Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

Họ và tên:

Phạm Đông Hưng

Lương Anh Huy

Phan Công Minh

Hồng Khải Nguyên

MSSV:

22520521

22520550

22520884

22520967

Lớp: IT007.O14.1

HỆ ĐIỀU HÀNH
BÁO CÁO LAB 4

CHECKLIST

3.5. BÀI TẬP THỰC HÀNH

	BT 1	BT 2
Vẽ lưu đồ giải thuật	☒	☒
Chạy tay lưu đồ giải thuật	☒	☒
Hiện thực code	☒	☒
Chạy code và kiểm chứng	☒	☒

3.6. BÀI TẬP ÔN TẬP

	BT 1
Vẽ lưu đồ giải thuật	☒

Chạy tay lưu đồ giải thuật	☒
Hiện thực code	☒
Chạy code và kiểm chứng	☒

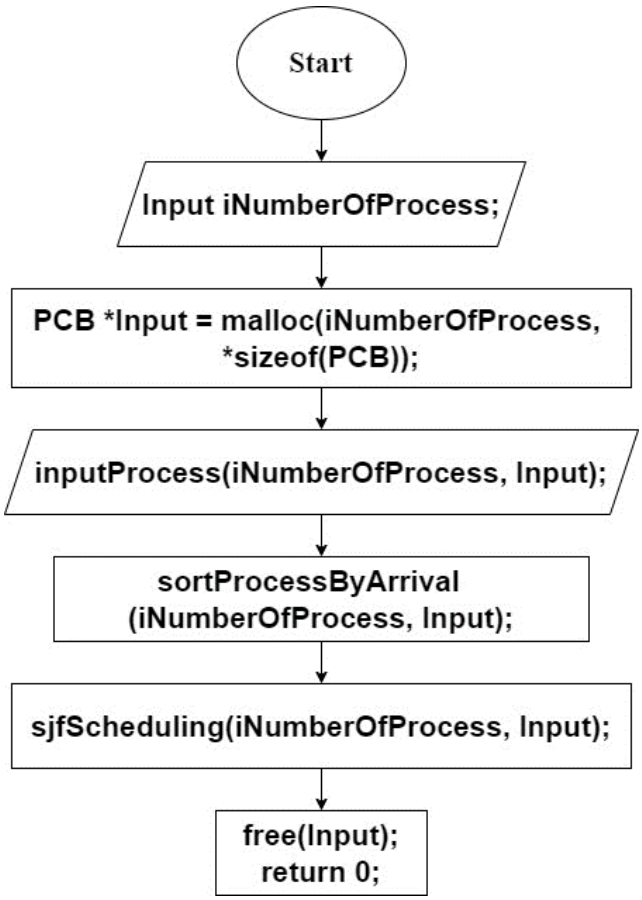
Tự chấm điểm: 9.5

**Lưu ý: Xuất báo cáo theo định dạng PDF, đặt tên theo cú pháp:*
<Tên nhóm>_LAB3.pdf

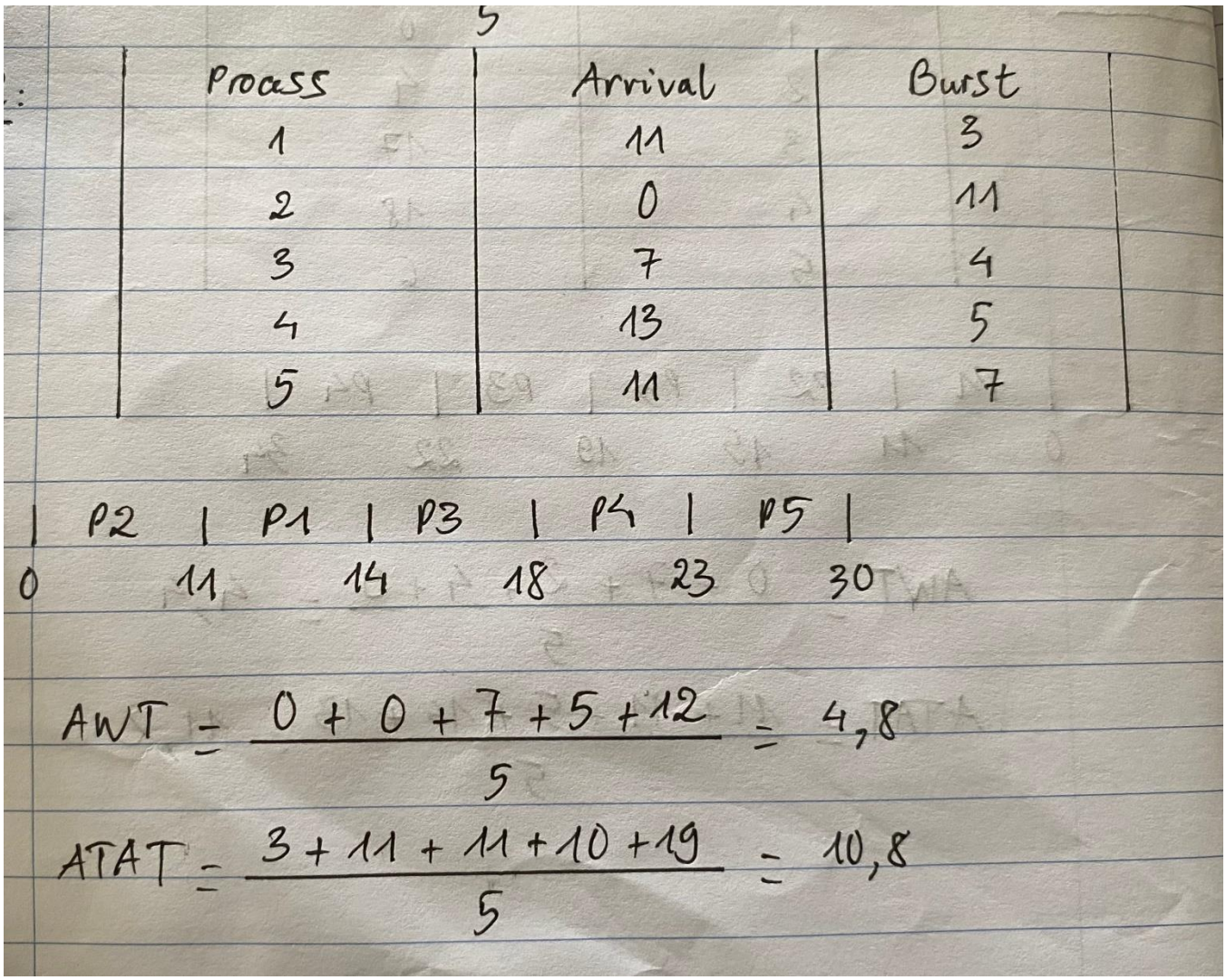
2.5. BÀI TẬP THỰC HÀNH

1. Giải thuật Shortest-Job-First

Vẽ lưu đồ giải thuật:



Chạy tay lưu đồ giải thuật:



Process	Arrival	Burst
1	11	3
2	0	11
3	7	4
4	13	5
5	11	7

P2	P1	P3	P4	P5
0	11	14	18	23	30

$$AWT = \frac{0 + 0 + 7 + 5 + 12}{5} = 4,8$$

$$ATAT = \frac{3 + 11 + 11 + 10 + 19}{5} = 10,8$$

Hiện thực code:

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

typedef struct {
    int iPID;
    int iArrival, iBurst;
    int iStart, iFinish, iWaiting, iResponse, iTaT;
} PCB;

void inputProcess(int n, PCB P[]) {
    srand(time(NULL)); // Khởi tạo seed cho hàm rand() từ thời gian hiện tại
    bool flag = false;
    for (int i = 0; i < n; i++) {
        printf("Input the PID: ");
        scanf("%d", &P[i].iPID);
        P[i].iArrival = rand() % 21; // Tạo giá trị ngẫu nhiên trong khoảng [0, 20]
        P[i].iBurst = rand() % 11 + 2; // Tạo giá trị ngẫu nhiên trong khoảng [2, 12]
        if (P[i].iArrival == 0) {
            flag = true;
        }
    }
    if (!flag) {
        int randomIndex = rand() % n;
        P[randomIndex].iArrival = 0;
    }
}

void swapProcess(PCB *P, PCB *Q) {
    PCB t = *P;
    *P = *Q;
    *Q = t;
}

void sortProcessesByArrival(int n, PCB P[]) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (P[j].iArrival > P[j + 1].iArrival) {
                swapProcess(&P[j], &P[j + 1]);
            }
        }
    }
}
```

```c
void sjfScheduling(int n, PCB P[]) {
    int currentTime = 0;
    int completedProcesses = 0;
    int minBurstIndex;
    float totalWaitingTime = 0;
    float totalTurnaroundTime = 0;

    while (completedProcesses < n) {
        minBurstIndex = -1;
        int minBurstTime = 100000000;

        for (int i = 0; i < n; i++) {
            if (P[i].iArrival <= currentTime && P[i].iBurst < minBurstTime && P[i].iFinish == 0) {
                minBurstTime = P[i].iBurst;
                minBurstIndex = i;
            }
        }

        if (minBurstIndex == -1) {
            currentTime++;
        } else {
            PCB *currentProcess = &P[minBurstIndex];
            currentProcess->iStart = currentTime;
            currentProcess->iFinish = currentTime + currentProcess->iBurst;
            currentProcess->iResponse = currentProcess->iStart - currentProcess->iArrival;
            currentProcess->iWaiting = currentProcess->iStart - currentProcess->iArrival;
            currentProcess->iTaT = currentProcess->iFinish - currentProcess->iArrival;

            totalWaitingTime += currentProcess->iWaiting;
            totalTurnaroundTime += currentProcess->iTaT;

            currentTime = currentProcess->iFinish;
            completedProcesses++;
        }
    }

    float averageWaitingTime = totalWaitingTime / n;
    float averageTurnaroundTime = totalTurnaroundTime / n;

    printf("SJF Scheduling:\n");
    printf("Process\tArrival\tBurst\tStart\tFinish\tWaiting\tResponse\tTurnaround\n");
    for (int i = 0; i < n; i++) {
        PCB currentProcess = P[i];
        printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\n", currentProcess.iPID, currentProcess.iArrival,
               currentProcess.iBurst, currentProcess.iStart, currentProcess.iFinish, currentProcess.iWaiting,
               currentProcess.iResponse, currentProcess.iTaT);
    }

    printf("Average Waiting Time: %.2f\n", averageWaitingTime);
    printf("Average Turnaround Time: %.2f\n", averageTurnaroundTime);
}
```

```c
int main() {
    int iNumberOfProcess;
    printf("Please input the number of processes: ");
    scanf("%d", &iNumberOfProcess);

    PCB *Input = malloc(iNumberOfProcess * sizeof(PCB));
    inputProcess(iNumberOfProcess, Input);
    sortProcessesByArrival(iNumberOfProcess, Input);
    sjfScheduling(iNumberOfProcess, Input);

    free(Input);

    return 0;
}
```

Chạy code và kiểm chứng:

Test case 1:

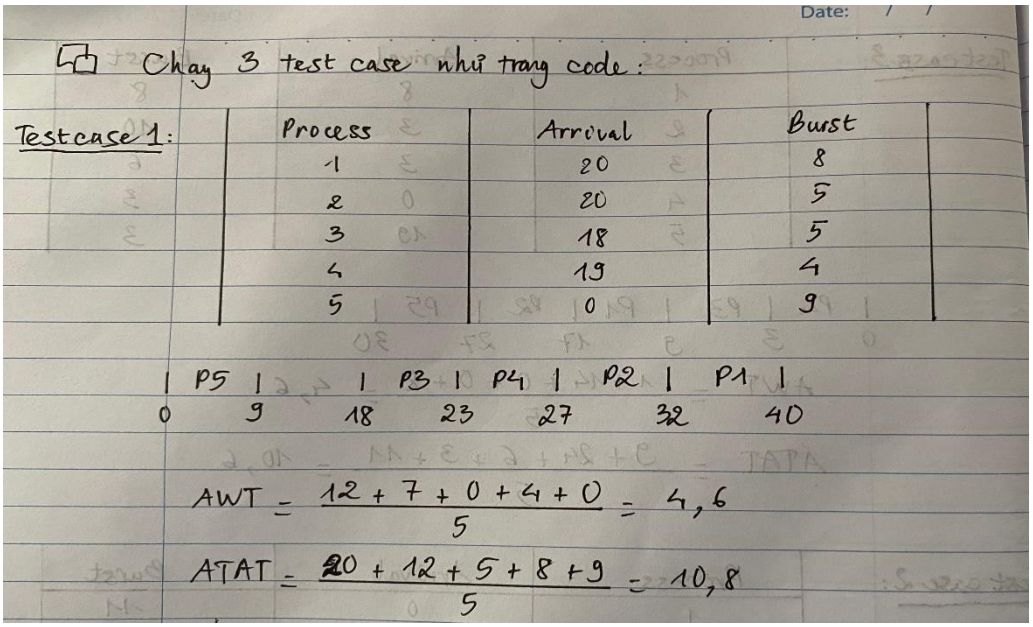
```
Please input the number of processes: 5
Input the PID: 1
Input the PID: 2
Input the PID: 3
Input the PID: 4
Input the PID: 5
SJF Scheduling:
Process Arrival Burst   Start   Finish  Waiting Response        Turnaround
5       0       9       0       9       0       0               9
3       18      5       18      23      0       0               5
4       19      4       23      27      4       4               8
1       20      8       32      40      12      12              20
2       20      5       27      32      7       7               12
Average Waiting Time: 4.60
Average Turnaround Time: 10.80
```

Test case 2:

```
Please input the number of processes: 5
Input the PID: 1
Input the PID: 2
Input the PID: 3
Input the PID: 4
Input the PID: 5
SJF Scheduling:
Process Arrival Burst   Start   Finish  Waiting Response        Turnaround
1       0       11      0       11      0       0               11
2       4       4       11      15      7       7               11
5       6       4       15      19      9       9               13
3       17      3       19      22      2       2               5
4       18      12      22      34      4       4               16
Average Waiting Time: 4.40
Average Turnaround Time: 11.20
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine
```

Test case 3:

```
Please input the number of processes: 5
Input the PID: 1
Input the PID: 2
Input the PID: 3
Input the PID: 4
Input the PID: 5
SJF Scheduling:
Process Arrival Burst   Start   Finish  Waiting Response        Turnaround
4       0       3       0       3       0       0               3
2       3       10      17      27      14      14              24
3       3       6       3       9       0       0               6
1       8       8       9       17      1       1               9
5       19      3       27      30      8       8               11
Average Waiting Time: 4.60
Average Turnaround Time: 10.60
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-b1dmduac.q3p" 1>"/tmp/Mi
```



Test case 2:

Process	Arrival	Burst
1	0	11
2	4	4
3	17	3
4	18	12
5	6	4

| P1 | P2 | P5 | P3 | P4 |
0 11 15 19 22 34

$$AWT = \frac{0 + 7 + 2 + 4 + 9}{5} = 4,4$$

$$ATAT = \frac{11 + 11 + 5 + 16 + 13}{5} = 11,2$$

Test case 3:

Process	Arrival	Burst
1	8	8
2	3	10
3	3	6
4	0	3
5	19	3

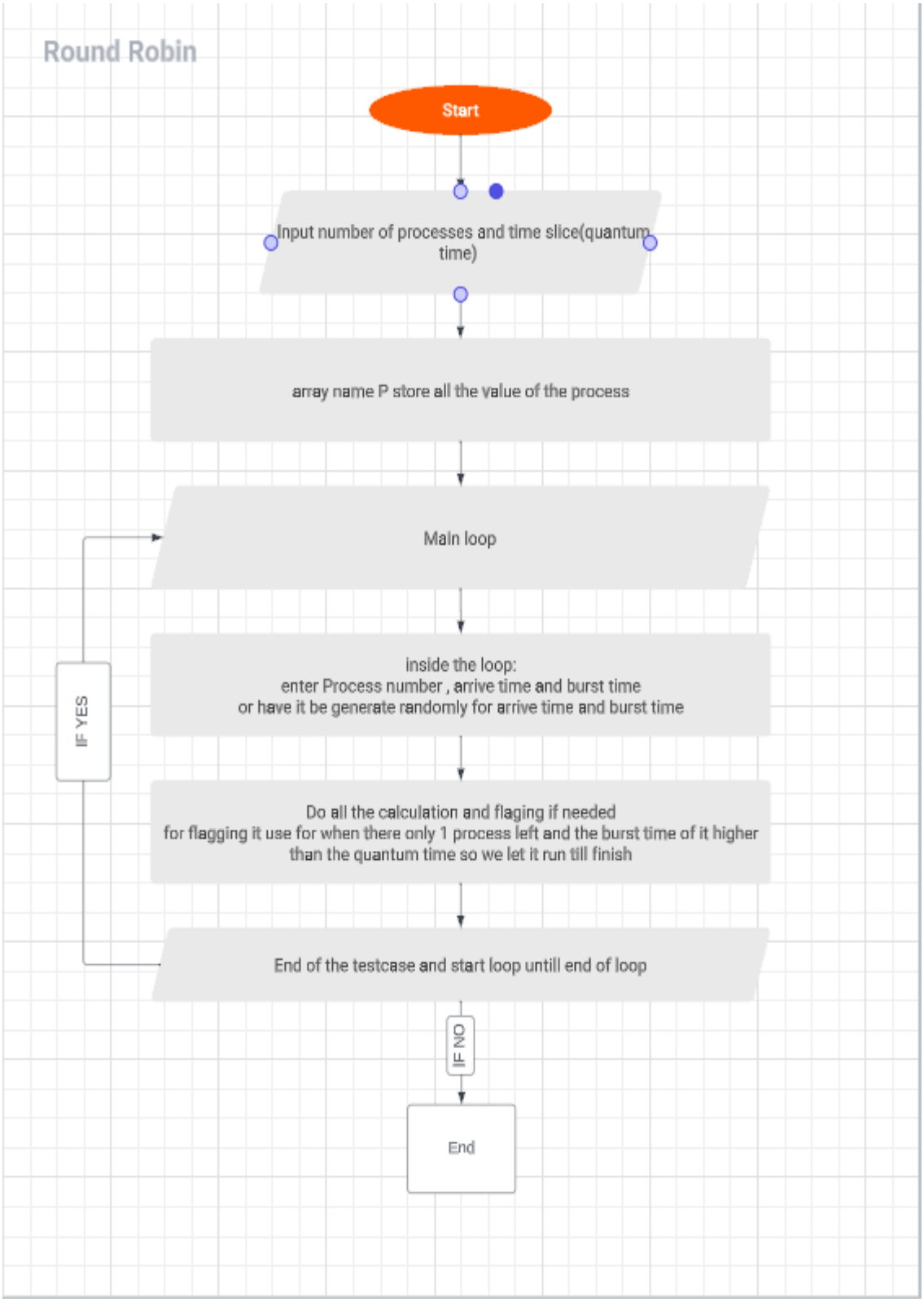
| P4 | P3 | P1 | P2 | P5 |
0 3 9 17 27 30

$$AWT = \frac{1 + 14 + 0 + 0 + 8}{5} = 4,6$$

$$ATAT = \frac{9 + 24 + 6 + 3 + 11}{5} = 10,6$$

2. Giải thuật Shortest-Remaining-Time-First hoặc Round Robin

Flowchart:



Phần code:

FCFS.c RR.c

RR.c > main()

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

struct P{
int AT,BT,ST[20],WT,FT,TAT,pos;
};

int quant;

void manualInput(struct P p[], int n)
{
    printf("Enter the process numbers:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &(p[i].pos));

    printf("Enter the Arrival time of processes:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &(p[i].AT));

    printf("Enter the Burst time of processes:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &(p[i].BT));
}

void randomInput(struct P p[], int n)
{
    for (int i = 0; i < n; i++)
    {
        p[i].pos = i + 1;
        p[i].AT = rand() % 21;
        p[i].BT = (rand() % 11) + 2;
    }
}

int main()
{
    int n, i, j;
    printf("Enter the number of processes :");
    scanf("%d", &n);
    struct P p[n];

    printf("Enter the quantum :\n");
    scanf("%d", &quant);

    for (int c = 0; c < 4; c++)
    {
        printf("Test Case %d:\n", c + 1);

        if (c == 3)
        {
            manualInput(p, n);
        }
        else
        {
            randomInput(p, n);
        }
        int c = n, s[n][20];
        float time = 0, mini = INT_MAX, b[n], a[n];

        int index = -1;
```

