

Name: Phạm Mai Dung

ID: 19520477

Class: IT007.L21.1

OPERATING SYSTEM LAB 4'S REPORT

SUMMARY

Task		Status	Page
Section 1.5	Ex 1	Done	2 – 7
	Ex 2	Done	8 – 13
	Ex 3	Done	14 – 18

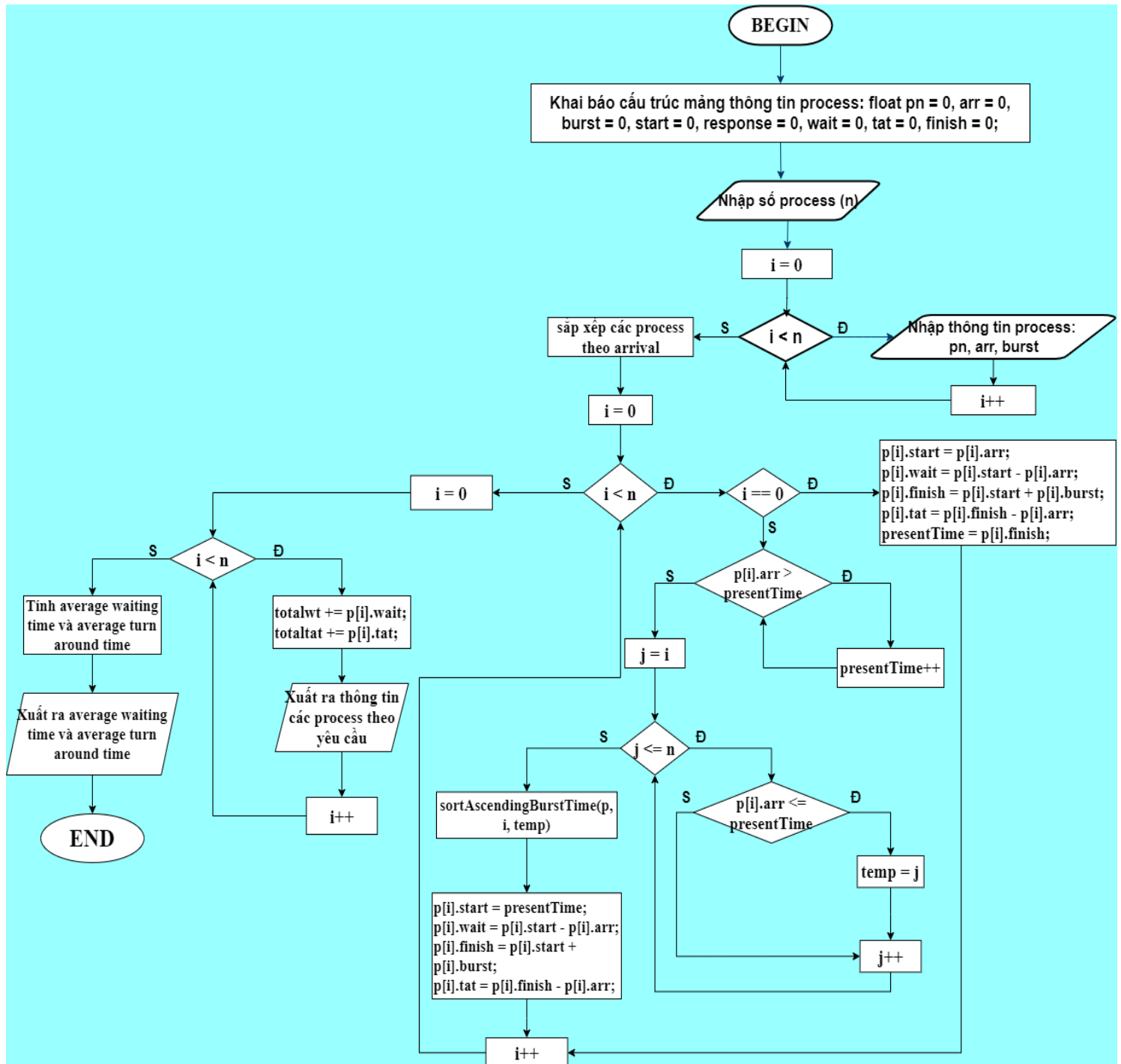
Self-scores: 9

Note: Export file to **PDF and name the file by following format:
LAB X – <Student ID>.pdf*

Section 1.5

1. Task name 1: Chương trình mô phỏng giải thuật SJF:

- Lưu đồ thuật toán:



Hình 1.1 _ lưu đồ giải thuật SJF

- Source code:

```
/*#####  
# University of Information Technology #  
# IT007 Operating System             #  
# Pham Mai Dung, 19520477            #  
# File: sjf.cpp                      #  
#####*/  
  
#include <iostream>  
  
using namespace std;  
  
struct Process  
{  
    float pn = 0, arr = 0, burst = 0, start = 0, response = 0, wait = 0,  
    tat = 0, finish = 0;  
};  
  
void swap(float& a, float& b) {  
    float temp;  
    temp = a;  
    a = b;  
    b = temp;  
}  
  
void sortAscendingBurstTime(Process p[], int m, int n) {  
    for (int i = m; i < n - 1; i++) {  
        for (int j = i + 1; j < n; j++) {  
            if (p[i].burst > p[j].burst) {  
                swap(p[i].pn, p[j].pn);  
                swap(p[i].arr, p[j].arr);  
                swap(p[i].burst, p[j].burst);  
            }  
        }  
    }  
}  
  
void sortAscendingArrivalTime(Process p[], int n)  
{  
    for (int i = 0; i < n - 1; i++)  
    {  
        for (int j = i + 1; j < n; j++)  
        {  
            if (p[i].arr > p[j].arr)  
            {  
                swap(p[i].pn, p[j].pn);  
                swap(p[i].arr, p[j].arr);  
                swap(p[i].burst, p[j].burst);  
            }  
        }  
    }  
}
```

```

        else if (p[i].arr == p[j].arr && p[i].burst > p[j].burst)
        {
            swap(p[i].pn, p[j].pn);
            swap(p[i].arr, p[j].arr);
            swap(p[i].burst, p[j].burst);
        }
    }
}

int main()
{
    Process* p;
    int num;
    cout << "Enter the number of processes: ";
    cin >> num;
    p = new Process[num];
    float totalBurst = 0;
    for (int i = 0; i < num; i++)
    {
        cout << "Enter Process Name: ";
        cin >> p[i].pn;
        cout << "Enter Arrival Time: ";
        cin >> p[i].arr;
        cout << "Enter Burst Time: ";
        cin >> p[i].burst;
    }
    sortAscendingArrivalTime(p, num);
    float presentTime = 0; int temp;
    for (int i = 0; i < num; i++)
    {
        if (i == 0)
        {
            p[i].start = p[i].arr;
            p[i].wait = p[i].start - p[i].arr;
            p[i].response = p[i].wait;
            p[i].finish = p[i].start + p[i].burst;
            p[i].tat = p[i].finish - p[i].arr;
            presentTime = p[i].finish;
        }
        else
        {
            while (p[i].arr > presentTime)
            {
                presentTime++;
            }
            for (int j = i; j <= num; j++)
            {
                if (p[j].arr <= presentTime)
                    temp = j;
            }

