Name: Hà Danh Tuấn

ID: 20522109

Class: IT007.M13

OPERATING SYSTEM
LAB 4'S REPORT

SUMMARY

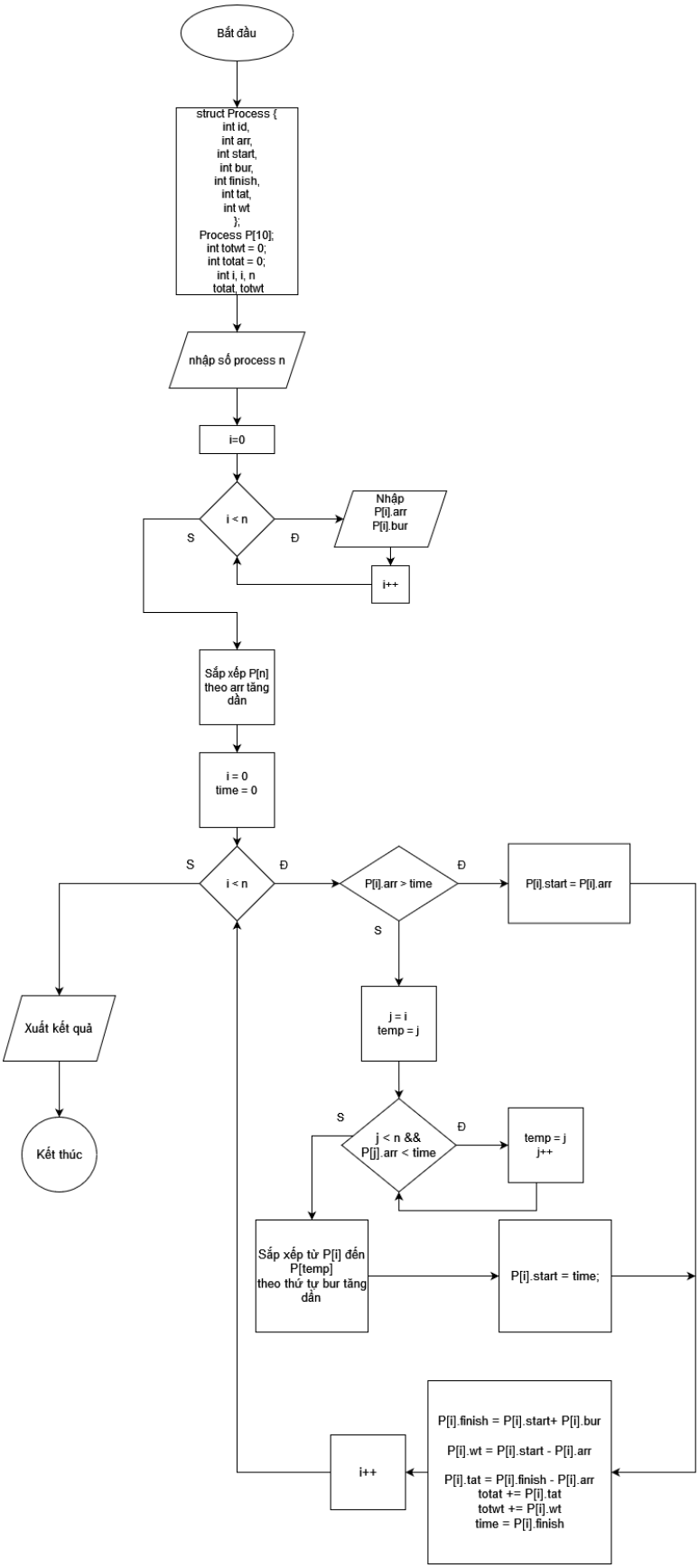
Self-scrores: 8

1. Vẽ lưu đồ, giải thích và hiện thực giải thuật SJF

Giải thích thuật toán:

Tạo kiểu dữ liệu struct Process gồm các thuộc tính id, arr, bur, start, finish, tat, wt
Sau khi nhập dữ liệu xong thì sắp xếp mảng các process theo thứ tự arr tăng dần.
Dùng biến time để làm mốc, process nào có thời gian tới(arr) lớn hơn time thì start của process đó sẽ bằng arr của process đó. Ngược lại thì sẽ tìm danh sách các tiến trình có arr bé hơn hoặc bằng mốc(time) rồi sẽ sắp xếp các tiến trình đó theo thứ tự bur tăng dần để chọn ra process có bur nhỏ nhất trong dãy vừa sắp xếp và thực hiện nó với start = time.
Những thuộc tính còn lại của process sẽ được tính ra sau khi tìm được start.

