

NHÓM 2	
Nguyễn Trần Bảo Anh	22520066
Trần Đồng Trúc Lam	22520746
Trần Mạnh Kiên	22520711
Hồ Văn Trí	23521637
Lớp: IT007.O21.CNVN	

HỆ ĐIỀU HÀNH

BÁO CÁO LAB 4

CHECKLIST

3.5. BÀI TẬP THỰC HÀNH

	BT 1	BT 2
Vẽ lưu đồ giải thuật	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Chạy tay lưu đồ giải thuật	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Hiện thực code	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Chạy code và kiểm chứng	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

3.6. BÀI TẬP ÔN TẬP

	BT 1
Vẽ lưu đồ giải thuật	<input checked="" type="checkbox"/>
Chạy tay lưu đồ giải thuật	<input checked="" type="checkbox"/>
Hiện thực code	<input checked="" type="checkbox"/>
Chạy code và kiểm chứng	

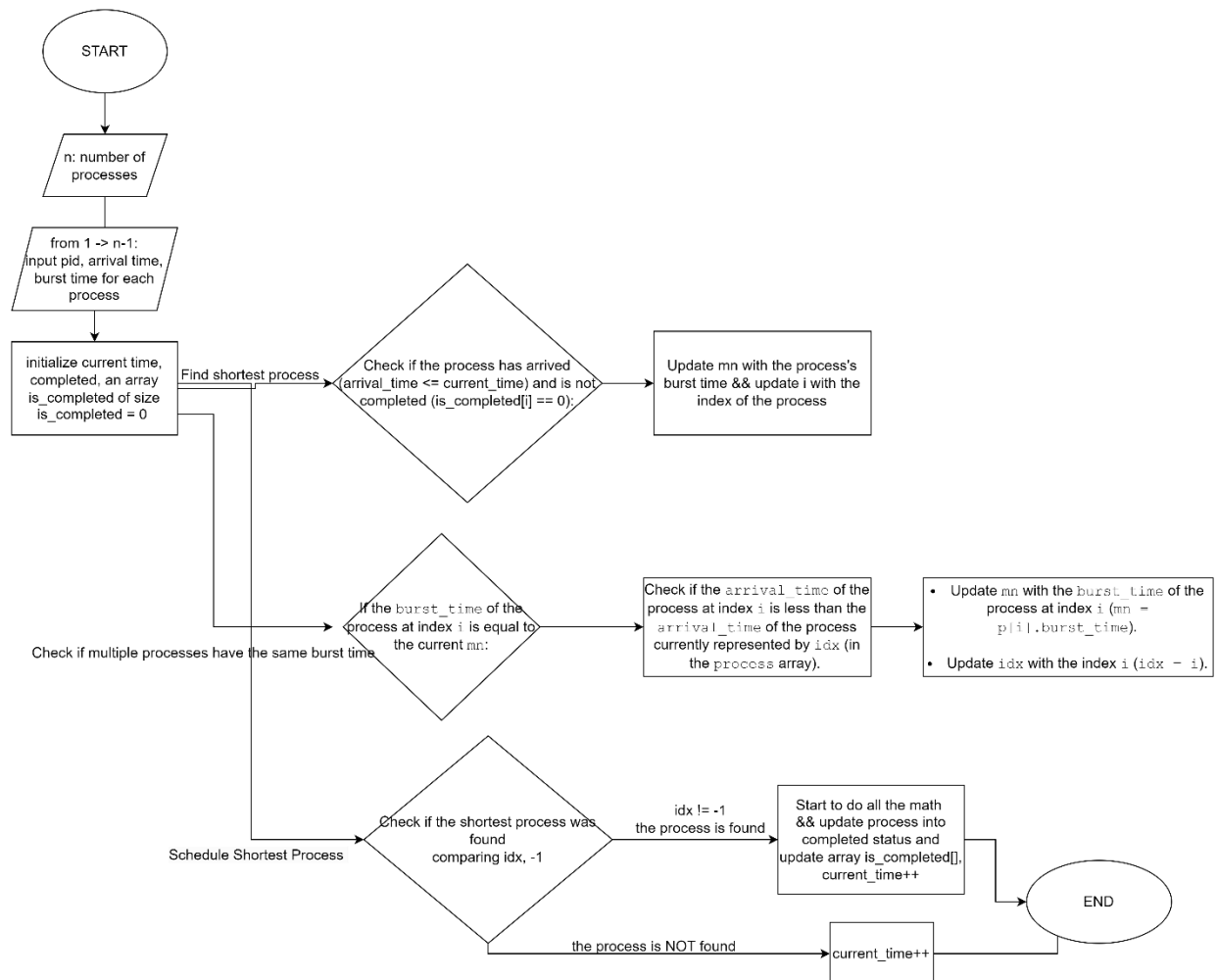
Tự chấm điểm:

2.5. BÀI TẬP THỰC HÀNH

1. Giải thuật Shortest-Job-First

Trả lời:

1.1. Flowchart giải thuật



1.2. Code của giải thuật

```
#include <iostream>
#include <algorithm>
#include <iomanip>
#include <string.h>
```

```
using namespace std;

struct process
{
    char pid;
    int arrival_time;
    int burst_time;
    int start_time;
    int finish_time;
    int turnaround_time;
    int waiting_time;
    int response_time;
};

bool compareArrival(process p1, process p2)
{
    return p1.arrival_time < p2.arrival_time;
}

int main()
{
    int n;
    struct process p[100];
    float avg_turnaround_time;
    float avg_waiting_time;
    float avg_response_time;
    int total_turnaround_time = 0;
    int total_waiting_time = 0;
    int total_response_time = 0;

    int current_time = 0;
```

```
int completed = 0;
int is_completed[100];
memset(is_completed, 0, sizeof(is_completed));

cout << setprecision(2) << fixed;

cout << "\n\nEnter the number of processes: ";
cin >> n;

for (int i = 0; i < n; i++)
{
    // Assigning PID
    char pid;
    if (i < 26)
        pid = 'A' + i;
    else
        pid = 'A' + (i - 26);
    p[i].pid = pid;
    cout << "\nEnter arrival time of process " << p[i].pid << ": ";
    cin >> p[i].arrival_time;
    cout << "Enter burst time of process " << p[i].pid << ": ";
    cin >> p[i].burst_time;
}

while (completed != n)
{
    int idx = -1;
    int mn = 10000000;
    for (int i = 0; i < n; i++)
    {
        // Check if it's arrived and has not been completed yet
        if (p[i].arrival_time <= current_time && is_completed[i] == 0)
```

```
{
    if (p[i].burst_time < mn)
    {
        mn = p[i].burst_time;
        idx = i;
    }
    // Handling multiple processes have the same burst time, choose based on its
arrival time
    if (p[i].burst_time == mn)
    {
        if (p[i].arrival_time < p[idx].arrival_time)
        {
            mn = p[i].burst_time;
            idx = i;
        }
    }
}

if (idx != -1)
{
    p[idx].start_time = current_time;
    p[idx].finish_time = p[idx].start_time + p[idx].burst_time;
    p[idx].turnaround_time = p[idx].finish_time - p[idx].arrival_time;
    p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
    p[idx].response_time = p[idx].start_time - p[idx].arrival_time;

    total_turnaround_time += p[idx].turnaround_time;
    total_waiting_time += p[idx].waiting_time;
    total_response_time += p[idx].response_time;

    is_completed[idx] = 1;
}
```

```
        completed++;
        current_time = p[idx].finish_time;
    }
    else
    {
        current_time++;
    }
}

avg_turnaround_time = (float)total_turnaround_time / n;
avg_waiting_time = (float)total_waiting_time / n;
avg_response_time = (float)total_response_time / n;

sort(p, p + n, compareArrival);

printf("\n\n      ===== SJF Scheduling =====      \n");
cout << "#P\t"
    << "AT\t"
    << "BT\t"
    << "ST\t"
    << "FT\t"
    << "TAT\t"
    << "WT\t"
    << "RT\t"
    << "\n";
for (int i = 0; i < n; i++)
{
    cout << p[i].pid << "\t" << p[i].arrival_time << "\t" << p[i].burst_time << "\t"
    << p[i].start_time << "\t" << p[i].finish_time << "\t" << p[i].turnaround_time << "\t"
    << p[i].waiting_time << "\t" << p[i].response_time << "\t"
    << "\n";
}
```

```
cout << "\nAverage Turnaround Time = " << avg_turnaround_time << endl;  
cout << "Average Waiting Time = " << avg_waiting_time << endl;  
cout << "Average Response Time = " << avg_response_time << endl;  
}
```

```
/*
```

AT - Arrival Time của tiến trình

BT - Burst time của tiến trình

ST - Thời gian bắt đầu của tiến trình

FT - Thời gian kết thúc của tiến trình

TAT - Thời gian hoàn thành của tiến trình

WT - Thời gian đợi của tiến trình

RT - Thời gian đáp ứng của tiến trình

Công thức sử dụng:

$TAT = FT - AT$

$WT = TAT - BT (= FT - AT - BT)$

$RT = ST - AT$

```
*/
```

1.3. Testcases

a. Ví dụ 1:

Process 1	Arrival Time	Burst Time
P1	0	9
P2	4	5

P3	2	7
P4	8	10
P5	10	13

— Kết quả khi chạy code


```
$ g++ *.cpp -o sjf && ./sjf

Enter the number of processes: 5

Enter arrival time of process A: 0
Enter burst time of process A: 9

Enter arrival time of process B: 4
Enter burst time of process B: 5

Enter arrival time of process C: 2
Enter burst time of process C: 7

Enter arrival time of process D: 8
Enter burst time of process D: 10

Enter arrival time of process E: 10
Enter burst time of process E: 13

===== SJF Scheduling =====
#P    AT    BT    ST    FT    TAT    WT    RT
A      0     9     0     9     9      0     0
C      2     7    14    21    19     12    12
B      4     5     9    14    10      5     5
D      8    10    21    31    23     13    13
E     10    13    31    44    34     21    21

Average Turnaround Time = 19.00
Average Waiting Time = 10.20
Average Response Time = 10.20
```