```

```

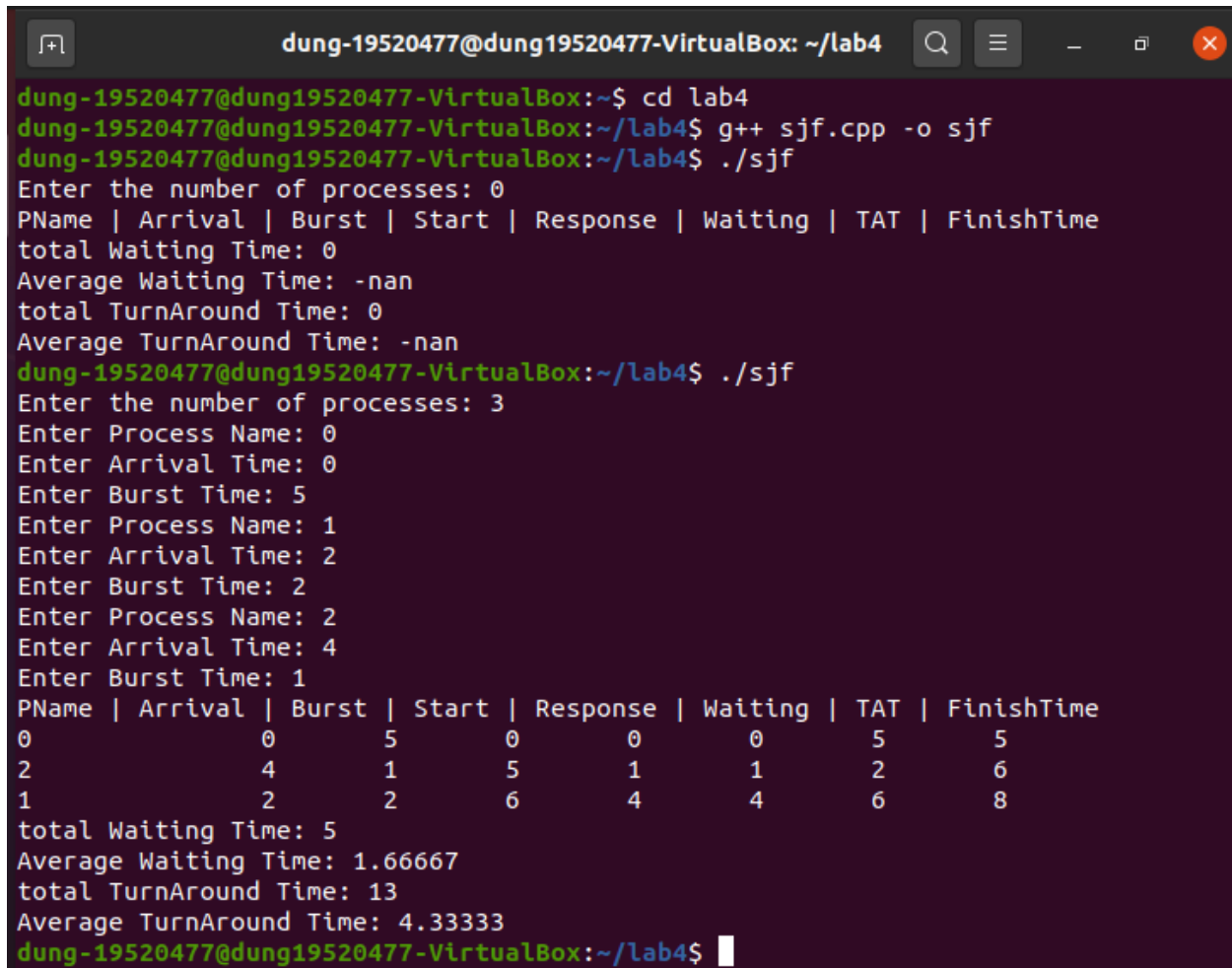
        sortAscendingBurstTime(p, i, temp);
        p[i].start = presentTime;
        p[i].wait = p[i].start - p[i].arr;
        p[i].response = p[i].wait;
        p[i].finish = p[i].start + p[i].burst;
        p[i].tat = p[i].finish - p[i].arr;
        presentTime = p[i].finish;
    }
}
float totalwt = 0, totaltat = 0;
cout << "PName | Arrival | Burst | Start | Response | Waiting | TAT |
FinishTime\n";
for (int i = 0; i < num; i++)
{
    cout << p[i].pn << "\t\t" << p[i].arr << "\t" << p[i].burst <<
"\t" << p[i].start << "\t" << p[i].response << "\t" << p[i].wait << "\t"
<< p[i].tat << "\t" << p[i].finish << "\n";
    totalwt += p[i].wait;
    totaltat += p[i].tat;
}
cout << "total Waiting Time: " << totalwt << endl;
cout << "Average Waiting Time: " << (float)totalwt / num << endl;
cout << "total TurnAround Time: " << totaltat << endl;
cout << "Average TurnAround Time: " << (float)totaltat / num << endl;
return 0;
}

```

- Giải thích:

- Khai báo một mảng cấu trúc process có n phần tử chứa thông tin của process.
- Viết hàm swap(float& a, float& b) để hoán vị 2 số a, b nhằm phục vụ các hàm sắp xếp.
- Hàm sortAscendingBurstTime() để sắp xếp các process theo thứ tự tăng dần burst time.
- Hàm sortAscendingArrivalTime() để sắp xếp các process theo thứ tự tăng dần arrival time, sắp xếp khi process vào không theo thứ tự.
- Sau khi nhập thông tin processes và sắp xếp theo arrival time, lúc này các phần tử trong mảng đã được gán theo thứ tự sắp xếp, process thứ 0 sẽ được thực thi (tính các giá trị theo yêu cầu) đầu tiên. Sau khi nó chạy xong, cập nhật biến temp = phần tử cuối cùng, sắp xếp các process theo burst time tăng dần từ phần tử i đến temp, rồi cho các process còn lại thực thi lần lượt. Xuất ra các thông tin process (arrival, burst, response, waiting, turn around, finish time); tính waiting time và turn around time trung bình, sau đó xuất ra màn hình, ta được kết quả như hình dưới.
- Source code được viết bằng c++, do đó em đã cài ide g++ và thực thi chương trình bằng lệnh `g++ sjf.cpp -o sjf`.

- Kết quả:



```
dung-19520477@dung19520477-VirtualBox: ~/lab4
dung-19520477@dung19520477-VirtualBox:~$ cd lab4
dung-19520477@dung19520477-VirtualBox:~/lab4$ g++ sjf.cpp -o sjf
dung-19520477@dung19520477-VirtualBox:~/lab4$ ./sjf
Enter the number of processes: 0
PName | Arrival | Burst | Start | Response | Waiting | TAT | FinishTime
total Waiting Time: 0
Average Waiting Time: -nan
total TurnAround Time: 0
Average TurnAround Time: -nan
dung-19520477@dung19520477-VirtualBox:~/lab4$ ./sjf
Enter the number of processes: 3
Enter Process Name: 0
Enter Arrival Time: 0
Enter Burst Time: 5
Enter Process Name: 1
Enter Arrival Time: 2
Enter Burst Time: 2
Enter Process Name: 2
Enter Arrival Time: 4
Enter Burst Time: 1
PName | Arrival | Burst | Start | Response | Waiting | TAT | FinishTime
0      | 0        | 5      | 0      | 0         | 0        | 5      | 5
2      | 4        | 1      | 5      | 1         | 1        | 2      | 6
1      | 2        | 2      | 6      | 4         | 4        | 6      | 8
total Waiting Time: 5
Average Waiting Time: 1.66667
total TurnAround Time: 13
Average TurnAround Time: 4.33333
dung-19520477@dung19520477-VirtualBox:~/lab4$
```

