MSSV: 21522683

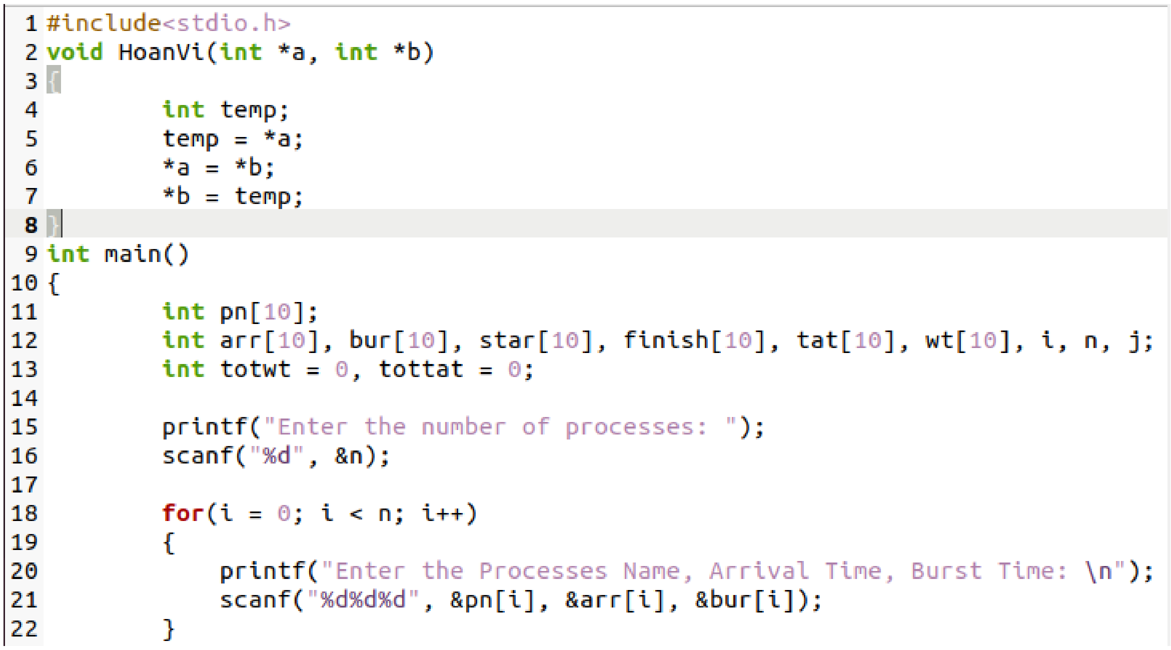
Lớp: IT007.N17.1

Họ tên: Phan Trọng Tính

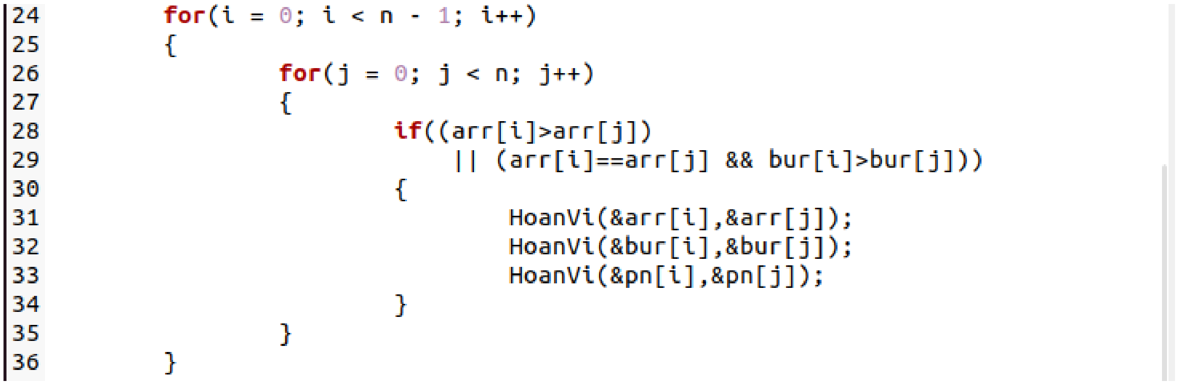
BÁO CÁO THỰC HÀNH HDH LAB 4

⇨ Tự kiểm tra kết quả của chương trình và gỡ lỗi nếu cần thiết. Sau đó bổ sung code để tính average waiting time và average turnaround.

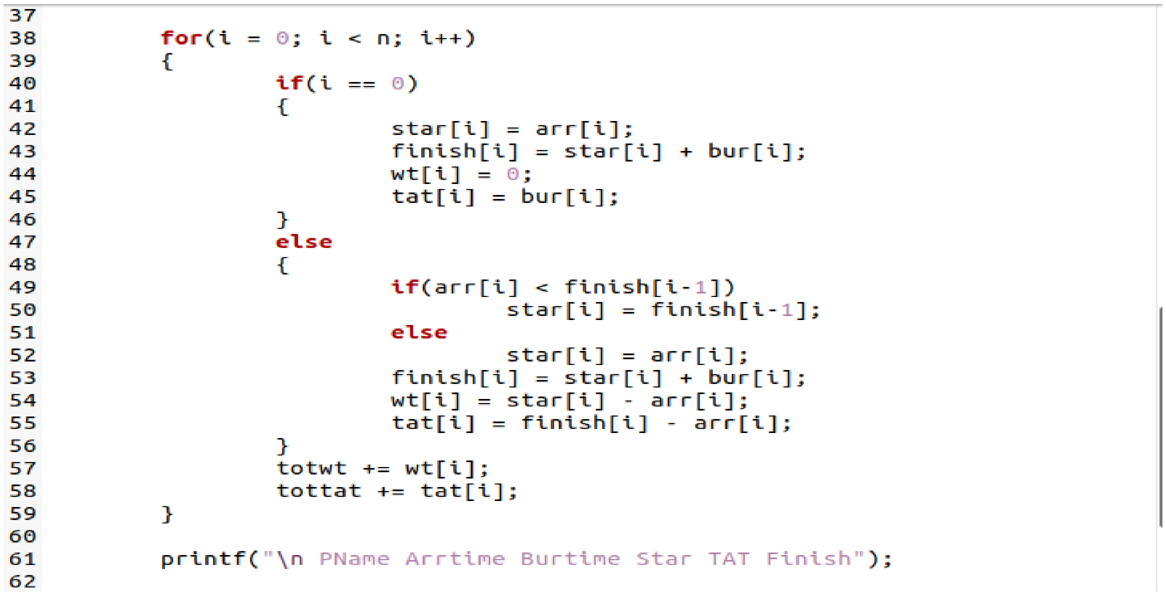
⇨ Source code FCFS sau khi gỡ lỗi và bổ sung:



