ubuntu:~/Desktop/Lab8$ gcc -o SJF SJF.c
moujanmirjalili@ubuntu:~/Desktop/Lab8$ ./SJF
Enter number of process:4

Enter Process Burst Time
Process 1: 8
Process 2: 2
Process 3: 5
Process 4: 14

Process         Burst Time      Waiting Time    Turnaround Time
Process 2               2               0               2
Process 3               5               2               7
Process 1               8               7               15
Process 4               14              15              29

Average Waiting Time=6.00
Average Turnaround Time=13.25
```

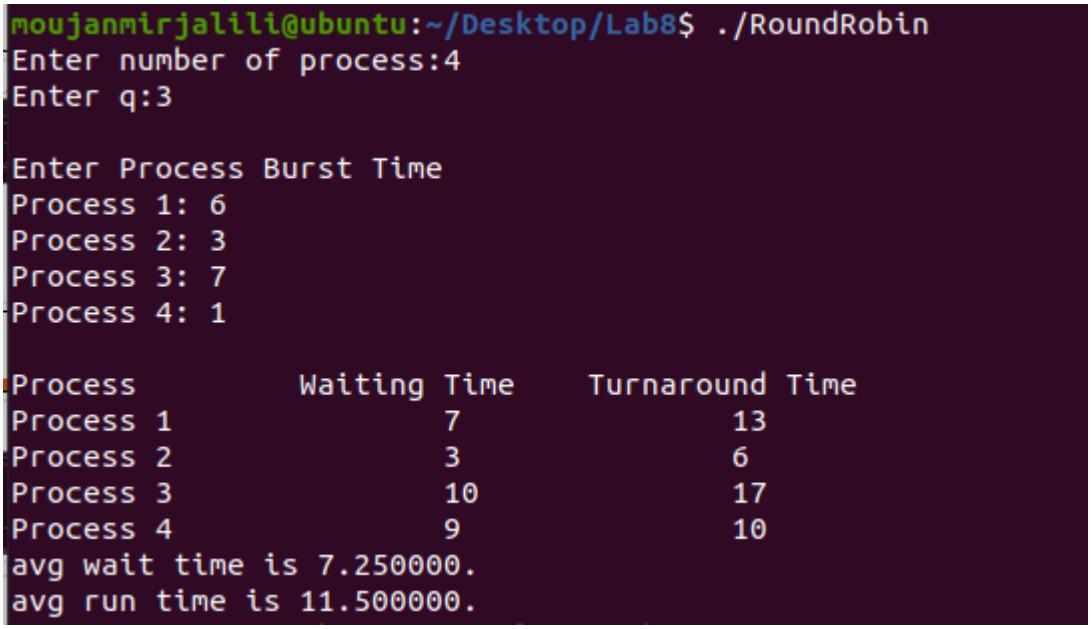
در اینجا کمترین زمان انتظار را داریم و هر فرآیند با burst time کمتر، زودتر اجرا می‌شود.

```c
#include<stdio.h>

typedef struct {
    int pid;
    int bt;
    int wt;
    int tt;
}Process;

void main()
{
    Process process[10];
    int n, i, j, total = 0, pos, temp;
    float avg_wt = 0, avg_tt = 0;

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

    printf("\nEnter Process Burst Time\n");
    for(i = 0; i < n;i++)
    {
        process[i].pid = i+1;
        printf("Process %d: ",process[i].pid);
        scanf("%d",&process[i].bt);
    }


    //sorting burst time in ascending order using selection sort
    for(i = 0; i < n; i++) {
        pos = i;
        for(j = i+1; j < n;j++) {
            if(process[j].bt < process[pos].bt)
                pos = j;
        }

        temp = process[i].bt;
        process[i].bt = process[pos].bt;
        process[pos].bt = temp;

        temp = process[i].pid;
        process[i].pid = process[pos].pid;
        process[pos].pid = temp;
    }

    process[0].wt = 0;    //waiting time for first process will be zero

    //calculate waiting time
    for(i = 1; i < n; i++)
    {
        process[i].wt = 0;
        for(j = 0; j < i; j++)
            process[i].wt += process[j].bt;

        total += process[i].wt;
    }

    avg_wt=(float)total/n;      //average waiting time
    total=0;

    printf("\nProcess\t\tBurst Time \tWaiting Time  \tTurnaround Time");
    for(i = 0; i < n; i++)
    {
        //calculate turnaround time
        process[i].tt = process[i].bt + process[i].wt;
        total += process[i].tt;
        printf("\nProcess %d\t\t%d\t\t%d\t\t%d", process[i].pid, process[i].bt,
process[i].wt, process[i].tt);
    }

    avg_tt = (float)total/n;     //average turnaround time
    printf("\n\nAverage Waiting Time=%0.2f",avg_wt);
    printf("\nAverage Turnaround Time=%0.2f\n",avg_tt);
}
```