FCFS.c RR.c 4

RR.c > main()

```
        int index = -1;
        for (i = 0; i < n; i++)
        {
            b[i] = p[i].BT;
            a[i] = p[i].AT;
            for (j = 0; j < 20; j++)
            {
                s[i][j] = -1;
            }
        }

        int tot_wt, tot_tat;
        tot_wt = 0;
        tot_tat = 0;
        bool flag = false;

        while (c != 0)
        {

            mini = INT_MAX;
            flag = false;

            for (i = 0; i < n; i++)
            {
                float p = time + 0.1;
                if (a[i] <= p && mini > a[i] && b[i] > 0)
                {
                    index = i;
                    mini = a[i];
                    flag = true;
                }
            }
            if (!flag)
            {
                time++;
                continue;
            }

            j = 0;

            while (s[index][j] != -1)
            {
                j++;
            }

            if (s[index][j] == -1)
            {
                s[index][j] = time;
                p[index].ST[j] = time;
            }

            if (b[index] <= quant)
            {
                time += b[index];
                b[index] = 0;
            }
            else
            {
                time += quant;
                b[index] -= quant;
            }
```

FCFS.c RR.c

RR.c > main()

```c
            if (b[index] <= quant)
            {
                time += b[index];
                b[index] = 0;
            }
            else
            {
                time += quant;
                b[index] -= quant;
            }

            if (b[index] > 0)
            {
                a[index] = time + 0.1;
            }

            if (b[index] == 0)
            {
                c--;
                p[index].FT = time;
                p[index].WT = p[index].FT - p[index].AT - p[index].BT;
                tot_wt += p[index].WT;
                p[index].TAT = p[index].BT + p[index].WT;
                tot_tat += p[index].TAT;
            }
        }

        // Move the header printing outside the loop
        printf("Process number ");
        printf("Arrival time ");
        printf("Burst time ");
        printf(" Wait Time ");
        printf(" TurnAround Time \n");

        for (i = 0; i < n; i++)
        {
            printf("%d \t\t", p[i].pos);
            printf("%d \t\t", p[i].AT);
            printf("%d \t", p[i].BT);
            j = 0;
            int v = 0;
            while (v != 40)
            {
                v += 1;
            }
            printf("%d \t", p[i].WT);
            printf("%d \t\t\n", p[i].TAT);
        }

        double avg_wt, avg_tat;
        avg_wt = tot_wt / (float)n;
        avg_tat = tot_tat / (float)n;

        printf("The average wait time is : %lf\n", avg_wt);
        printf("The average TurnAround time is : %lf\n", avg_tat);
    }

    return 0;
}
```

Phần chạy testcase:

Bằng tay:

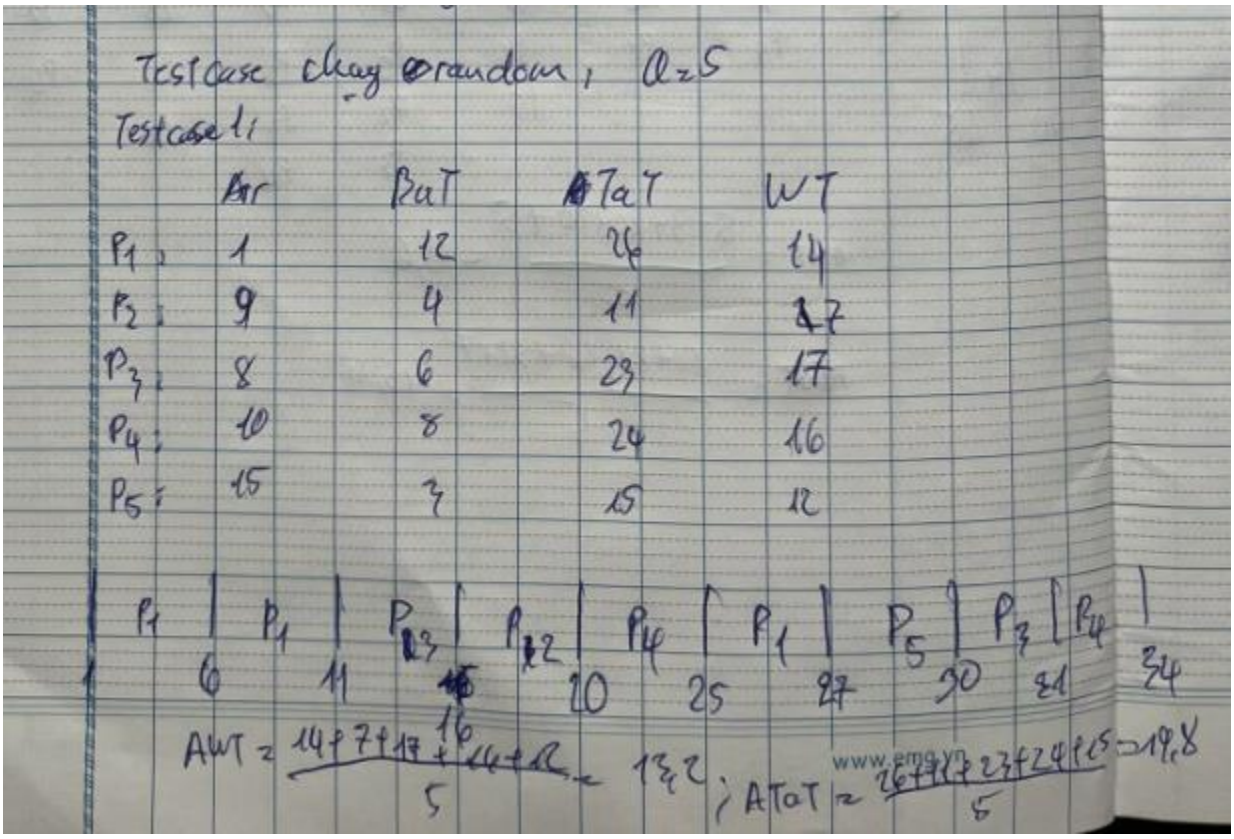
Kiểm chứng:

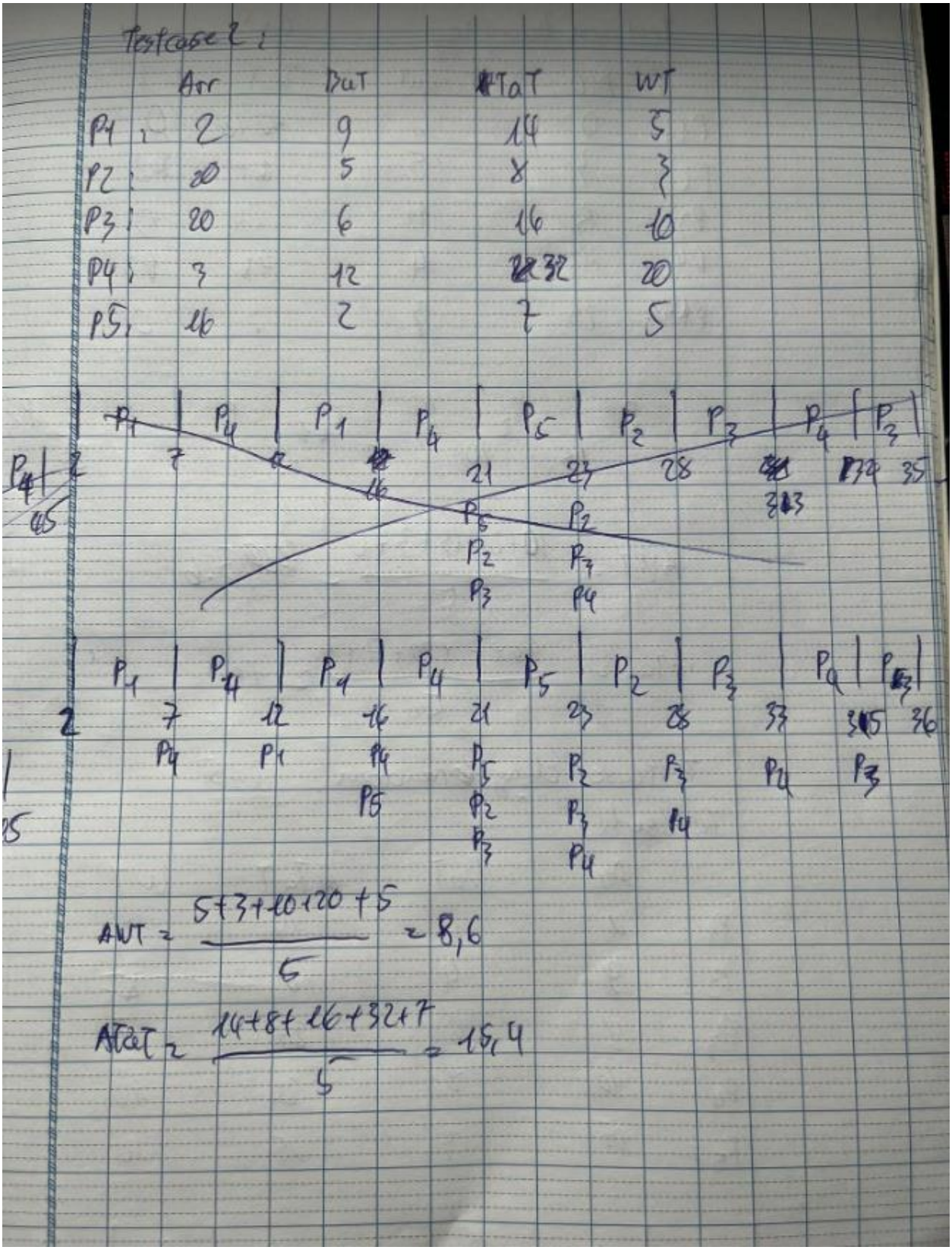
```
Enter the process numbers:
1 2 3 4 5
Enter the Arrival time of processes:
0 3 5 13 25
Enter the Burst time of processes:
6 5 11 4 3
Process number Arrival time Burst time  Wait Time  TurnAround Time
1              0            6          10         16
2              3            5          2          7
3              5            11         13         24
4              13           4          3          7
5              25           3          0          3
The average wait time is : 5.600000
The average TurnAround time is : 11.400000
```

Chạy bằng random 3 test case:

```
Enter the number of processes :5
Enter the quantum :
5
Test Case 1:
Process number Arrival time Burst time  Wait Time  TurnAround Time
1              1            12      14      26
2              9            4       7       11
3              8            6       17      23
4              10           8       16      24
5              15           3       12      15
The average wait time is : 13.200000
The average TurnAround time is : 19.799999
Test Case 2:
Process number Arrival time Burst time  Wait Time  TurnAround Time
1              2            9       5       14
2              20           5       3       8
3              20           6       10      16
4              3            12      20      32
5              16           2       5       7
The average wait time is : 8.600000
The average TurnAround time is : 15.400000
Test Case 3:
Process number Arrival time Burst time  Wait Time  TurnAround Time
1              17           10      18      28
2              12           2       12      14
3              2            8       14      22
4              5            12      23      35
5              1            12      25      37
The average wait time is : 18.400000
The average TurnAround time is : 27.200001
```

Kiểm chứng:


Testcase chạy random, Q=5
Testcase 1:
Ar
BuT
ToT
WT
P1: 1 12 26 14
P2: 9 4 11 7
P3: 8 6 23 17
P4: 10 8 24 16
P5: 15 3 15 12
P1 P1 P3 P2 P4 P1 P5 P3 P4
6 11 16 20 25 27 30 31 34
AWT = 14+7+17+16+12 / 5 = 13,2 ; ATaT = 26+11+23+24+15 / 5 = 19,8
www.emgvn

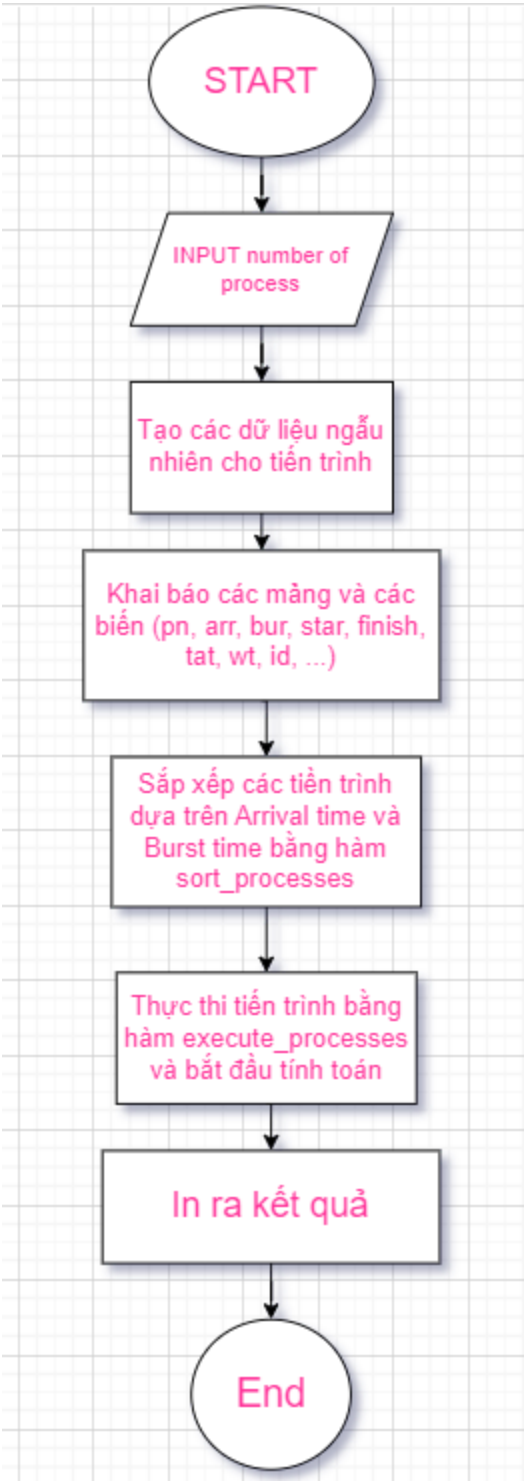

Testcase 2:
Arr
BuT
WT
P4: 2 9 5
P2: 20 5 3
P3: 20 6 10
P4: 3 12 20
P5: 16 2 5
AWT = (5+3+10+20+5)/5 = 8,6
ATaT = (14+8+16+32+7)/5 = 15,4