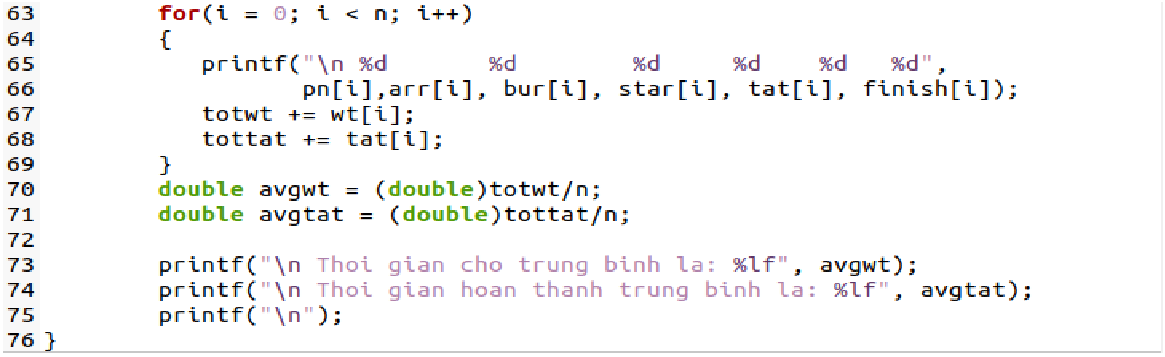
```
#include<stdio.h>
void HoanVi(int *a, int *b)
{
        int temp;
        temp = *a;
        *a = *b;
        *b = temp;
}
int main()
{
        int pn[10];
        int arr[10], bur[10], star[10], finish[10], tat[10], wt[10], i, n, j;
        int totwt = 0, tottat = 0;

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

        for(i = 0; i < n; i++)
        {
            printf("Enter the Processes Name, Arrival Time, Burst Time: \n");
            scanf("%d%d%d", &pn[i], &arr[i], &bur[i]);
        }
```



```
        for(i = 0; i < n - 1; i++)
        {
                for(j = 0; j < n; j++)
                {
                        if((arr[i]>arr[j])
                            || (arr[i]==arr[j] && bur[i]>bur[j]))
                        {
                                HoanVi(&arr[i],&arr[j]);
                                HoanVi(&bur[i],&bur[j]);
                                HoanVi(&pn[i],&pn[j]);
                        }
                }
        }
```

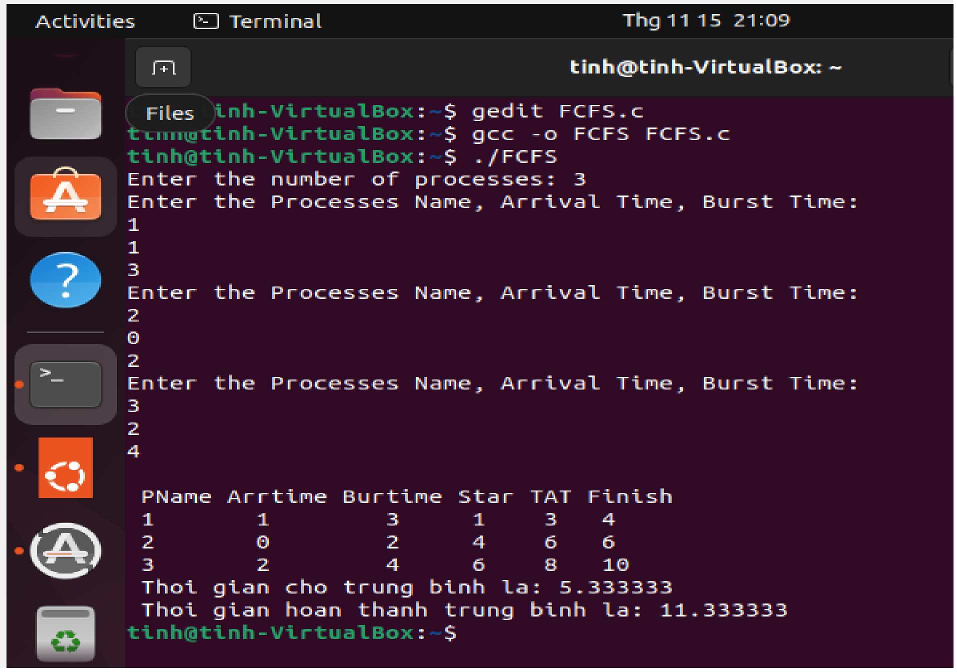


```
        for(i = 0; i < n; i++)
        {
                if(i == 0)
                {
                        star[i] = arr[i];
                        finish[i] = star[i] + bur[i];
                        wt[i] = 0;
                        tat[i] = bur[i];
                }
                else
                {
                        if(arr[i] < finish[i-1])
                                star[i] = finish[i-1];
                        else
                                star[i] = arr[i];
                        finish[i] = star[i] + bur[i];
                        wt[i] = star[i] - arr[i];
                        tat[i] = finish[i] - arr[i];
                }
                totwt += wt[i];
                tottat += tat[i];
        }

        printf("\n PName Arrtime Burtime Star TAT Finish");

```



```
        for(i = 0; i < n; i++)
        {
           printf("\n %d       %d        %d     %d    %d   %d",
                  pn[i],arr[i], bur[i], star[i], tat[i], finish[i]);
           totwt += wt[i];
           tottat += tat[i];
        }
        double avgwt = (double)totwt/n;
        double avgtat = (double)tottat/n;

        printf("\n Thoi gian cho trung binh la: %lf", avgwt);
        printf("\n Thoi gian hoan thanh trung binh la: %lf", avgtat);
        printf("\n");
}
```

Kết quả chạy thử:

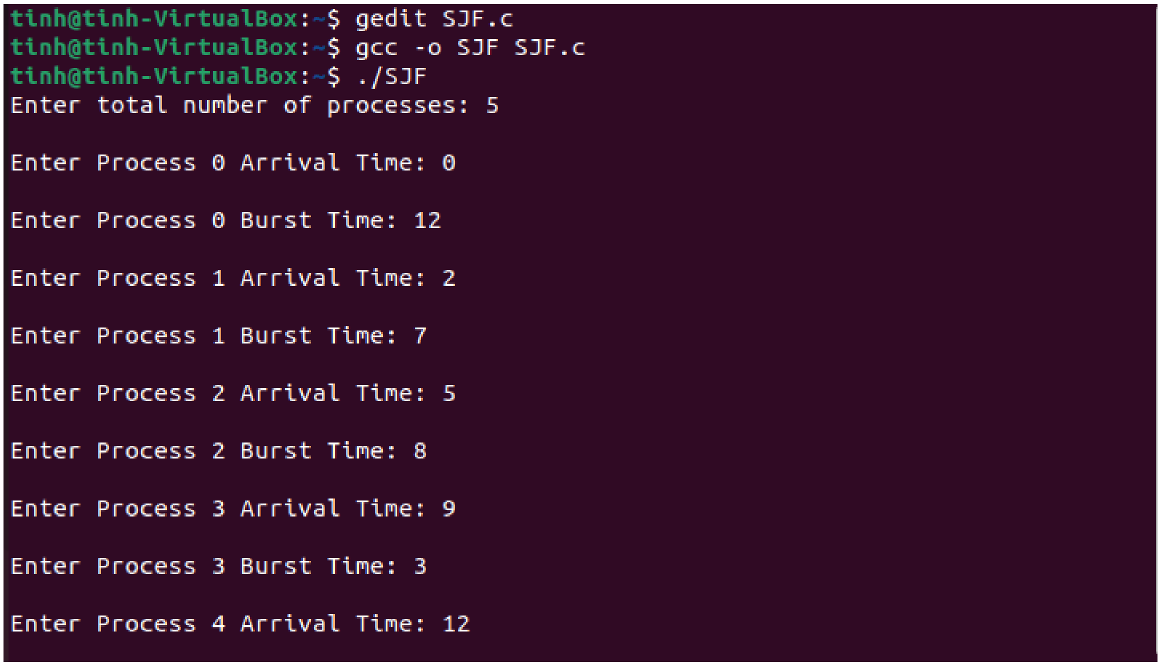


```
tinh@tinh-VirtualBox:~$ gedit FCFS.c
tinh@tinh-VirtualBox:~$ gcc -o FCFS FCFS.c
tinh@tinh-VirtualBox:~$ ./FCFS
Enter the number of processes: 3
Enter the Processes Name, Arrival Time, Burst Time:
1
1
3
Enter the Processes Name, Arrival Time, Burst Time:
2
0
2
Enter the Processes Name, Arrival Time, Burst Time:
3
2
4

 PName Arrtime Burtime Star TAT Finish
 1       1        3     1    3   4
 2       0        2     4    6   6
 3       2        4     6    8   10
 Thoi gian cho trung binh la: 5.333333
 Thoi gian hoan thanh trung binh la: 11.333333
tinh@tinh-VirtualBox:~$
```