Hình 1.2 _ Trường hợp không có process nào và trường hợp process đầu tiên đến lúc 0

```

dung-19520477@dung19520477-VirtualBox:~/lab4$ ./sjf
Enter the number of processes: 5
Enter Process Name: 1
Enter Arrival Time: 1
Enter Burst Time: 4
Enter Process Name: 2
Enter Arrival Time: 0
Enter Burst Time: 6
Enter Process Name: 3
Enter Arrival Time: 2
Enter Burst Time: 2
Enter Process Name: 4
Enter Arrival Time: 2
Enter Burst Time: 1
Enter Process Name: 5
Enter Arrival Time: 4
Enter Burst Time: 1
PName | Arrival | Burst | Start | Response | Waiting | TAT | FinishTime
2      |         | 0     | 6     | 0         | 0        | 6   | 6
4      |         | 2     | 6     | 4         | 4        | 5   | 7
5      |         | 4     | 7     | 3         | 3        | 4   | 8
3      |         | 2     | 8     | 6         | 6        | 8   | 10
1      |         | 1     | 10    | 9         | 9        | 13  | 14
total Waiting Time: 22
Average Waiting Time: 4.4
total TurnAround Time: 36
Average TurnAround Time: 7.2
dung-19520477@dung19520477-VirtualBox:~/lab4$ █

```

Hình 1.3 _ Có 5 process, trong đó process đầu tiên arrival != 0, có 2 process arrival time bằng nhau và 2 process có burst time bằng nhau.

```

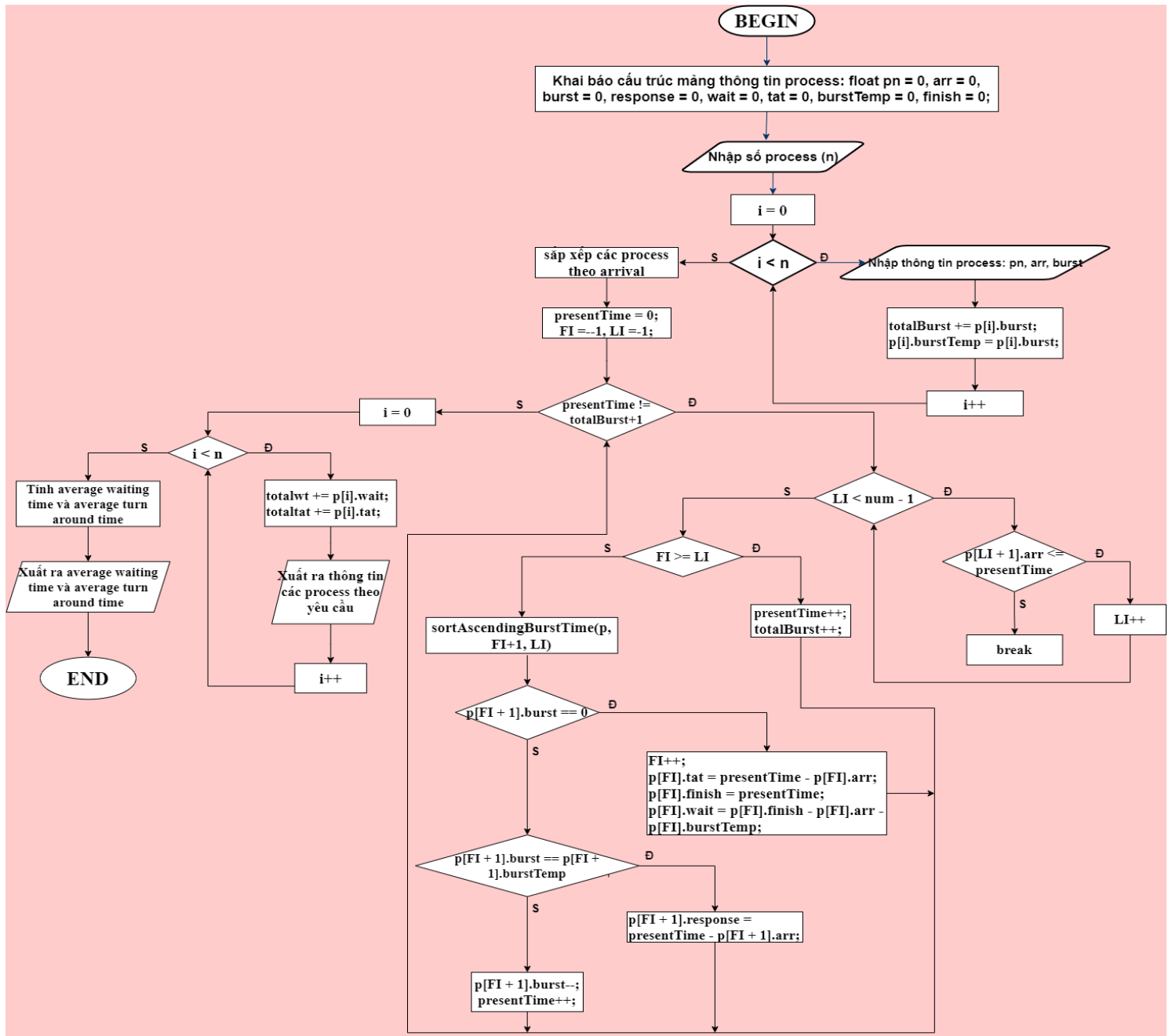
dung-19520477@dung19520477-VirtualBox:~/lab4$ ./sjf
Enter the number of processes: 3
Enter Process Name: 1
Enter Arrival Time: 0
Enter Burst Time: 3
Enter Process Name: 2
Enter Arrival Time: 4
Enter Burst Time: 4
Enter Process Name: 3
Enter Arrival Time: 4
Enter Burst Time: 2
PName | Arrival | Burst | Start | Response | Waiting | TAT | FinishTime
1      |         | 3     | 0     | 0         | 0        | 3   | 3
3      |         | 4     | 4     | 0         | 0        | 2   | 6
2      |         | 4     | 6     | 2         | 2        | 6   | 10
total Waiting Time: 2
Average Waiting Time: 0.666667
total TurnAround Time: 11
Average TurnAround Time: 3.66667
dung-19520477@dung19520477-VirtualBox:~/lab4$ █