```
moujanmirjalili@ubuntu:~/Desktop/88$ gcc -o priority priority.c
moujanmirjalili@ubuntu:~/Desktop/88$ ./priority
Enter Total Number of Process:4

Enter Burst Time and Priority

Process 1
Burst Time:21
Priority:2

Process 2
Burst Time:3
Priority:1

Process 3
Burst Time:6
Priority:4

Process 4
Burst Time:2
Priority:3

Process         Burst Time      Waiting Time    Turnaround Time
Process 2               3               0               3
Process 1               21              3               24
Process 4               2               24              26
Process 3               6               26              32

Average Waiting Time=13.25
Average Turnaround Time=21.25
```

در اینجا فرآیند با الویت بالاتر، زودتر انجام می‌شود.

```c
#include<stdio.h>

typedef struct {
    int pid;
    int bt;
    int pr; // priority
    int wt;
    int tt;
}Process;

int main()
{
    Process process[10];
    int n, i, j, total = 0, pos, temp;
    float avg_wt = 0, avg_tt = 0;
    printf("Enter Total Number of Process:");
    scanf("%d",&n);

    printf("\nEnter Burst Time and Priority\n");
    for(i = 0; i < n; i++)
    {
        process[i].pid = i+1;
        printf("\nProcess %d\n",i+1);
        printf("Burst Time:");
        scanf("%d",&process[i].bt);
        printf("Priority:");
        scanf("%d",&process[i].pr);
    }

    //sorting burst time, priority and process number in ascending order using selection sort
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j = i+1; j < n; j++)
        {
            if(process[j].pr < process[pos].pr)
                pos = j;
        }

        temp=process[i].pr;
        process[i].pr = process[pos].pr;
        process[pos].pr = temp;

        temp = process[i].bt;
        process[i].bt = process[pos].bt;
        process[pos].bt = temp;

        temp = process[i].pid;
        process[i].pid = process[pos].pid;
        process[pos].pid = temp;
    }

    process[0].wt = 0; //waiting time for first process is zero

    //calculate waiting time
    for(i = 1; i < n; i++)
    {
        process[i].wt = 0;
        for(j = 0; j < i; j++)
            process[i].wt += process[j].bt;

        total += process[i].wt;
    }

    avg_wt = (float)total/n;      //average waiting time
    total=0;

    printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");
    for(i = 0; i < n; i++)
    {
        //calculate turnaround time
        process[i].tt = process[i].bt + process[i].wt;
        total += process[i].tt;
        printf("\nProcess %d\t\t%d\t\t%d\t\t%d", process[i].pid, process[i].bt, process[i].wt, process[i].tt);
    }

    avg_tt = (float)total/n;     //average turnaround time

    printf("\n\nAverage Waiting Time=%0.2f",avg_wt);
    printf("\nAverage Turnaround Time=%0.2f\n",avg_tt);

return 0;
}
```



در این فرآیند پردازه‌ها محدودیت زمانی دارند و همه برنامه‌ها تا حد خوبی پیش می‌روند و زمان انتظار بهینه تر است.

```c
#include<stdio.h>

typedef struct {
    int pid;
    int bt;
    int wt;
    int tt;
}Process;

int n, q, cur, total_rn, total_wt;
int tmp[100], l, r;

int main() {
        Process process[10];
        printf("Enter number of process:");
        scanf("%d",&n);
        printf("Enter q:");
        scanf("%d",&q);

        printf("\nEnter Process Burst Time\n");
        for(int i = 0; i < n; i ++) {
                printf("Process %d: ",i+1);
                scanf("%d", &process[i].bt);
                tmp[r ++] = i;
                process[i].wt = -process[i].bt;
        }

        for(; l < r; l ++) {
                int i = tmp[l];
                if(process[i].bt <= q) {
                        cur += process[i].bt;
                        process[i].tt = cur;
                        process[i].wt += process[i].tt;
                }
                else {
                        cur += q;
                        process[i].bt -= q;
                        tmp[r ++] = i;
                }
        }
        printf("\nProcess\t\tWaiting Time\tTurnaround Time");
        for(int i = 0; i < n; i ++) {
        printf("\nProcess %d\t\t%d\t\t%d", i+1, process[i].wt, process[i].tt);
                total_wt += process[i].wt;
                total_rn += process[i].tt;
        }
        printf("\n");
        float avg_wait = 1.0 * total_wt / n;
        float avg_run = 1.0 * total_rn / n;
        printf("avg wait time is %f.\n", avg_wait);
        printf("avg run time is %f.\n", avg_run);
}
```