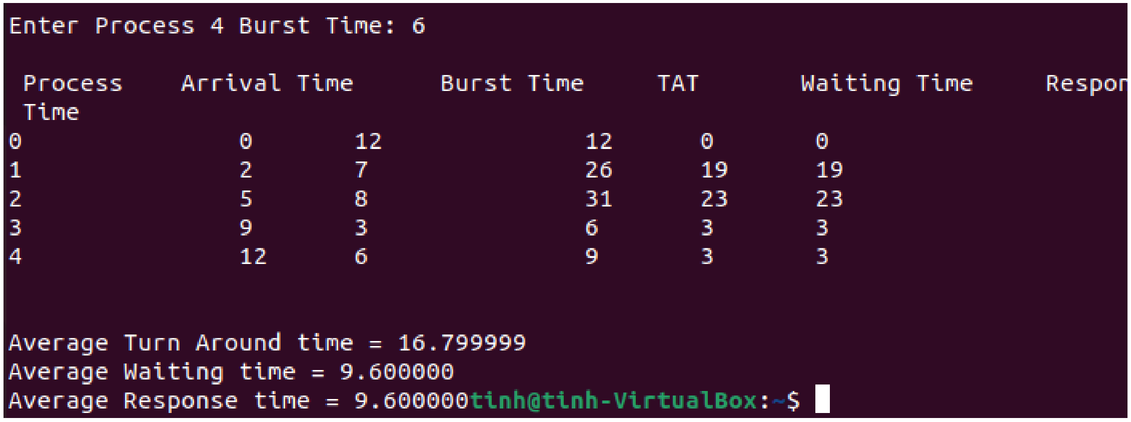
Câu 1: Viết chương trình mô phỏng giải thuật SJF với các yêu cầu sau:

- ⇨ **Nhập số lượng process**
- ⇨ **Nhập process name, arrival time, burst time**
- ⇨ **In ra Process name, response time, waiting time, turnaround time, average waiting time, average turnaround time**

Kết quả chạy SJF:

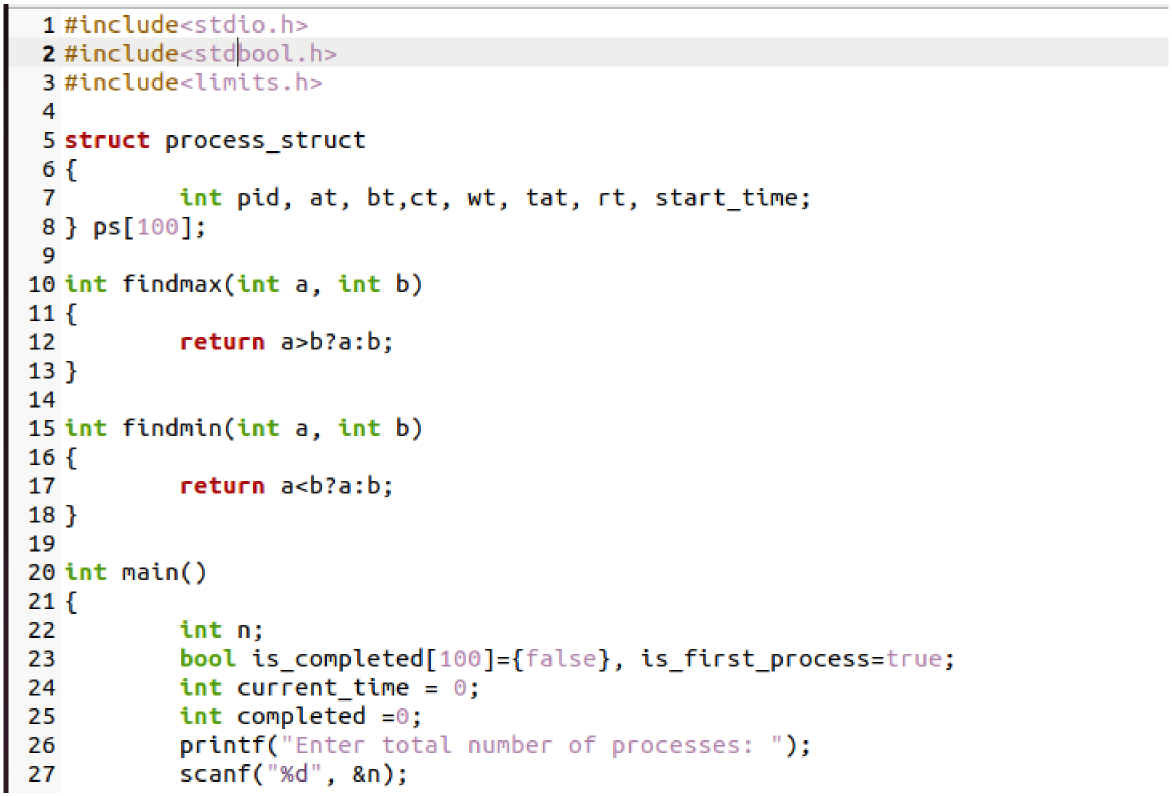


```
tinh@tinh-VirtualBox:~$ gedit SJF.c
tinh@tinh-VirtualBox:~$ gcc -o SJF SJF.c
tinh@tinh-VirtualBox:~$ ./SJF
Enter total number of processes: 5

Enter Process 0 Arrival Time: 0

Enter Process 0 Burst Time: 12

Enter Process 1 Arrival Time: 2

Enter Process 1 Burst Time: 7

Enter Process 2 Arrival Time: 5

Enter Process 2 Burst Time: 8

Enter Process 3 Arrival Time: 9

Enter Process 3 Burst Time: 3

Enter Process 4 Arrival Time: 12
```