```

Hình 1.4 _ Trường hợp có thời điểm 1 tiến process kết thúc mà chưa có process nào vào hàng đợi

2. Task name 2: Chương trình mô phỏng giải thuật SRTF:

- Lưu đồ thuật toán:



Hình 2.1 _ Lưu đồ giải thuật SRTF

- Source code:

```
/*#####  
# University of Information Technology #  
# IT007 Operating System #  
# Pham Mai Dung, 19520477 #  
# File: srtf.cpp #  
#####*/  
  
#include <iostream>  
  
using namespace std;  
  
struct Process  
{  
    float pn = 0, arr = 0, burst = 0, response = 0, wait = 0, tat = 0,  
    burstTemp = 0, finish = 0;  
};  
  
void swap(float& a, float& b) {  
    float temp;  
    temp = a;  
    a = b;  
    b = temp;  
}  
  
void sortAscendingBurstTime(Process p[], int m, int n) {  
    for (int i = m; i < n; i++) {  
        for (int j = i + 1; j <= n; j++) {  
            if (p[i].burst > p[j].burst) {  
                swap(p[i].pn, p[j].pn);  
                swap(p[i].arr, p[j].arr);  
                swap(p[i].burst, p[j].burst);  
                swap(p[i].burstTemp, p[j].burstTemp);  
            }  
        }  
    }  
}  
  
void sortAscendingArrivalTime(Process p[], int n)  
{  
    for (int i = 0; i < n - 1; i++)  
    {  
        for (int j = i + 1; j < n; j++)  
        {  
            if (p[i].arr > p[j].arr)  
            {  
                swap(p[i].pn, p[j].pn);  
                swap(p[i].arr, p[j].arr);  
                swap(p[i].burst, p[j].burst);  
            }  
        }  
    }  
}
```

```

        swap(p[i].burstTemp, p[j].burstTemp);
    }
}
}

int main() {
    Process* p;
    int num;
    cout << "Enter the number of processes: ";
    cin >> num;
    p = new Process[num];
    float totalBurst = 0;
    for (int i = 0; i < num; i++)
    {
        cout << "Enter Process Name: ";
        cin >> p[i].pn;
        cout << "Enter Arrival Time: ";
        cin >> p[i].arr;
        cout << "Enter Burst Time: ";
        cin >> p[i].burst;
        totalBurst += p[i].burst;
        p[i].burstTemp = p[i].burst;
    }
    sortAscendingArrivalTime(p, num);
    float presentTime = 0;
    int FirstIndex = -1, LastIndex = -1;
    while (presentTime != totalBurst + 1)
    {
        while (LastIndex < num - 1)
        {
            if (p[LastIndex + 1].arr <= presentTime)
                LastIndex++;
            else break;
        }
        if (FirstIndex >= LastIndex)
        {
            presentTime++;
            totalBurst++;
        }
        else
        {
            sortAscendingBurstTime(p, FirstIndex + 1, LastIndex);
            if (p[FirstIndex + 1].burst == 0)
            {
                FirstIndex++;
                p[FirstIndex].tat = presentTime - p[FirstIndex].arr;
                p[FirstIndex].finish = presentTime;
                p[FirstIndex].wait = p[FirstIndex].finish -
p[FirstIndex].arr - p[FirstIndex].burstTemp;