Lưu đồ thuật toán:


Bắt đầu
struct Process {
int id,
int arr,
int start,
int bur,
int finish,
int tat,
int wt
};
Process P[10];
int totwt = 0;
int totat = 0;
int i, i, n
totat, totwt
nhập số process n
i=0
i < n
S
Đ
Nhập
P[i].arr
P[i].bur
i++
Sắp xếp P[n]
theo arr tăng
dần
i = 0
time = 0
i < n
S
Đ
P[i].arr > time
Đ
P[i].start = P[i].arr
S
Xuất kết quả
Kết thúc
j = i
temp = j
j < n &&
P[j].arr < time
S
Đ
temp = j
j++
Sắp xếp từ P[i] đến
P[temp]
theo thứ tự bur tăng
dần
P[i].start = time;
P[i].finish = P[i].start+ P[i].bur
P[i].wt = P[i].start - P[i].arr
P[i].tat = P[i].finish - P[i].arr
totat += P[i].tat
totwt += P[i].wt
time = P[i].finish
i++

Hiện thực thuật toán

```c
#include<stdio.h>

struct Process {
        int id,  arr, bur, start, wt, tat, finish;
};

void swap(struct Process *a, struct Process *b)
{
        struct Process t;
        t = *a;
        *a = *b;
        *b = t;
}


int main ()
{
        struct Process P[10];
        int i, j, n, temp, time;
        int totat = 0, totwt = 0;
        float avgat, avgwt;
        printf("Enter the number of processes: ");
        scanf("%d", &n);
        for(i=0; i<n; i++)
        {
                printf("Enter the Process Name, Arrival Time & Burst Time: ");
                scanf("%d%d%d", &P[i].id, &P[i].arr, &P[i].bur);
        }

        for(i=0; i<n; i++)
        {
                for(j=i+1; j<n; j++)
                {
                        if(P[i].arr > P[j].arr)
                                swap(&P[i], &P[j]);
                        else if(P[i].arr == P[j].arr && P[i].bur > P[j].bur)
                                swap(&P[i], &P[j]);
                }
        }

        time=0;
        for(i=0; i<n; i++)
        {
                if(P[i].arr >= time)
                        P[i].start = P[i].arr;
                else
                {
```

```
                P[i].start = P[i].arr;
        else
        {
                j = i;
                temp = j;
                while(j < n && P[j].arr <= time)
                {
                        temp = j;
                        j++;
                }

                int k,l;
                for(k=i; k<temp; k++)
                {
                        for(l=k+1; l<=temp; l++)
                        {
                                if(P[k].bur > P[l].bur)
                                swap(&P[k], &P[l]);
                        }
                }

                P[i].start = time;
        }

        P[i].finish = P[i].start + P[i].bur;
        P[i].wt = P[i].start - P[i].arr;
        P[i].tat = P[i].finish - P[i].arr;
        totat += P[i].tat;
        totwt += P[i].wt;
        time = P[i].finish;
}



printf("\nPname \t Arrtime \t Burtime Start \t TAT \tWaiting \tFinish");
for(i=0; i<n; i++)
{
        printf("\n%d\t%6d\t\t%6d\t%6d\t%6d\t%6d", P[i].id, P[i].arr, P[i].bur, P[i].start, P[i].tat, P[i].wt, P[i].finish);
}

avgat = (float)totat/n;
avgwt = (float)totwt/n;
printf("\nAverage Turnaround Time: ");
printf("%f", avgat);
printf("\nAverage Waiting Time: ");
printf("%f",  avgwt);
```