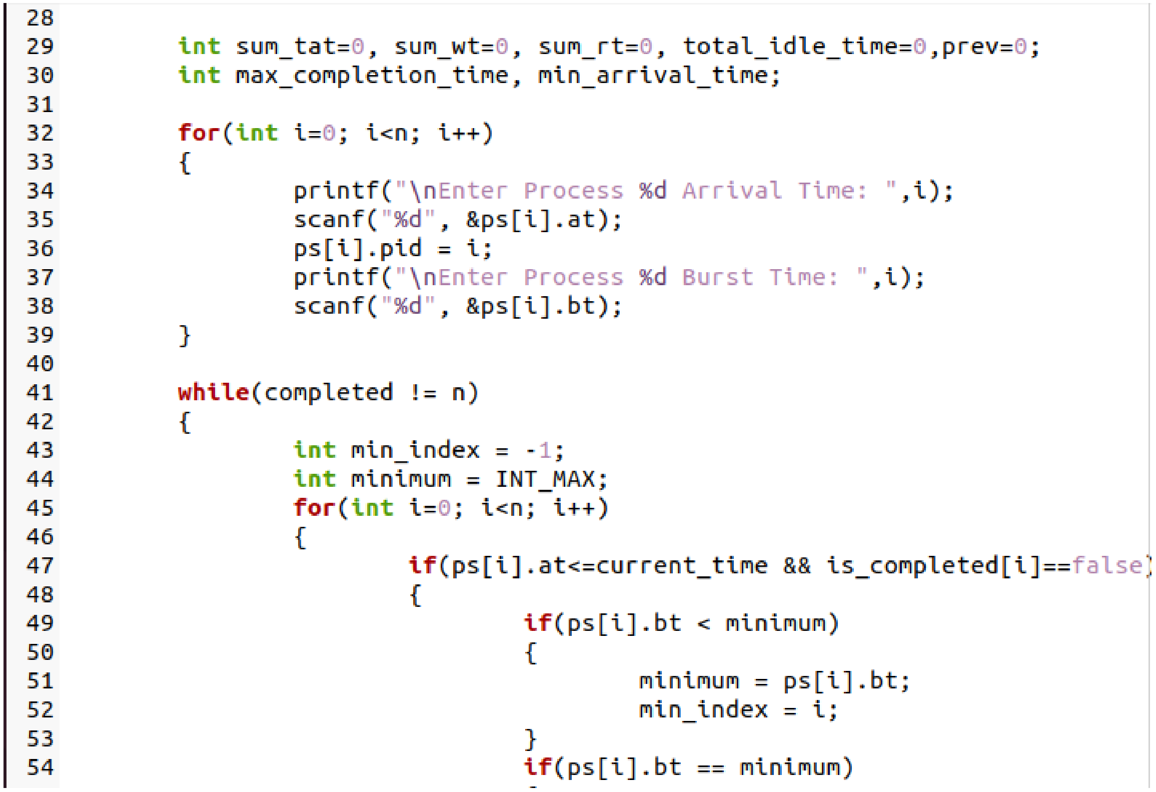


```
Enter Process 4 Burst Time: 6

 Process     Arrival Time       Burst Time      TAT       Waiting Time      Respon
 Time
0                0       12              12       0        0
1                2       7               26       19       19
2                5       8               31       23       23
3                9       3               6        3        3
4                12      6               9        3        3


Average Turn Around time = 16.799999
Average Waiting time = 9.600000
Average Response time = 9.600000tinh@tinh-VirtualBox:~$
```

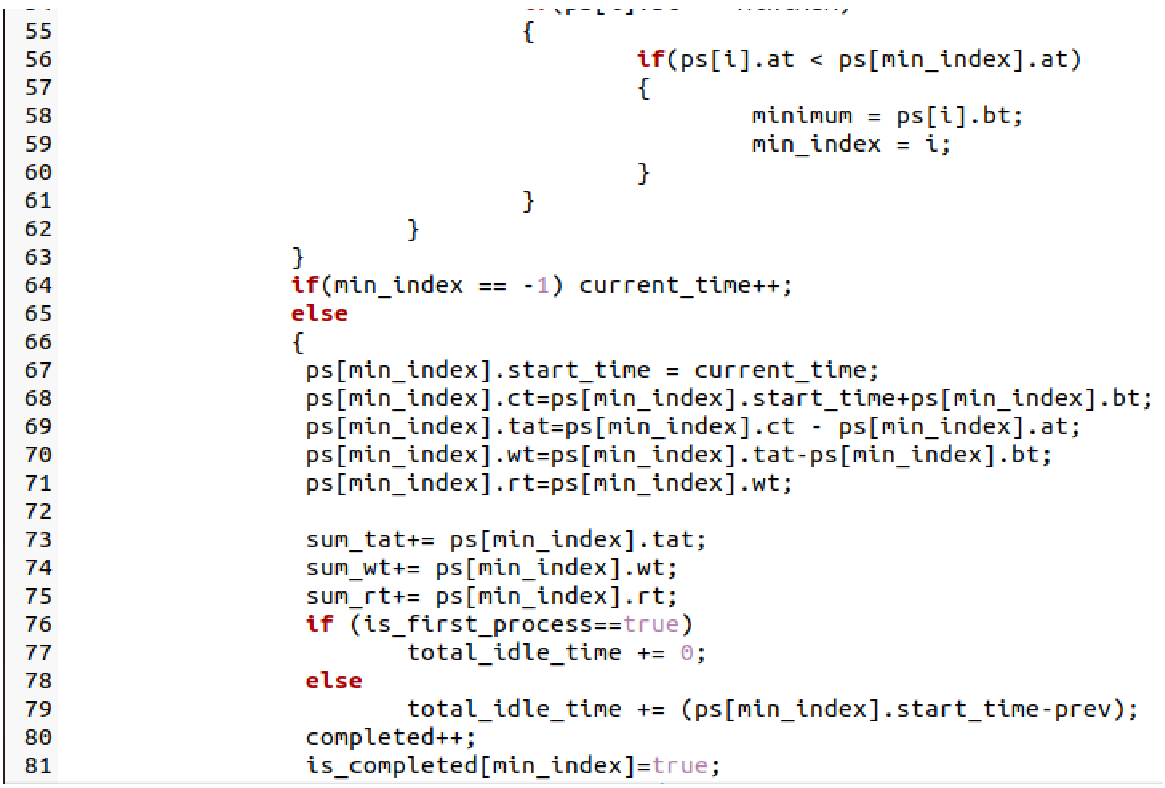
Source code SJF với ngôn ngữ C:



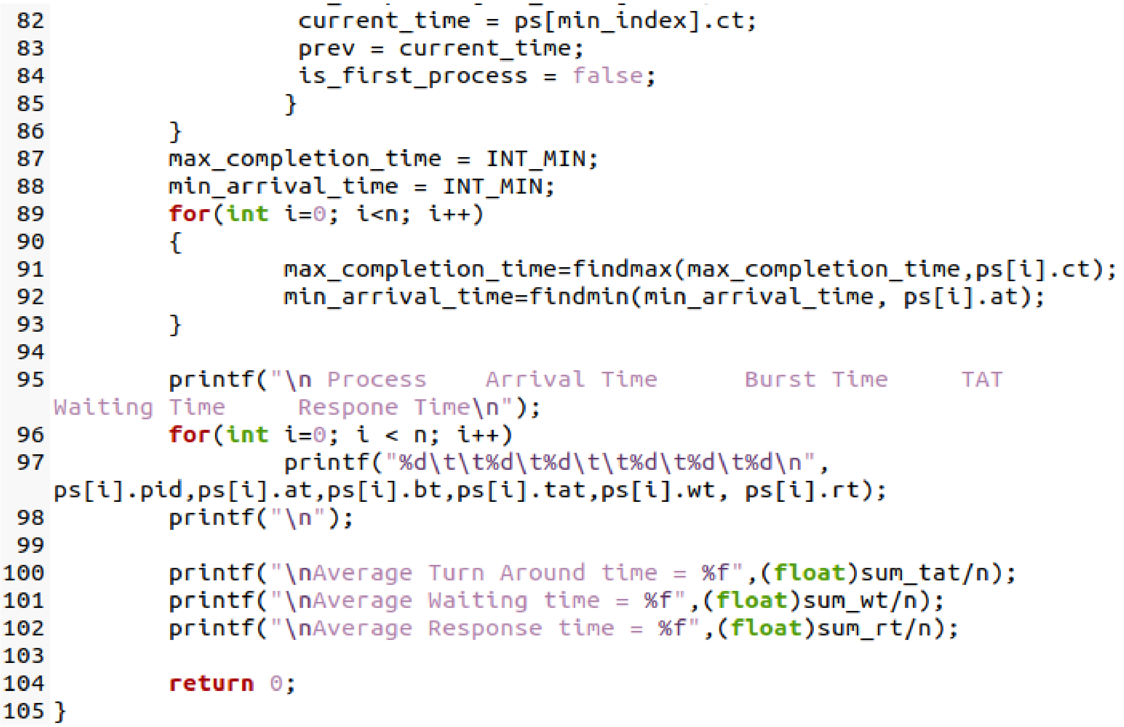
```
#include<stdio.h>
#include<stdbool.h>
#include<limits.h>

struct process_struct
{
        int pid, at, bt,ct, wt, tat, rt, start_time;
} ps[100];

int findmax(int a, int b)
{
        return a>b?a:b;
}

int findmin(int a, int b)
{
        return a<b?a:b;
}

int main()
{
        int n;
        bool is_completed[100]={false}, is_first_process=true;
        int current_time = 0;
        int completed =0;
        printf("Enter total number of processes: ");
        scanf("%d", &n);
```



```

        int sum_tat=0, sum_wt=0, sum_rt=0, total_idle_time=0,prev=0;
        int max_completion_time, min_arrival_time;

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

        while(completed != n)
        {
                int min_index = -1;
                int minimum = INT_MAX;
                for(int i=0; i<n; i++)
                {
                        if(ps[i].at<=current_time && is_completed[i]==false)
                        {
                                if(ps[i].bt < minimum)
                                {
                                        minimum = ps[i].bt;
                                        min_index = i;
                                }
                                if(ps[i].bt == minimum)
```



```
                                if(ps[i].bt == minimum)
                                {
                                        if(ps[i].at < ps[min_index].at)
                                        {
                                                minimum = ps[i].bt;
                                                min_index = i;
                                        }
                                }
                        }
                }
                if(min_index == -1) current_time++;
                else
                {
                 ps[min_index].start_time = current_time;
                 ps[min_index].ct=ps[min_index].start_time+ps[min_index].bt;
                 ps[min_index].tat=ps[min_index].ct - ps[min_index].at;
                 ps[min_index].wt=ps[min_index].tat-ps[min_index].bt;
                 ps[min_index].rt=ps[min_index].wt;

                 sum_tat+= ps[min_index].tat;
                 sum_wt+= ps[min_index].wt;
                 sum_rt+= ps[min_index].rt;
                 if (is_first_process==true)
                        total_idle_time += 0;
                 else
                        total_idle_time += (ps[min_index].start_time-prev);
                 completed++;
                 is_completed[min_index]=true;
```