```

```

    }
    else
    {
        if (p[FirstIndex + 1].burst == p[FirstIndex +
1].burstTemp)
            p[FirstIndex + 1].response = presentTime -
p[FirstIndex + 1].arr;
            p[FirstIndex + 1].burst--;
            presentTime++;
        }
    }
}
float totalwt = 0, totaltat = 0;
cout << "PName | Arrival | Burst | Response | Waiting | TAT |
FinishTime\n";
for (int i = 0; i < num; i++)
{
    cout << p[i].pn << "\t\t" << p[i].arr << "\t" << p[i].burstTemp
<< "\t" << p[i].response << "\t" << p[i].wait << "\t" << p[i].tat << "\t"
<< p[i].finish << "\n";
    totalwt += p[i].wait;
    totaltat += p[i].tat;
}
cout << "total Waiting Time: " << totalwt << endl;
cout << "Average Waiting Time: " << (float)totalwt / num << endl;
cout << "total TurnAround Time: " << totaltat << endl;
cout << "Average TurnAround Time: " << (float)totaltat / num << endl;
return 0;
}

```

- Giải thích:

- Khai báo một mảng cấu trúc process có n phần tử chứa thông tin của process.
- Viết hàm swap(float& a, float& b) để hoán vị 2 số a, b nhằm phục vụ các hàm sắp xếp.
- Hàm sortAscendingBurstTime() để sắp xếp các process theo thứ tự tăng dần burst time.
- Hàm sortAscendingArrivalTime() để sắp xếp các process theo thứ tự tăng dần arrival time, sắp xếp khi process vào không theo thứ tự.
- Sau khi nhập thông tin processes và sắp xếp theo arrival time, lúc này các phần tử trong mảng đã được gán theo thứ tự sắp xếp.
- Khai báo các biến: totalBurst_tính tổng burst time của các process; burstTemp_lưu giá trị của burst trong quá trình thực thi sẽ bị thay đổi (burst--); FirstIndex và LastIndex_lưu vị trí đầu cuối hàng đợi; presentTime_ thời gian hoạt động, khi ==totalBurst là các process được thực thi hết.

- Khi `presentTime != totalBurst`, cập nhật `LastIndex ==` vị trí cuối hàng đợi, sắp xếp mảng process tăng dần từ `FirstIndex+1` đến `LastIndex`. Nếu `FI >= LI` không có process nào trong hàng đợi, cập nhật `totalBurst` và `presentTime` tăng đến khi có process vào hàng đợi. Nếu có process trong hàng đợi, hàm sắp xếp tăng dần `bursttime` và thực thi process có burst nhỏ nhất (cập nhật các giá trị `response`, `tat`, `waiting`,...) và cập nhật tiếp thời điểm tiếp theo (burst process đang chạy sẽ giảm dần, `presentTime` tăng dần) và quay lại vòng lặp.
 - Tính `waiting` và `turn around time` trung bình, xuất ra các thông tin theo yêu cầu.
 - Thực thi chương trình bằng lệnh `g++ srtf.cpp -o srtf`.
- Kết quả:

```
dung-19520477@dung19520477-VirtualBox:~/lab4$ g++ srtf.cpp -o srtf
dung-19520477@dung19520477-VirtualBox:~/lab4$ ./srtf
Enter the number of processes: 0
PName | Arrival | Burst | Response | Waiting | TAT | FinishTime
total Waiting Time: 0
Average Waiting Time: -nan
total TurnAround Time: 0
Average TurnAround Time: -nan
dung-19520477@dung19520477-VirtualBox:~/lab4$ ./srtf
Enter the number of processes: 3
Enter Process Name: 1
Enter Arrival Time: 0
Enter Burst Time: 6
Enter Process Name: 2
Enter Arrival Time: 1
Enter Burst Time: 3
Enter Process Name: 3
Enter Arrival Time: 0
Enter Burst Time: 2
PName | Arrival | Burst | Response | Waiting | TAT | FinishTime
3      | 0       | 2     | 0        | 0       | 2   | 2
2      | 1       | 3     | 1        | 1       | 4   | 5
1      | 0       | 6     | 5        | 5       | 11  | 11
total Waiting Time: 6
Average Waiting Time: 2
total TurnAround Time: 17
Average TurnAround Time: 5.66667
dung-19520477@dung19520477-VirtualBox:~/lab4$
```