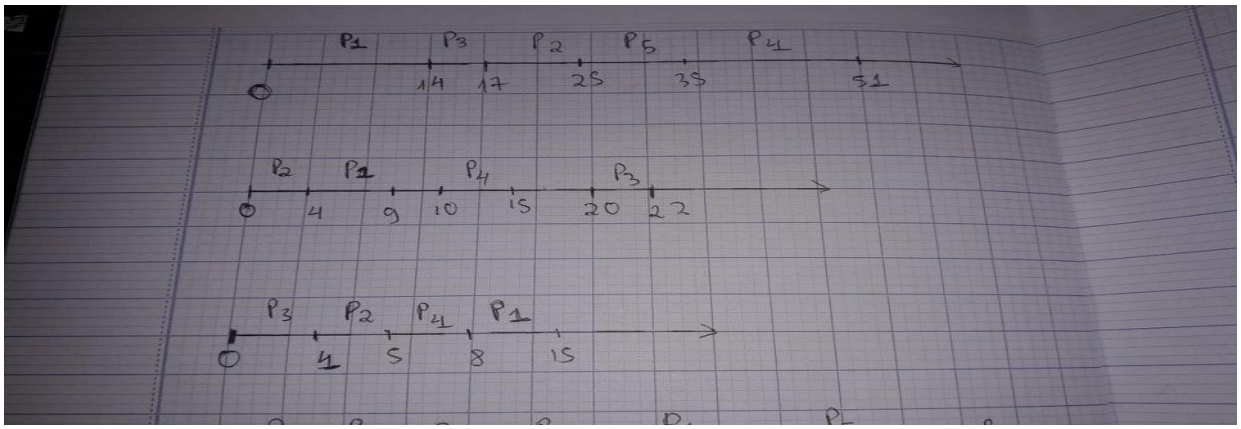
Test case:

```
tuan_20522109@tuan-VirtualBox:~/Desktop$ gcc sjf.c -o sjf
tuan_20522109@tuan-VirtualBox:~/Desktop$ ./sjf
Enter the number of processes: 5
Enter the Process Name, Arrival Time & Burst Time: 1 0 14
Enter the Process Name, Arrival Time & Burst Time: 2 5 8
Enter the Process Name, Arrival Time & Burst Time: 3 7 3
Enter the Process Name, Arrival Time & Burst Time: 4 7 16
Enter the Process Name, Arrival Time & Burst Time: 5 12 10

Pname    Arrtime         Burtime Start   TAT    Waiting         Finish
1            0              14       0      14       0       14
3            7               3      14      10       7       17
2            5               8      17      20      12       25
5           12              10      25      23      13       35
4            7              16      35      44      28       51
Average Turnaround Time: 22.200001
Average Waiting Time: 12.000000tuan_20522109@tuan-VirtualBox:~/Desktop$
tuan_20522109@tuan-VirtualBox:~/Desktop$ ./sjf
Enter the number of processes: 4
Enter the Process Name, Arrival Time & Burst Time: 1 3 5
Enter the Process Name, Arrival Time & Burst Time: 2 0 4
Enter the Process Name, Arrival Time & Burst Time: 3 20 2
Enter the Process Name, Arrival Time & Burst Time: 4 10 5

Pname    Arrtime         Burtime Start   TAT    Waiting         Finish
2            0               4       0       4       0        4
1            3               5       4       6       1        9
4           10               5      10       5       0       15
3           20               2      20       2       0       22
Average Turnaround Time: 4.250000
Average Waiting Time: 0.250000tuan_20522109@tuan-VirtualBox:~/Desktop$
tuan_20522109@tuan-VirtualBox:~/Desktop$ ./sjf
Enter the number of processes: 4
Enter the Process Name, Arrival Time & Burst Time: 1 4 7
Enter the Process Name, Arrival Time & Burst Time: 2 2 1
Enter the Process Name, Arrival Time & Burst Time: 3 0 4
Enter the Process Name, Arrival Time & Burst Time: 4 5 3

Pname    Arrtime         Burtime Start   TAT    Waiting         Finish
3            0               4       0       4       0        4
2            2               1       4       3       2        5
4            5               3       5       3       0        8
1            4               7       8      11       4       15
Average Turnaround Time: 5.250000
Average Waiting Time: 1.500000tuan_20522109@tuan-VirtualBox:~/Desktop$
```



2. Vẽ lưu đồ, giải thích và hiện thực giải thuật SRT

Giải thích thuật toán:

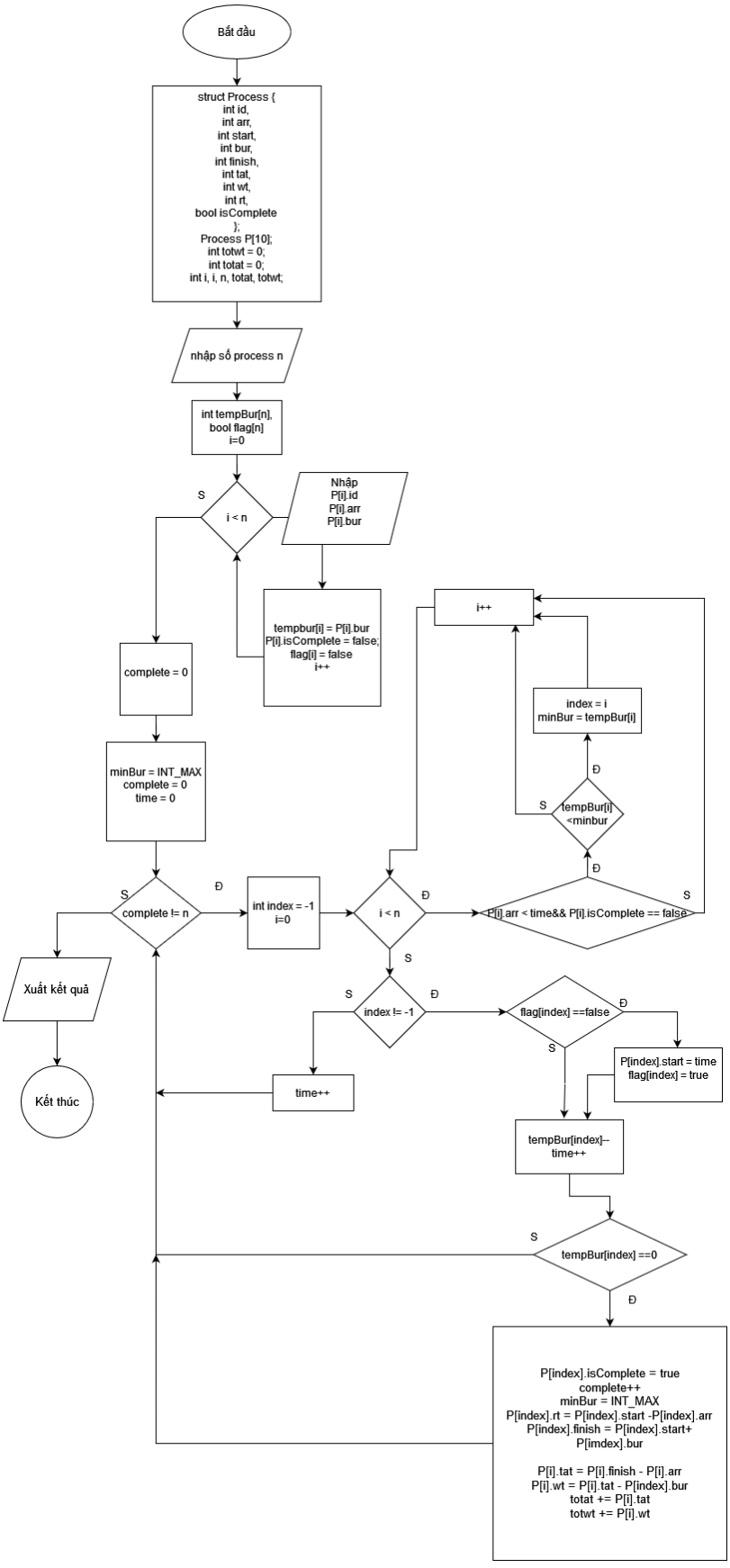
Tạo kiểu dữ liệu struct Process gồm các thuộc tính id, arr, bur, start, finish, tat, wt, rt và có thêm thuộc tính kiểu bool isComplete với mục đích làm cờ hiệu khi tiến trình đã thực hiện xong.

Tạo mảng phụ tempBur[n] để thực hiện trừ dần bur của từng tiến trình, mảng flag[n] để làm cờ hiệu xem tiến trình vừa được bắt đầu chạy hay không.

Biến complete sẽ đếm số tiến trình được hoàn thành.

Dùng biến time để đo đơn vị thời gian hiện tại, với mỗi lần lặp thì biến time sẽ tăng thêm 1 giá trị, dò trong số các tiến trình có thời gian đến bé hơn hoặc bằng time và cờ hiệu isComplete chưa được bật để tìm ra tiến trình có bur nhỏ nhất. Dùng mảng flag[n] để xem tiến trình đó có phải vừa được đưa ra khỏi hàng chờ và chạy lần đầu hay không, nếu có thì gán start = time, bật cờ hiệu flag[i] lên, nếu không thì giảm giá trị bur của tiến trình đó xuống 1 và tăng giá trị biến time lên 1, nếu bur của tiến trình đó bằng 0 thì có nghĩa tiến trình đó đã được thực hiện xong, tăng giá trị complete lên 1 và bật cờ isComplete của tiến trình đó lên rồi tiếp tục thuật toán.