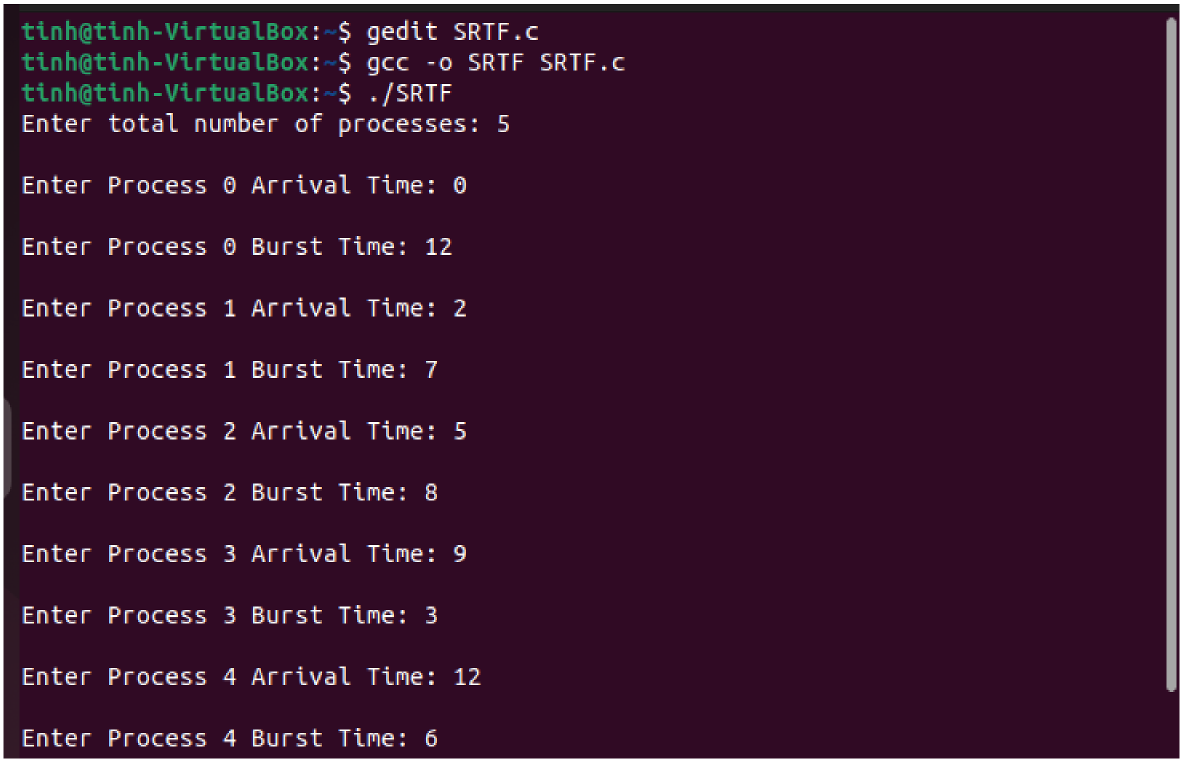


```
                 current_time = ps[min_index].ct;
                 prev = current_time;
                 is_first_process = false;
                }
        }
        max_completion_time = INT_MIN;
        min_arrival_time = INT_MIN;
        for(int i=0; i<n; i++)
        {
                max_completion_time=findmax(max_completion_time,ps[i].ct);
                min_arrival_time=findmin(min_arrival_time, ps[i].at);
        }

        printf("\n Process    Arrival Time      Burst Time     TAT
Waiting Time      Respone Time\n");
        for(int i=0; i < n; i++)
                printf("%d\t\t%d\t%d\t\t%d\t%d\t%d\n",
ps[i].pid,ps[i].at,ps[i].bt,ps[i].tat,ps[i].wt, ps[i].rt);
        printf("\n");

        printf("\nAverage Turn Around time = %f",(float)sum_tat/n);
        printf("\nAverage Waiting time = %f",(float)sum_wt/n);
        printf("\nAverage Response time = %f",(float)sum_rt/n);

        return 0;
}
```

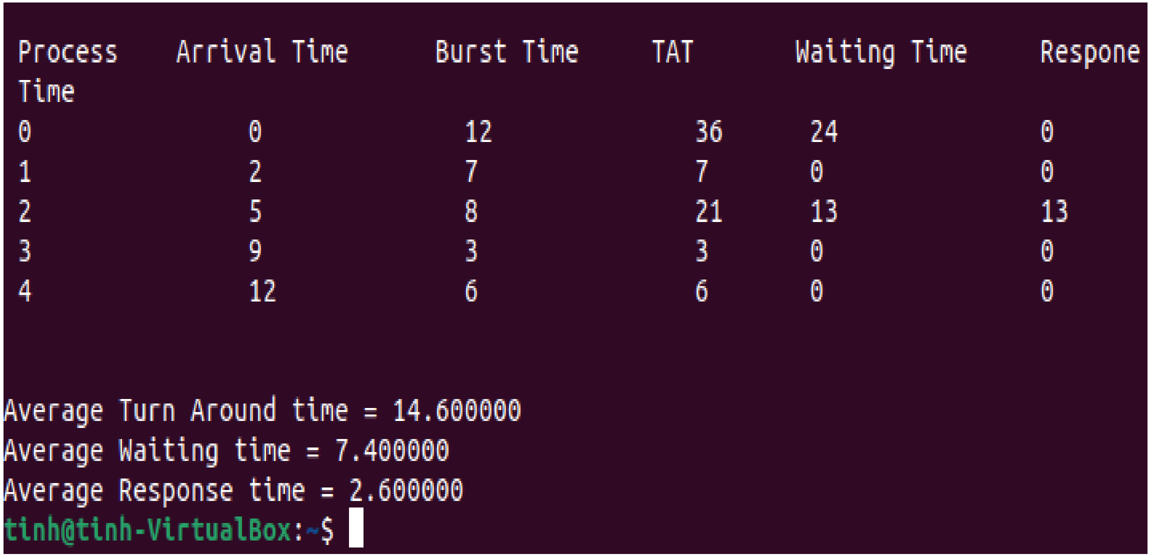
Câu 2: Viết chương trình mô phỏng giải thuật SRT với các yêu cầu sau:

Nhập số lượng process. Nhập process name, arrival time, burst time. In ra Process name, response time, waiting time, turnaround time, average waiting time, average turnaround time.