Hình 2.2_ Trường hợp không có process nào và trường hợp process đầu tiên đến lúc 0.

```

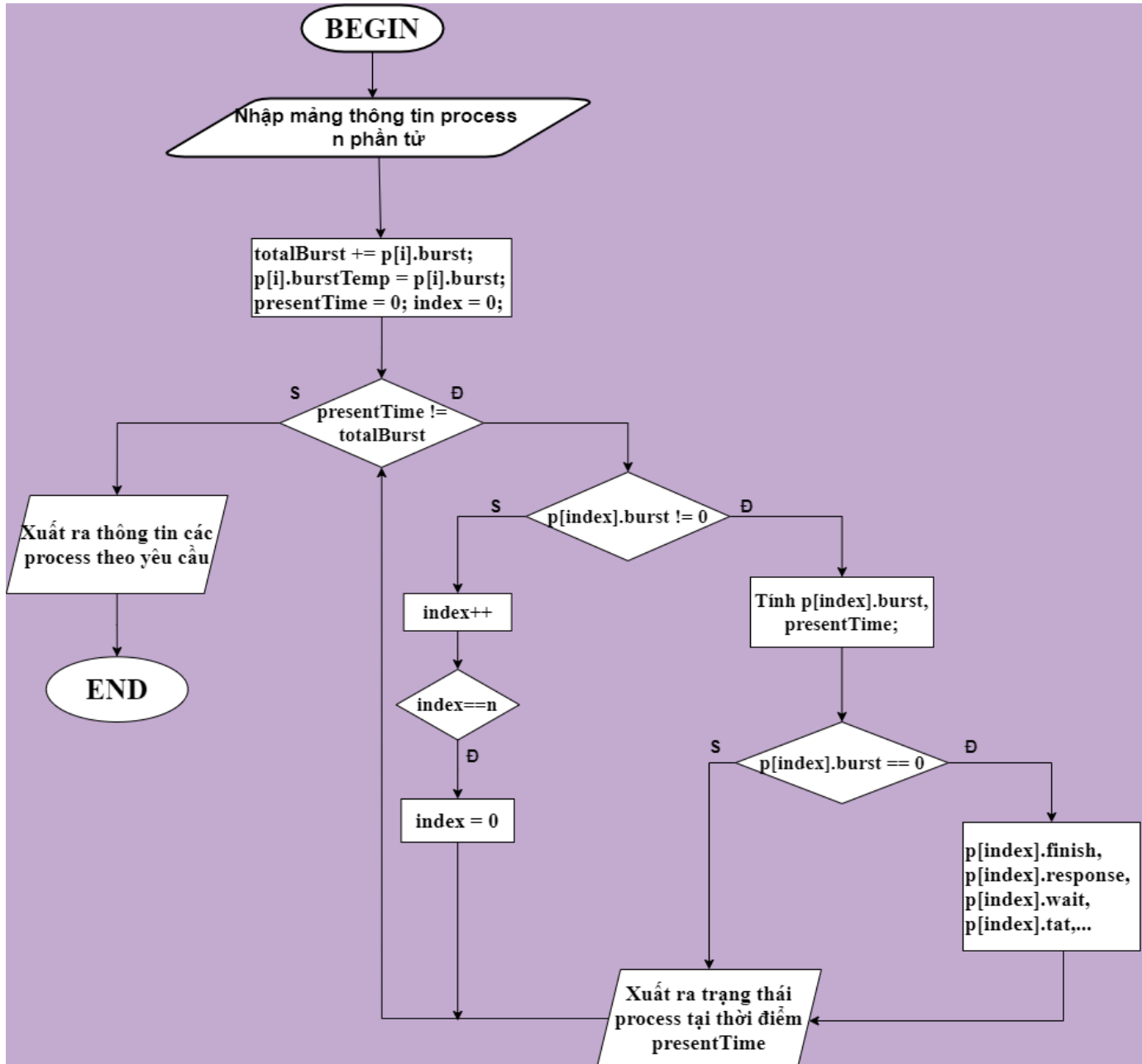
dung-19520477@dung19520477-VirtualBox:~/lab4$ ./srtf
Enter the number of processes: 5
Enter Process Name:
1
Enter Arrival Time: 1
Enter Burst Time: 4
Enter Process Name: 3
Enter Arrival Time: 0
Enter Burst Time: 6
Enter Process Name: 2
Enter Arrival Time: 2
Enter Burst Time: 2
Enter Process Name: 4
Enter Arrival Time: 3
Enter Burst Time: 1
Enter Process Name: 5
Enter Arrival Time: 4
Enter Burst Time: 1
PName | Arrival | Burst | Response | Waiting | TAT | FinishTime
2      |         | 2     | 0         | 0        | 2   | 4
4      |         | 1     | 1         | 1        | 2   | 5
5      |         | 1     | 1         | 1        | 2   | 6
1      | 1       | 4     | 0         | 4        | 8   | 9
3      | 0       | 6     | 0         | 8        | 14  | 14
total Waiting Time: 14
Average Waiting Time: 2.8
total TurnAround Time: 28
A ShowApplications and Time: 5.6
dung-19520477@dung19520477-VirtualBox:~/lab4$

```

Hình 2.3 _ Kết quả mô phỏng SRTF với 5 processes

3. Task name 3: Chương trình mô phỏng giải thuật RR:

- Lưu đồ giải thuật:



Hình 3.1 _ Lưu đồ giải thuật Round Robin.