2.6. BÀI TẬP ÔN TẬP

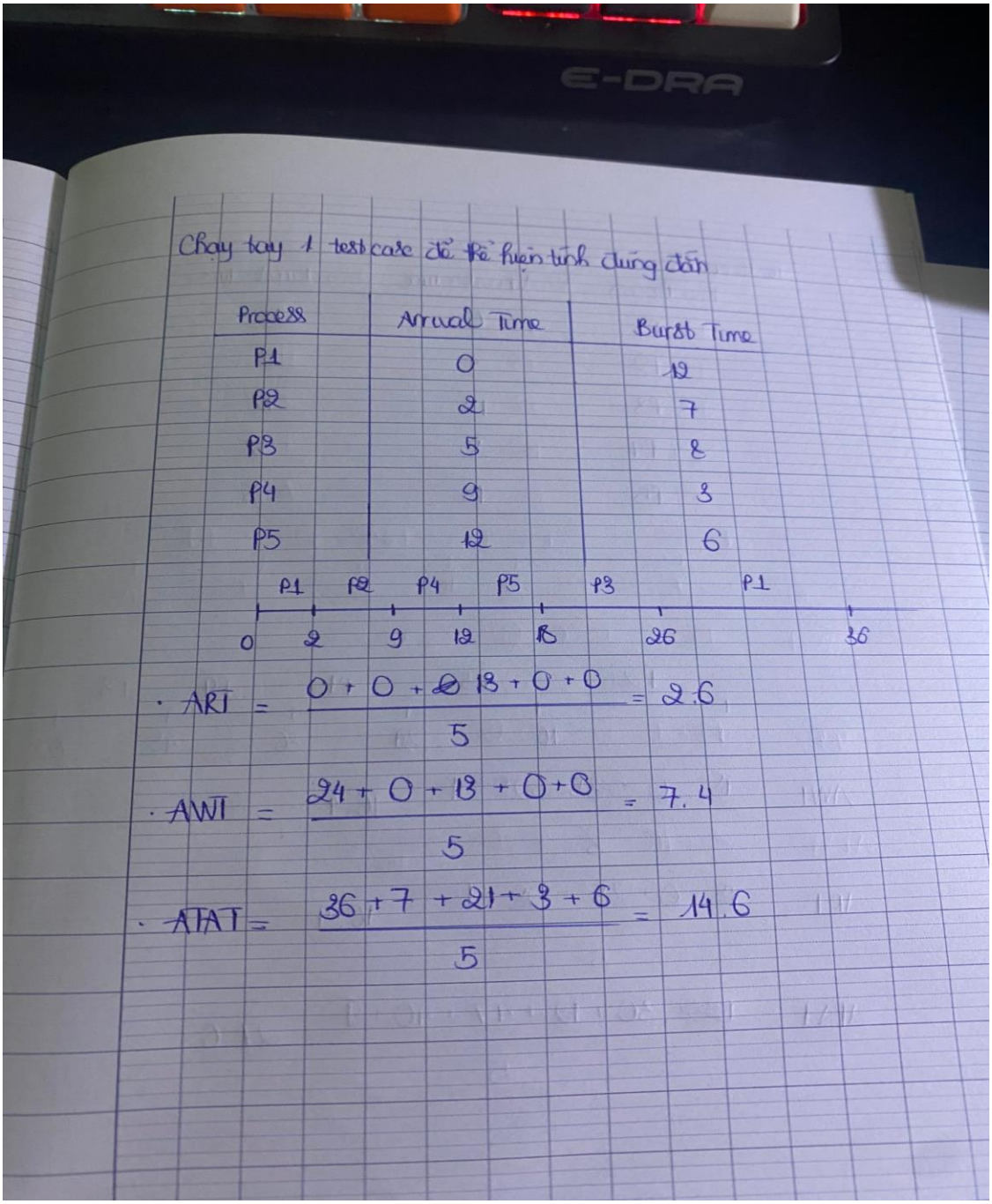
1. Giải thuật Shortest-Remaining-Time-First hoặc Round Robin

Giải thuật lựa chọn: Shortest-Remaining-Time-First

Lưu đồ giải thuật:



Chạy tay lưu đồ giải thuật:



Phần Code:

```c
#include <stdio.h>
#include <stdbool.h>
#include<limits.h>
#include<stdlib.h>
#include<time.h>
int pn[10];
int arr[10], bur[10], star[10], finish[10], tat[10], wt[10], id[10], remain[10], i, n, run[10];
float totwt = 0, tottat = 0, totrp = 0;

bool chk_remain()
{
    for (int i = 0; i < n; i++)
        if (remain[i] > 0)
            return true;
    return false;
}

void input_processes()
{
    srand(time(NULL));
    for (i = 0; i < n; i++)
    {
        pn[i] = i + 1;
        arr[i] = rand() % 21;
        bur[i] = rand() % 11 + 2;
        remain[i] = bur[i];
        id[i] = i;
        star[i] = -1;
        finish[i] = 0;
        tat[i] = 0;
        wt[i] = 0;
        run[i] = arr[i];
    }
}
void sort_processes()
{
    for (i = 0; i < n - 1; i++)
        for (int j = i + 1; j < n; j++)
            if ((arr[id[i]] > arr[id[j]]) || (arr[id[i]] == arr[id[j]] && remain[id[i]] > remain[id[j]]))
            {
                int temp = id[i];
                id[i] = id[j];
                id[j] = temp;
            }
}
```