Kết quả chạy source code SRTF với ngôn ngữ C:

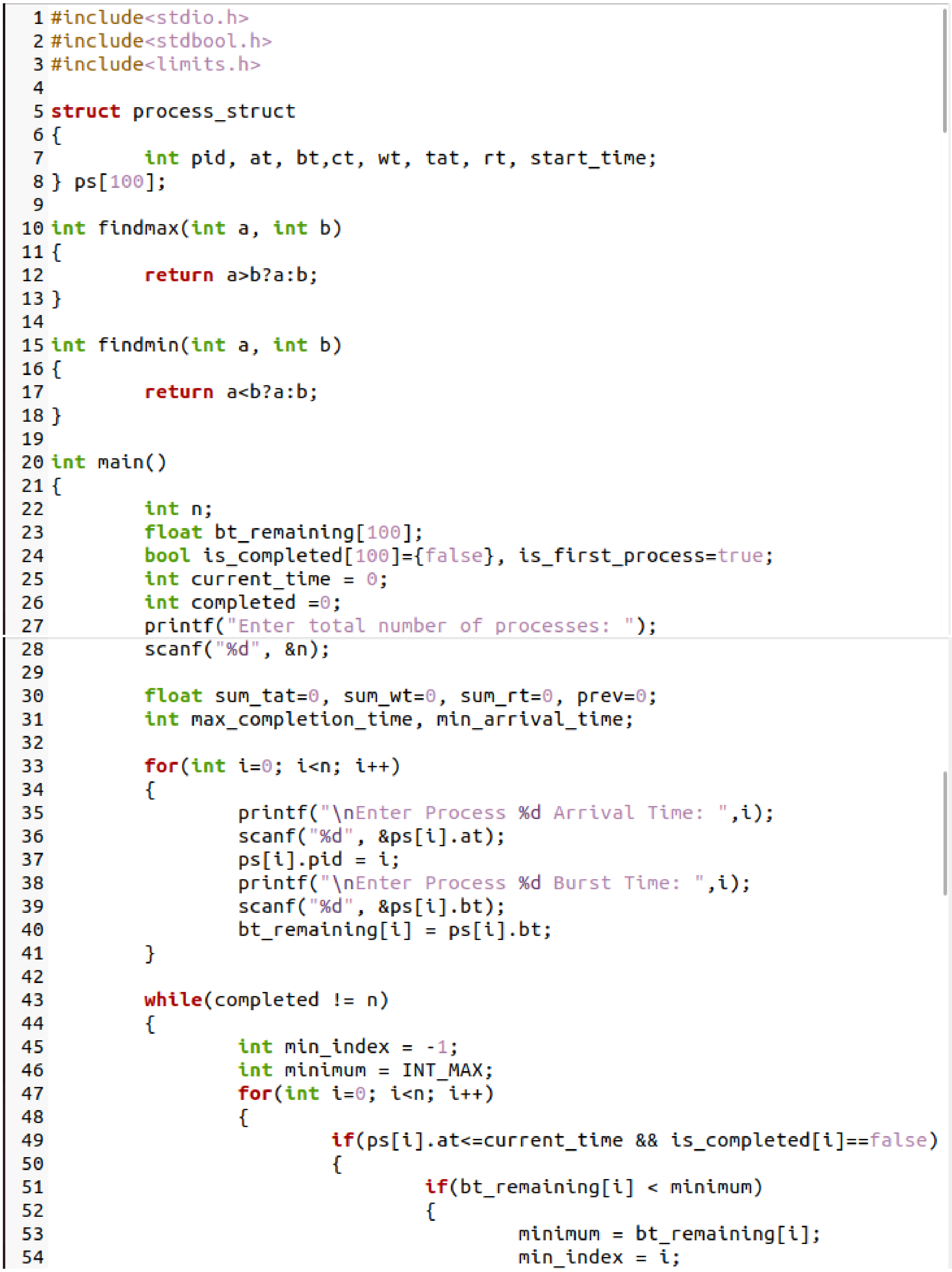


```
tinh@tinh-VirtualBox:~$ gedit SRTF.c
tinh@tinh-VirtualBox:~$ gcc -o SRTF SRTF.c
tinh@tinh-VirtualBox:~$ ./SRTF
Enter total number of processes: 5

Enter Process 0 Arrival Time: 0

Enter Process 0 Burst Time: 12

Enter Process 1 Arrival Time: 2

Enter Process 1 Burst Time: 7

Enter Process 2 Arrival Time: 5

Enter Process 2 Burst Time: 8

Enter Process 3 Arrival Time: 9

Enter Process 3 Burst Time: 3

Enter Process 4 Arrival Time: 12

Enter Process 4 Burst Time: 6
```

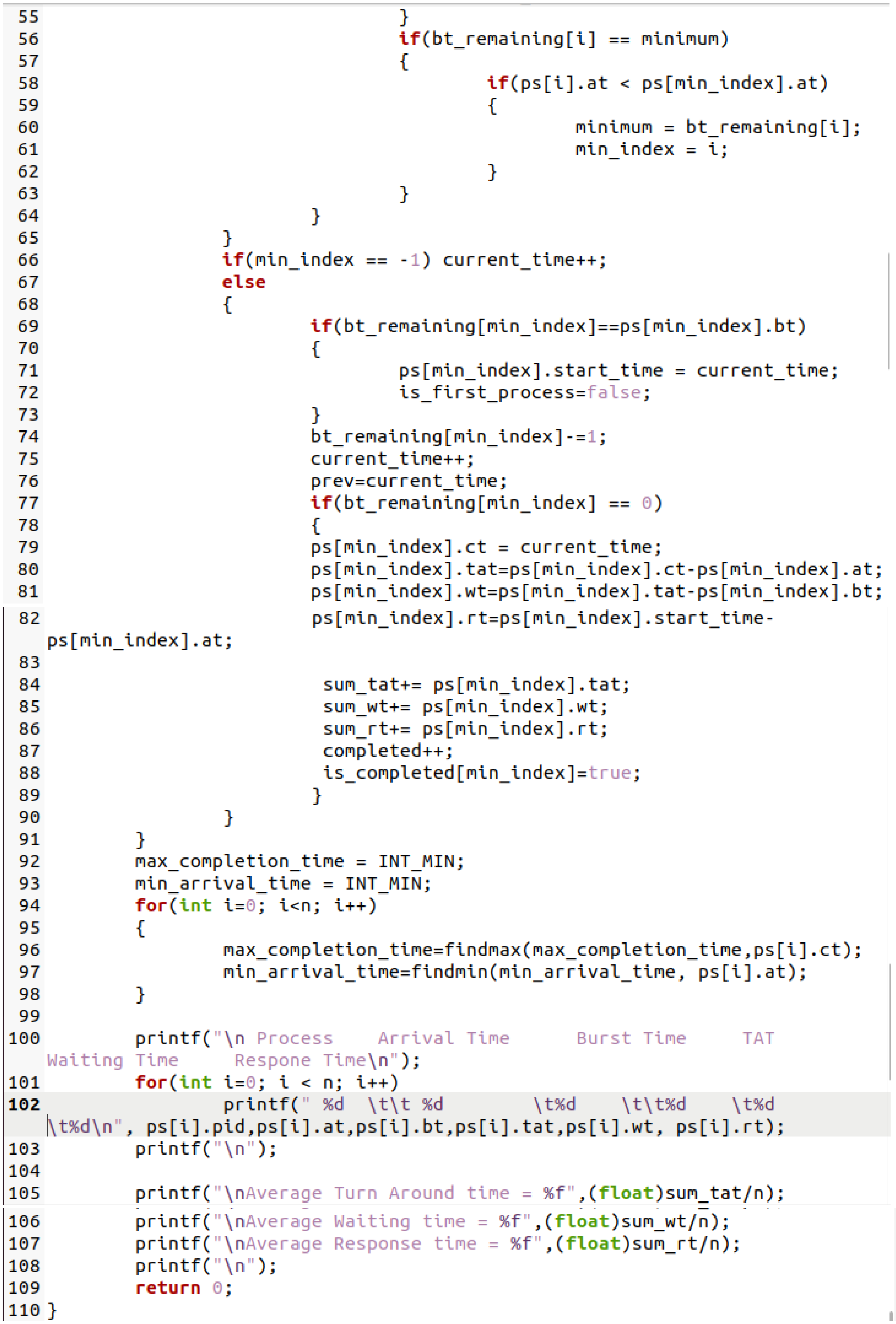


```
Process    Arrival Time     Burst Time    TAT       Waiting Time     Respone
Time
0               0              12            36      24               0
1               2              7             7       0                0
2               5              8             21      13               13
3               9              3             3       0                0
4               12             6             6       0                0


Average Turn Around time = 14.600000
Average Waiting time = 7.400000
Average Response time = 2.600000
tinh@tinh-VirtualBox:~$
```