- Source code:

```
/*#####  
# University of Information Technology #  
# IT007 Operating System #  
# Pham Mai Dung, 19520477 #  
# File: roundRobin.cpp #  
#####*/  
  
#include <iostream>  
  
using namespace std;  
  
struct Process  
{  
    float pn = 0, burst = 0, response = 0, wait = 0, tat = 0, burstTemp =  
    0, finish = 0;  
};  
  
int main()  
{  
    Process* p;  
    int num;  
    float qTime;  
    cout << "Nhap so tien trinh: ";  
    cin >> num;  
    p = new Process[num];  
    float totalBurst = 0;  
    for (int i = 0; i < num; i++)  
    {  
        cout << "Process Name: ";  
        cin >> p[i].pn;  
        cout << "Burst Time: ";  
        cin >> p[i].burst;  
        totalBurst += p[i].burst;  
        p[i].burstTemp = p[i].burst;  
    }  
    cout << "Nhap Quantum Time: ";  
    cin >> qTime;  
    cout << "PName" << "\t" << "Time" << "\t" << "Status" << endl;  
    float presentTime = 0;  
    int index = 0;  
    while (presentTime != totalBurst)  
    {  
        if (p[index].burst == 0)  
        {  
            index++;  
            if (index == num)  
                index = 0;  
        }  
    }  
}
```

```

else
{
    if (p[index].burst > qTime)
    {
        if (p[index].burst == p[index].burstTemp)
        {
            p[index].response = presentTime;
        }
        cout << p[index].pn << "\t" << presentTime <<
"\t" << " Start \n";

        presentTime = presentTime + qTime;
        cout << p[index].pn << "\t" << presentTime <<
"\t" << " Stop \n";

        p[index].burst = p[index].burst - qTime;
        index++;
        if (index == num)
            index = 0;
    }
    else
    {
        if (p[index].burst == p[index].burstTemp)
        {
            p[index].response = presentTime;
        }
        cout << p[index].pn << "\t" << presentTime <<
"\t" << " Start \n";

        presentTime = presentTime + p[index].burst;
        cout << p[index].pn << "\t" << presentTime <<
"\t" << " Stop \n";

        p[index].burst = 0;
        p[index].finish = presentTime;
        p[index].wait = p[index].finish -
p[index].burstTemp;

        p[index].tat = p[index].finish -
p[index].response;

        index++;
        if (index == num) index = 0;
    }
}

}

float totalwt = 0, totaltat = 0;
for (int i = 0; i < num; i++)
{
    totaltat += p[i].tat;
    totalwt += p[i].wait;
}
cout << "total Waiting Time: " << totalwt << endl;
cout << "Average Waiting Time: " << (float)totalwt / num << endl;
cout << "total TurnAround Time: " << totaltat << endl;

```



```

        cout << "Average TurnAround Time: " << (float)totaltat / num <<
endl;
        return 0;
}

```

- Giải thích:

- Khai báo một mảng cấu trúc process có n phần tử chứa thông tin của process.
- Nhập thông tin các process, tính totalBurst, cập nhật burstTemp, nhập quantumTime.
- Nếu $p[i].burst \neq 0$, $burst > quantumTime$ thì $presentTime = presentTime + Quantum$ và $burst = burst - quantumTime$, ngược lại thì $Time = Time + Burst$, $Burst = 0 \Rightarrow$ xuất ra trạng thái của $p[i]$ tại thời điểm presentTime.
- Index tăng dần đến num và lặp lại đến khi các process được thực thi hết.
- Tính waiting và turn around time trung bình, xuất ra các thông tin.
- Thực thi chương trình bằng lệnh `g++ srtf.cpp -o srtf`.

- Kết quả:

```

dung-19520477@dung19520477-VirtualBox:~/lab4$ g++ roundRobin.cpp -o rr
dung-19520477@dung19520477-VirtualBox:~/lab4$ ./rr
Nhap so tien trinh: 4
Process Name: 1
Burst Time: 5
Process Name: 2
Burst Time: 4
Process Name: 3
Burst Time: 7
Process Name: 4
Burst Time: 2
Nhap Quantum Time: 3

```

Hình 3.2 _ Nhập các process, quantumTime

```
PName    Time    Status
1         0      Start
1         3      Stop
2         3      Start
2         6      Stop
3         6      Start
3         9      Stop
4         9      Start
4        11      Stop
1        11      Start
1        13      Stop
2        13      Start
2        14      Stop
3        14      Start
3        17      Stop
3        17      Start
3        18      Stop
total Waiting Time: 38
Average Waiting Time: 9.5
total TurnAround Time: 38
Average TurnAround Time: 9.5
dung-19520477@dung19520477-VirtualBox:~/lab4$
```

Hình 3.3 _ Kết quả mô phỏng round robin theo các dữ liệu vừa nhập trên