Lưu đồ thuật toán:


Bắt đầu
struct Process {
int id,
int arr,
int start,
int bur,
int finish,
int tat,
int wt,
int rt,
bool isComplete
};
Process P[10];
int totwt = 0;
int totat = 0;
int i, n, totat, totwt;
nhập số process n
int tempBur[n],
bool flag[n]
i=0
S
i < n
Nhập
P[i].id
P[i].arr
P[i].bur
tempbur[i] = P[i].bur
P[i].isComplete = false;
flag[i] = false
i++
complete = 0
minBur = INT_MAX
complete = 0
time = 0
S
complete != n
Đ
int index = -1
i=0
Xuất kết quả
Kết thúc
Đ
i < n
S
i++
S
P[i].arr < time&& P[i].isComplete == false
Đ
index = i
minBur = tempBur[i]
Đ
S
tempBur[i]
<minbur
Đ
S
index != -1
Đ
time++
Đ
flag[index] ==false
S
P[index].start = time
flag[index] = true
tempBur[index]--
time++
S
tempBur[index] ==0
Đ
P[index].isComplete = true
complete++
minBur = INT_MAX
P[index].rt = P[index].start -P[index].arr
P[index].finish = P[index].start+
P[imdex].bur
P[i].tat = P[i].finish - P[i].arr
P[i].wt = P[i].tat - P[index].bur
totat += P[i].tat
totwt += P[i].wt

Hiện thực thuật toán:

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

struct Process {
        int id,  arr, bur, start, wt, tat, finish, rt;
        bool isComplete;
};


int main ()
{
        struct Process P[10];
        int totat = 0, totwt = 0, i, j, n;
        int complete = 0, time = 0, minBur = INT_MAX;
        float avgwt, avgat;

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

        int tempBur[n];
        bool flag[n];

        for(i=0; i<n; i++)
        {
                printf("Enter the Process Name, Arrival Time & Burst Time: ");
                scanf("%d%d%d", &P[i].id, &P[i].arr, &P[i].bur);
                tempBur[i]=P[i].bur;
        }

        for(i=0; i<n; i++)
        {
                P[i].isComplete = false;
                flag[i] = false;
        }

        while(complete != n )
        {
                int index = -1;
                for(i=0; i<n; i++)
                {
                        if(P[i].arr <= time && P[i].isComplete == false)
                        {
                                if(tempBur[i] < minBur)
                                {
                                        index = i;
                                        minBur = tempBur[i];
```

```
                                index = i;
                                minBur = tempBur[i];
                        }
                }
        }
        if(index != -1)
        {
                if(flag[index]==false)
                {
                        P[index].start = time;
                        flag[index] = true;
                }


                tempBur[index]--;
                time++;
                if(tempBur[index] == 0)
                {
                        P[index].isComplete = true;
                        complete++;
                        minBur = INT_MAX;

                        P[index].rt = P[index].start - P[index].arr;
                        P[index].finish = time;
                        P[index].tat = P[index].finish - P[index].arr;
                        P[index].wt = P[index].tat - P[index].bur;
                        totat += P[index].tat;
                        totwt += P[index].wt;
                }
        }
        else {
                time++; }

}

printf("\nPname \tArrtime \tBurtime  Start \tTAT \tWaiting   Responsive Finish");
for(i=0; i<n; i++)
{
        printf("\n%d\t%6d\t\t%6d\t%6d\t%6d\t%6d\t%6d\t%6d", P[i].id, P[i].arr, P[i].bur, P[i].start, P[i].tat, P[i].wt, P[i].rt, P[i].finish);
}