Source code giải thuật SRTF bằng ngôn ngữ C:



```c
#include<stdio.h>
#include<stdbool.h>
#include<limits.h>

struct process_struct
{
        int pid, at, bt,ct, wt, tat, rt, start_time;
} ps[100];

int findmax(int a, int b)
{
        return a>b?a:b;
}

int findmin(int a, int b)
{
        return a<b?a:b;
}

int main()
{
        int n;
        float bt_remaining[100];
        bool is_completed[100]={false}, is_first_process=true;
        int current_time = 0;
        int completed =0;
        printf("Enter total number of processes: ");
        scanf("%d", &n);

        float sum_tat=0, sum_wt=0, sum_rt=0, prev=0;
        int max_completion_time, min_arrival_time;

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

        while(completed != n)
        {
                int min_index = -1;
                int minimum = INT_MAX;
                for(int i=0; i<n; i++)
                {
                        if(ps[i].at<=current_time && is_completed[i]==false)
                        {
                                if(bt_remaining[i] < minimum)
                                {
                                        minimum = bt_remaining[i];
                                        min_index = i;
```

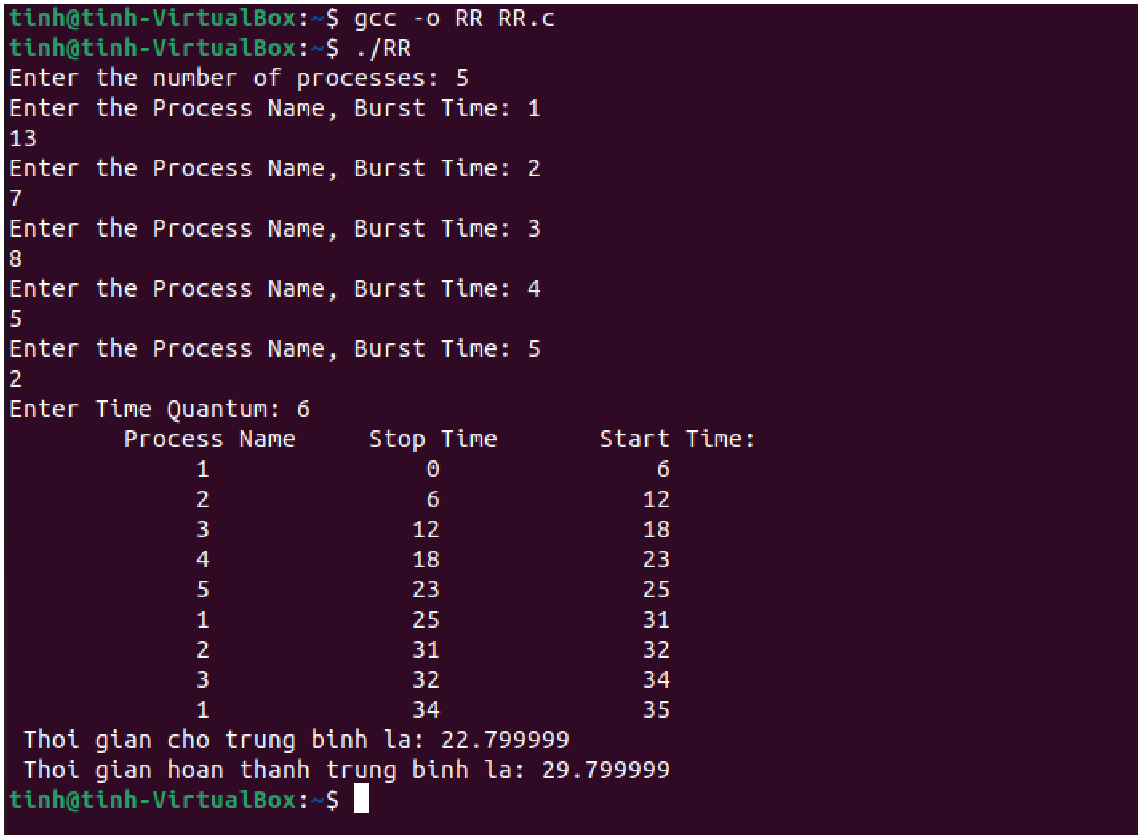


```c
                                }
                                if(bt_remaining[i] == minimum)
                                {
                                        if(ps[i].at < ps[min_index].at)
                                        {
                                                minimum = bt_remaining[i];
                                                min_index = i;
                                        }
                                }
                        }
                }
                if(min_index == -1) current_time++;
                else
                {
                        if(bt_remaining[min_index]==ps[min_index].bt)
                        {
                                ps[min_index].start_time = current_time;
                                is_first_process=false;
                        }
                        bt_remaining[min_index]-=1;
                        current_time++;
                        prev=current_time;
                        if(bt_remaining[min_index] == 0)
                        {
                        ps[min_index].ct = current_time;
                        ps[min_index].tat=ps[min_index].ct-ps[min_index].at;
                        ps[min_index].wt=ps[min_index].tat-ps[min_index].bt;
                        ps[min_index].rt=ps[min_index].start_time-ps[min_index].at;

                         sum_tat+= ps[min_index].tat;
                         sum_wt+= ps[min_index].wt;
                         sum_rt+= ps[min_index].rt;
                         completed++;
                         is_completed[min_index]=true;
                        }
                }
        }
        max_completion_time = INT_MIN;
        min_arrival_time = INT_MIN;
        for(int i=0; i<n; i++)
        {
                max_completion_time=findmax(max_completion_time,ps[i].ct);
                min_arrival_time=findmin(min_arrival_time, ps[i].at);
        }

        printf("\n Process    Arrival Time      Burst Time     TAT     Waiting Time     Respone Time\n");
        for(int i=0; i < n; i++)
                printf(" %d  \t\t %d         \t%d     \t\t%d     \t%d \t%d\n", ps[i].pid,ps[i].at,ps[i].bt,ps[i].tat,ps[i].wt, ps[i].rt);
        printf("\n");

        printf("\nAverage Turn Around time = %f",(float)sum_tat/n);
        printf("\nAverage Waiting time = %f",(float)sum_wt/n);
        printf("\nAverage Response time = %f",(float)sum_rt/n);
        printf("\n");
        return 0;
}
```