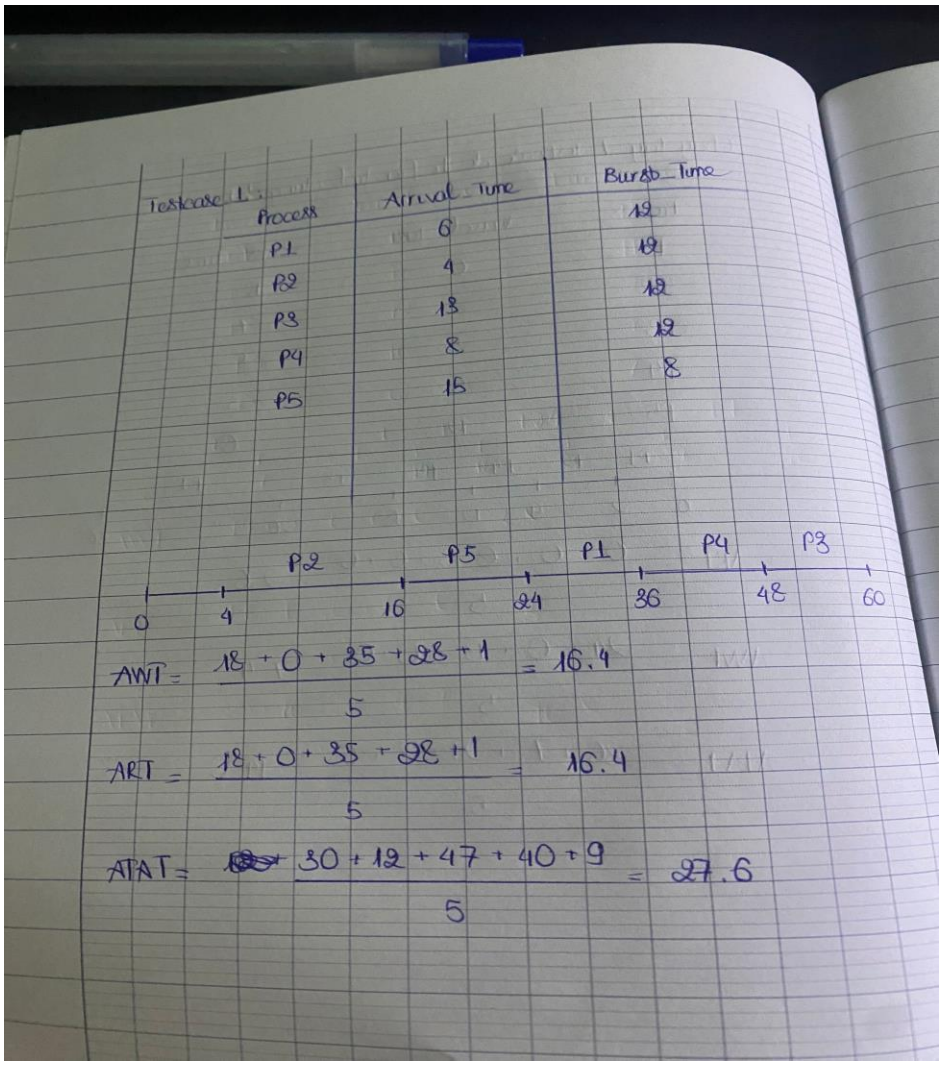
```
void execute_processes()
{
    while (chk_remain())
    {
        if (star[id[0]] == -1)
            star[id[0]] = run[id[0]];

        int min_gr = 0, min_sm = 0;

        if (remain[id[0]] > 0)
        {
            for (int i = 1; i < n; i++)
            {
                if (remain[id[i]] > 0)
                {
                    if (run[id[i]] < run[id[0]] + remain[id[0]] && run[id[i]] > run[id[0]] &&
                        remain[id[i]] < remain[id[0]] - run[id[i]] + run[id[0]])
                    {
                        if (min_gr == 0)
                            min_gr = i;
                        else
                        {
                            if ((run[id[min_gr]] > run[id[i]]) || (run[id[min_gr]] == run[id[i]] &&
                                                                  remain[id[min_gr]] > remain[id[i]]))
                                min_gr = i;
                        }
                    }
                }
            }

            remain[id[0]] = remain[id[0]] - run[id[min_gr]] + run[id[0]];
            run[id[0]] = run[id[min_gr]];
            int temp = id[min_gr];
            id[min_gr] = id[0];
            id[0] = temp;
        }

        if (min_gr == 0)
        {
            for (int i = 1; i < n; i++)
            {
                if (remain[id[i]] > 0)
                {
                    if (run[id[i]] <= run[id[0]] + remain[id[0]])
                    {
                        if (min_sm == 0)
                            min_sm = i;
                        else
                        {
                            if (remain[id[min_sm]] > remain[id[i]])
                                min_sm = i;
                        }
                    }
                }
            }

            run[id[0]] += remain[id[0]];
            finish[id[0]] = run[id[0]];
            tat[id[0]] = finish[id[0]] - arr[id[0]];
            remain[id[0]] = 0;

            wt[id[min_sm]] += (finish[id[0]] - run[id[min_sm]]);
            run[id[min_sm]] = finish[id[0]];
            int temp = id[min_sm];
            id[min_sm] = id[0];
            id[0] = temp;
        }

        if (min_gr == 0 && min_sm == 0)
        {
            int min_run = 0;

            for (int i = 1; i < n; i++)
                if (remain[id[i]] > 0)
                {
                    if (min_run == 0)
                        min_run = i;
                    else if ((run[id[min_run]] > run[id[i]]) || (run[id[min_run]] == run[id[i]] &&
                                                                remain[id[min_run]] > remain[id[i]]))
                        min_run = i;
                }

            run[id[0]] += remain[id[0]];
            finish[id[0]] = run[id[0]];
            tat[id[0]] = finish[id[0]] - arr[id[0]];
            remain[id[0]] = 0;

            int temp = id[min_run];
            id[min_run] = id[0];
            id[0] = temp;
        }
    }
}
```

```c
void print_results()
{
    printf("\nPName\tArrtime\tBurstime\tResponse\tWaiting\tTAT\tFinish");
    for (i = 0; i < n; i++)
    {
        printf("\n%d\t%d\t%d\t%d\t\t%d\t%d\t%d", pn[i], arr[i], bur[i], star[i] - arr[i], wt[i], tat[i], finish[i]);
        totrp += (star[i] - arr[i]);
        totwt += wt[i];
        tottat += tat[i];
    }

    if (n > 0)
    {
        printf("\nAverage response time = %.2f", (totrp  / n));
        printf("\nAverage waiting time = %.2f", (totwt / n));
        printf("\nAverage turnaround time = %.2f", (tottat / n));
    }
    else
    {
        printf("\nNo processes to calculate averages.");
    }
    printf("\n");
}

int main()
{
    printf("Enter the number of processes:");
    scanf("%d", &n);
    if (n <= 0 || n > 10)
    {
        printf("Invalid number of processes. Please enter a number between 1 and 10.");
        return 1;
    }
    input_processes();
    sort_processes();
    execute_processes();
    print_results();
    return 0;
}
```

Chạy tay và chạy code kiểm chứng:

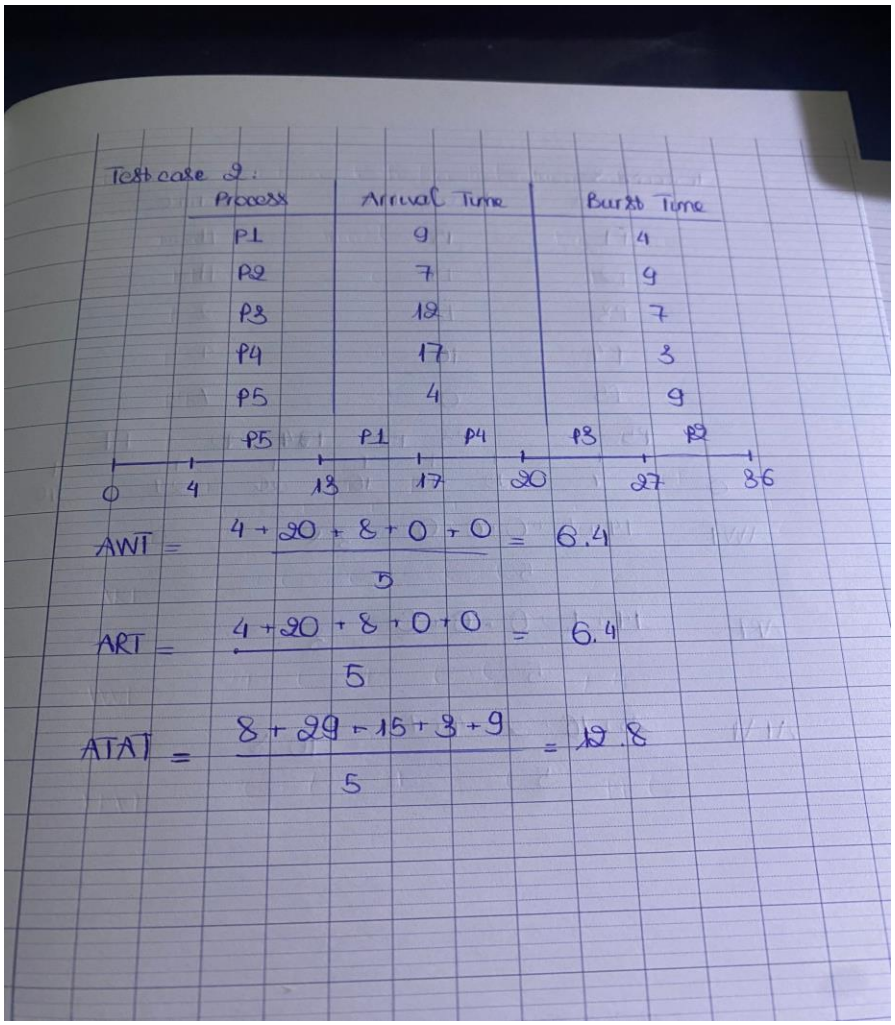
- Test case 1:

```
donghungpham@22520521-Hung:~$ ./SRTF
Enter the number of processes:5

PName   Arrtime Burtime Response        Waiting TAT     Finish
1       6       12      18              18      30      36
2       4       12      0               0       12      16
3       13      12      35              35      47      60
4       8       12      28              28      40      48
5       15      8       1               1       9       24
Average response time = 16.40
Average waiting time = 16.40
Average turnaround time = 27.60
```



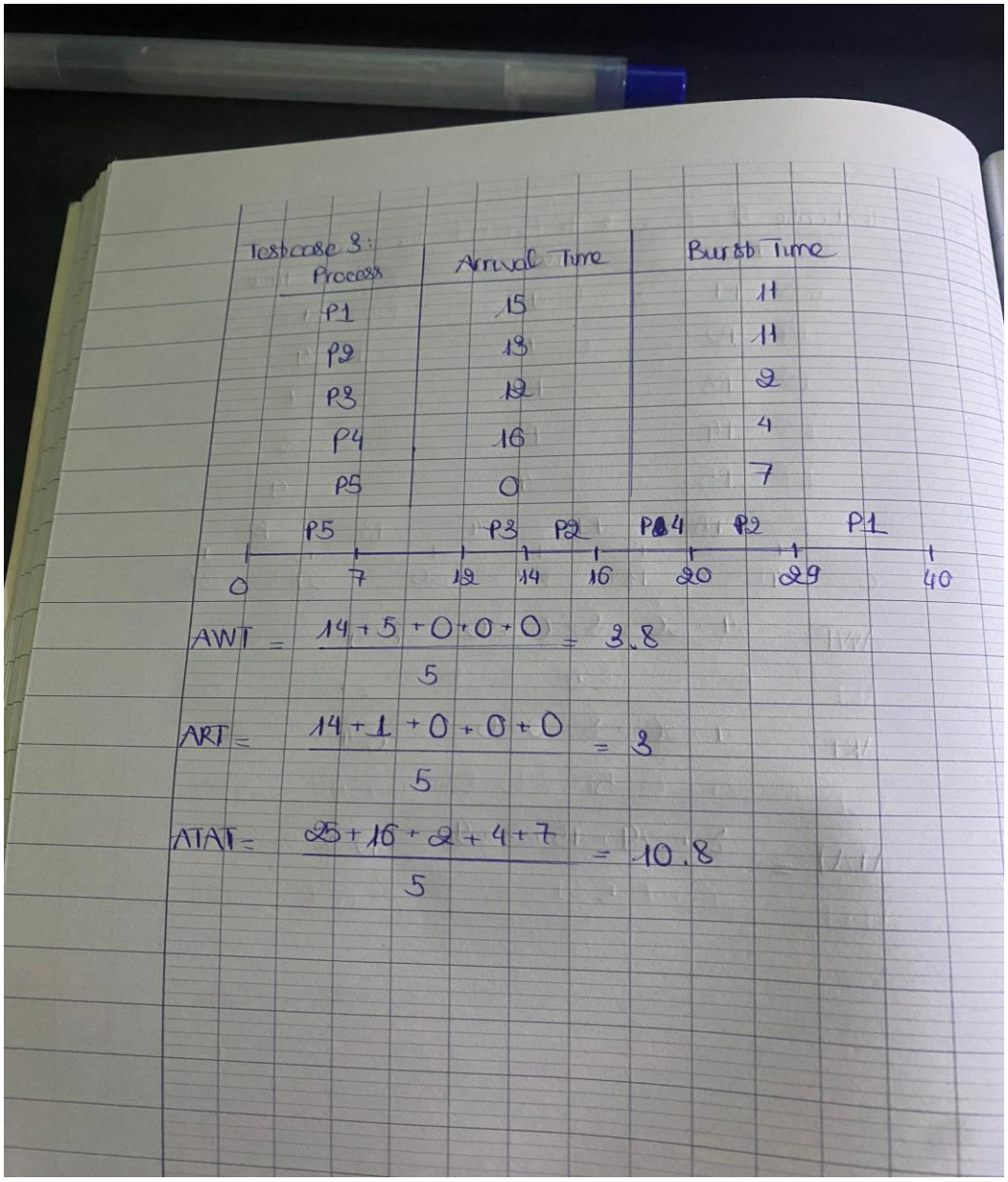
- Test case 2:

```
donghungpham@22520521-Hung:~$ ./SRTF
Enter the number of processes:5

PName   Arrtime Burtime Response        Waiting TAT     Finish
1       9       4       4               4       8       17
2       7       9       20              20      29      36
3       12      7       8               8       15      27
4       17      3       0               0       3       20
5       4       9       0               0       9       13
Average response time = 6.40
Average waiting time = 6.40
Average turnaround time = 12.80
```



- Test case 3:

```
donghungpham@22520521-Hung:~$ ./SRTF
Enter the number of processes:5

PName   Arrtime Burtime Response        Waiting TAT     Finish
1       15      11      14              14      25      40
2       13      11      1               5       16      29
3       12      2       0               0       2       14
4       16      4       0               0       4       20
5       0       7       0               0       7       7
Average response time = 3.00
Average waiting time = 3.80
Average turnaround time = 10.80
```


Testcase 3:
Process | Arrival Time | Burst Time
P1 | 15 | 11
P2 | 13 | 11
P3 | 12 | 2
P4 | 16 | 4
P5 | 0 | 7
P5 P3 P2 P4 P2 P1
0 7 12 14 16 20 29 40
AWT = 14+5+0+0+0 / 5 = 3.8
ART = 14+1+0+0+0 / 5 = 3
ATAT = 25+16+2+4+7 / 5 = 10.8