avgat = (float)totat/n;
avgwt = (float)totwt/n;
printf("\nAverage Turnaround Time: ");
printf("%f", avgat);
printf("\nAverage Waiting Time: ");
printf("%f",  avgwt);
```

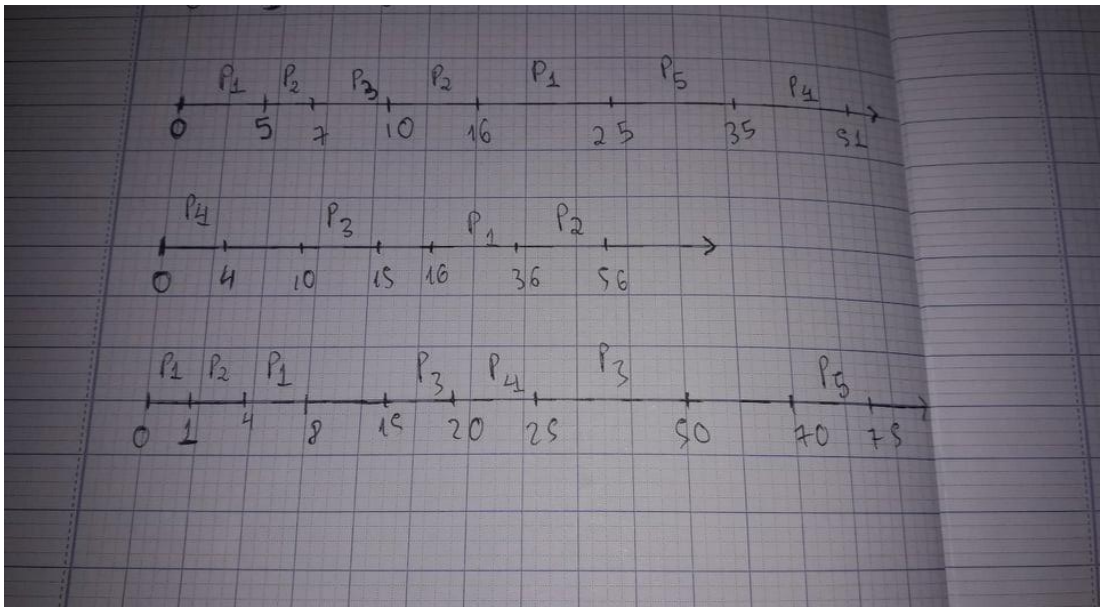
Test case:

```
tuan_20522109@tuan-VirtualBox:~/Desktop$ ./srt
Enter the number of processes: 5
Enter the Process Name, Arrival Time & Burst Time: 1 0 14
Enter the Process Name, Arrival Time & Burst Time: 2 5 8
Enter the Process Name, Arrival Time & Burst Time: 3 7 3
Enter the Process Name, Arrival Time & Burst Time: 4 7 16
Enter the Process Name, Arrival Time & Burst Time: 5 12 10

Pname   Arrtime         Burtime  Start  TAT     Waiting   Responsive Finish
1            0              14       0      25       11       0      25
2            5               8       5      11        3       0      16
3            7               3       7       3        0       0      10
4            7              16      35      44       28      28      51
5           12              10      25      23       13      13      35
Average Turnaround Time: 21.200001
Average Waiting Time: 11.000000tuan_20522109@tuan-VirtualBox:~/Desktop$
tuan_20522109@tuan-VirtualBox:~/Desktop$
tuan_20522109@tuan-VirtualBox:~/Desktop$ ./srt
Enter the number of processes: 4
Enter the Process Name, Arrival Time & Burst Time: 0 16 20
Enter the Process Name, Arrival Time & Burst Time: 2 23 20
Enter the Process Name, Arrival Time & Burst Time: 3 10 5
Enter the Process Name, Arrival Time & Burst Time: 4 0 4

Pname   Arrtime         Burtime  Start  TAT     Waiting   Responsive Finish
0           16              20      16      20        0       0      36
2           23              20      36      33       13      13      56
3           10               5      10       5        0       0      15
4            0               4       0       4        0       0       4
Average Turnaround Time: 15.500000
Average Waiting Time: 3.250000tuan_20522109@tuan-VirtualBox:~/Desktop$
tuan_20522109@tuan-VirtualBox:~/Desktop$
tuan_20522109@tuan-VirtualBox:~/Desktop$ ./srt
Enter the number of processes: 5
Enter the Process Name, Arrival Time & Burst Time: 1 0 5
Enter the Process Name, Arrival Time & Burst Time: 2 1 3
Enter the Process Name, Arrival Time & Burst Time: 3 15 30
Enter the Process Name, Arrival Time & Burst Time: 4 20 5
Enter the Process Name, Arrival Time & Burst Time: 5 70 5

Pname   Arrtime         Burtime  Start  TAT     Waiting   Responsive Finish
1            0               5       0       8        3       0       8
2            1               3       1       3        0       0       4
3           15              30      15      35        5       0      50
4           20               5      20       5        0       0      25
5           70               5      70       5        0       0      75
Average Turnaround Time: 11.200000
```


P1 P2 P3 P2 P1 P5 P4
0 5 7 10 16 25 35 51
P4 P3 P1 P2
0 4 10 15 16 36 56
P1 P2 P1 P3 P4 P3 P5
0 1 4 8 15 20 25 50 70 75