Câu 3: Viết chương trình mô phỏng giải thuật RR với các yêu cầu sau (giả sử tất cả các tiến trình đều có arrival time là 0):

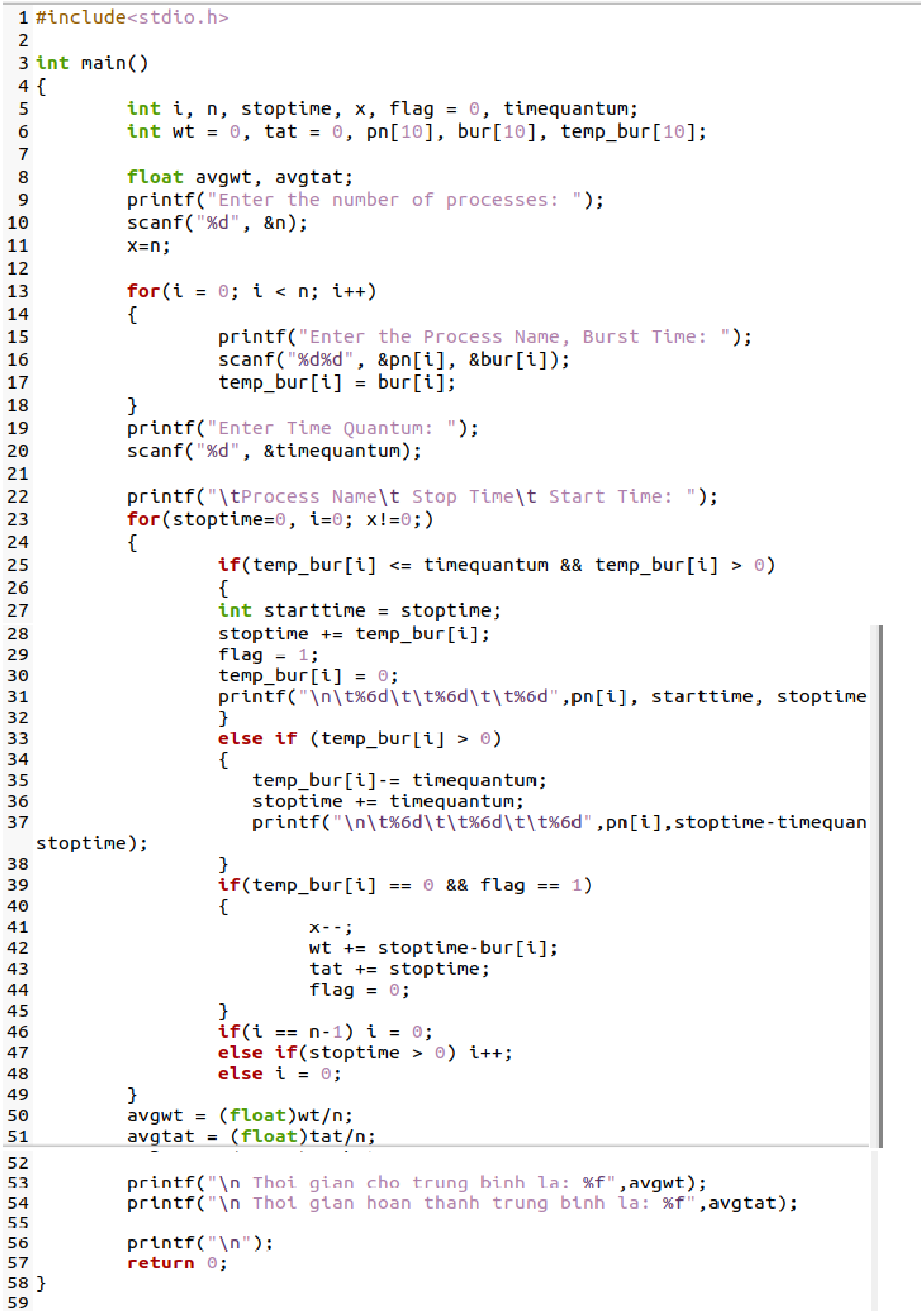
- **Nhập số process.**
- **Nhập quantum time.**
- **Nhập process name, burst time.**
- **In ra Gantt chart với các thông số: process name, start processor time, stop processor time.**
- **In ra average waiting time và average turnaround time.**

Kết quả chạy source code của giải thuật Round Robin:



```
tinh@tinh-VirtualBox:~$ gcc -o RR RR.c
tinh@tinh-VirtualBox:~$ ./RR
Enter the number of processes: 5
Enter the Process Name, Burst Time: 1
13
Enter the Process Name, Burst Time: 2
7
Enter the Process Name, Burst Time: 3
8
Enter the Process Name, Burst Time: 4
5
Enter the Process Name, Burst Time: 5
2
Enter Time Quantum: 6
        Process Name     Stop Time       Start Time:
             1               0               6
             2               6              12
             3              12              18
             4              18              23
             5              23              25
             1              25              31
             2              31              32
             3              32              34
             1              34              35
 Thoi gian cho trung binh la: 22.799999
 Thoi gian hoan thanh trung binh la: 29.799999
tinh@tinh-VirtualBox:~$
```

Source code RR với ngôn ngữ C:



```
#include<stdio.h>

int main()
{
        int i, n, stoptime, x, flag = 0, timequantum;
        int wt = 0, tat = 0, pn[10], bur[10], temp_bur[10];

        float avgwt, avgtat;
        printf("Enter the number of processes: ");
        scanf("%d", &n);
        x=n;

        for(i = 0; i < n; i++)
        {
                printf("Enter the Process Name, Burst Time: ");
                scanf("%d%d", &pn[i], &bur[i]);
                temp_bur[i] = bur[i];
        }
        printf("Enter Time Quantum: ");
        scanf("%d", &timequantum);

        printf("\tProcess Name\t Stop Time\t Start Time: ");
        for(stoptime=0, i=0; x!=0;)
        {
                if(temp_bur[i] <= timequantum && temp_bur[i] > 0)
                {
                int starttime = stoptime;
                stoptime += temp_bur[i];
                flag = 1;
                temp_bur[i] = 0;
                printf("\n\t%6d\t\t%6d\t\t%6d",pn[i], starttime, stoptime
                }
                else if (temp_bur[i] > 0)
                {
                   temp_bur[i]-= timequantum;
                   stoptime += timequantum;
                   printf("\n\t%6d\t\t%6d\t\t%6d",pn[i],stoptime-timequan
stoptime);
                }
                if(temp_bur[i] == 0 && flag == 1)
                {
                        x--;
                        wt += stoptime-bur[i];
                        tat += stoptime;
                        flag = 0;
                }
                if(i == n-1) i = 0;
                else if(stoptime > 0) i++;
                else i = 0;
        }
        avgwt = (float)wt/n;
        avgtat = (float)tat/n;

        printf("\n Thoi gian cho trung binh la: %f",avgwt);
        printf("\n Thoi gian hoan thanh trung binh la: %f",avgtat);

        printf("\n");
        return 0;
}

```