Hình 1: Kết quả khi giải ví dụ 1 bằng code giải thuật SJF

- Kết quả khi giải tay

+ Giản đồ Gantt:



+ Thời gian đáp ứng:

$$P1 = 0, P2 = 5, P3 = 12, P4 = 13, P5 = 21$$

$$\Rightarrow \text{Thời gian đáp ứng trung bình: } (0 + 5 + 12 + 13 + 21) / 5 = 10.2$$

+ Thời gian đợi:

$$P1 = 0, P2 = 5, P3 = 12, P4 = 13, P5 = 21$$

$$\Rightarrow \text{Thời gian đợi trung bình: } (0 + 5 + 12 + 13 + 21) / 5 = 10.2$$

+ Thời gian hoàn thành:

$$P1 = 9, P2 = 10, P3 = 19, P4 = 23, P5 = 34$$

$$\Rightarrow \text{Thời gian hoàn thành trung bình: } (9 + 10 + 19 + 23 + 34) / 5 = 19$$

Hình 2: Kết quả khi giải tay ví dụ 1 bằng giải thuật SJF

b. Ví dụ 2:

Process	Arriva Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

i. Kết quả khi chạy code

```
$ g++ *.cpp -o sjf && ./sjf

Enter the number of processes: 5

Enter arrival time of process A: 0
Enter burst time of process A: 12

Enter arrival time of process B: 2
Enter burst time of process B: 7

Enter arrival time of process C: 5
Enter burst time of process C: 8

Enter arrival time of process D: 9
Enter burst time of process D: 3

Enter arrival time of process E: 12
Enter burst time of process E: 6

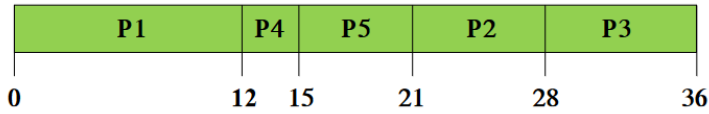
===== SJF Scheduling =====
#P      AT      BT      ST      FT      TAT      WT      RT
A        0      12       0      12      12       0       0
B        2       7      21      28      26      19      19
C        5       8      28      36      31      23      23
D        9       3      12      15       6       3       3
E       12       6      15      21       9       3       3

Average Turnaround Time = 16.80
Average Waiting Time = 9.60
Average Response Time = 9.60
```

Hình 3: Kết quả khi giải ví dụ 2 bằng code giải thuật SJF

ii. Kết quả khi giải tay

■ Giản đồ Gantt



■ Thời gian chờ:

□ $P1 = 0, P2 = 19, P3 = 23, P4 = 3, P5 = 3$

□ Thời gian chờ trung bình: $(0 + 19 + 23 + 3 + 3)/5 = 9.6$

■ Thời gian đáp ứng:

□ $P1 = 0, P2 = 19, P3 = 23, P4 = 3, P5 = 3$

□ Thời gian đáp ứng trung bình: $(0 + 19 + 23 + 3 + 3)/5 = 9.6$

■ Thời gian hoàn thành:

□ $P1 = 12, P2 = 26, P3 = 31, P4 = 6, P5 = 9$

□ Thời gian hoàn thành trung bình: $(12 + 26 + 31 + 6 + 9)/5 = 16.8$

Hình 4: Kết quả khi giải tay ví dụ 2 bằng giải thuật SJF

b. Ví dụ 3:

Process	Arriva Time	Burst Time
P1	0	8
P2	2	19
P3	4	3
P4	5	6
P5	7	12

i. Kết quả khi chạy code

```
$ g++ *.cpp -o sjf && ./sjf

Enter the number of processes: 5

Enter arrival time of process A: 0
Enter burst time of process A: 8

Enter arrival time of process B: 2
Enter burst time of process B: 19

Enter arrival time of process C: 4
Enter burst time of process C: 3

Enter arrival time of process D: 5
Enter burst time of process D: 6

Enter arrival time of process E: 7
Enter burst time of process E: 12

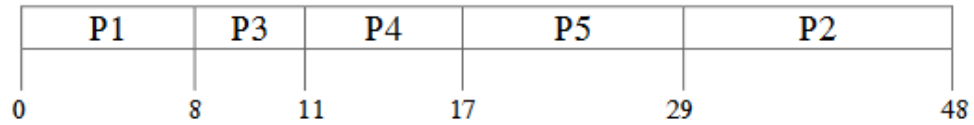
===== SJF Scheduling =====
#P      AT      BT      ST      FT      TAT      WT      RT
A        0        8        0        8        8        0        0
B        2       19       29       48       46       27       27
C        4        3        8       11        7        4        4
D        5        6       11       17       12        6        6
E        7       12       17       29       22       10       10

Average Turnaround Time = 19.00
Average Waiting Time = 9.40
Average Response Time = 9.40
```

Hình 5: Kết quả khi giải ví dụ 3 bằng code giải thuật SJF

ii. Kết quả khi giải tay

+ Giản đồ Gantt:



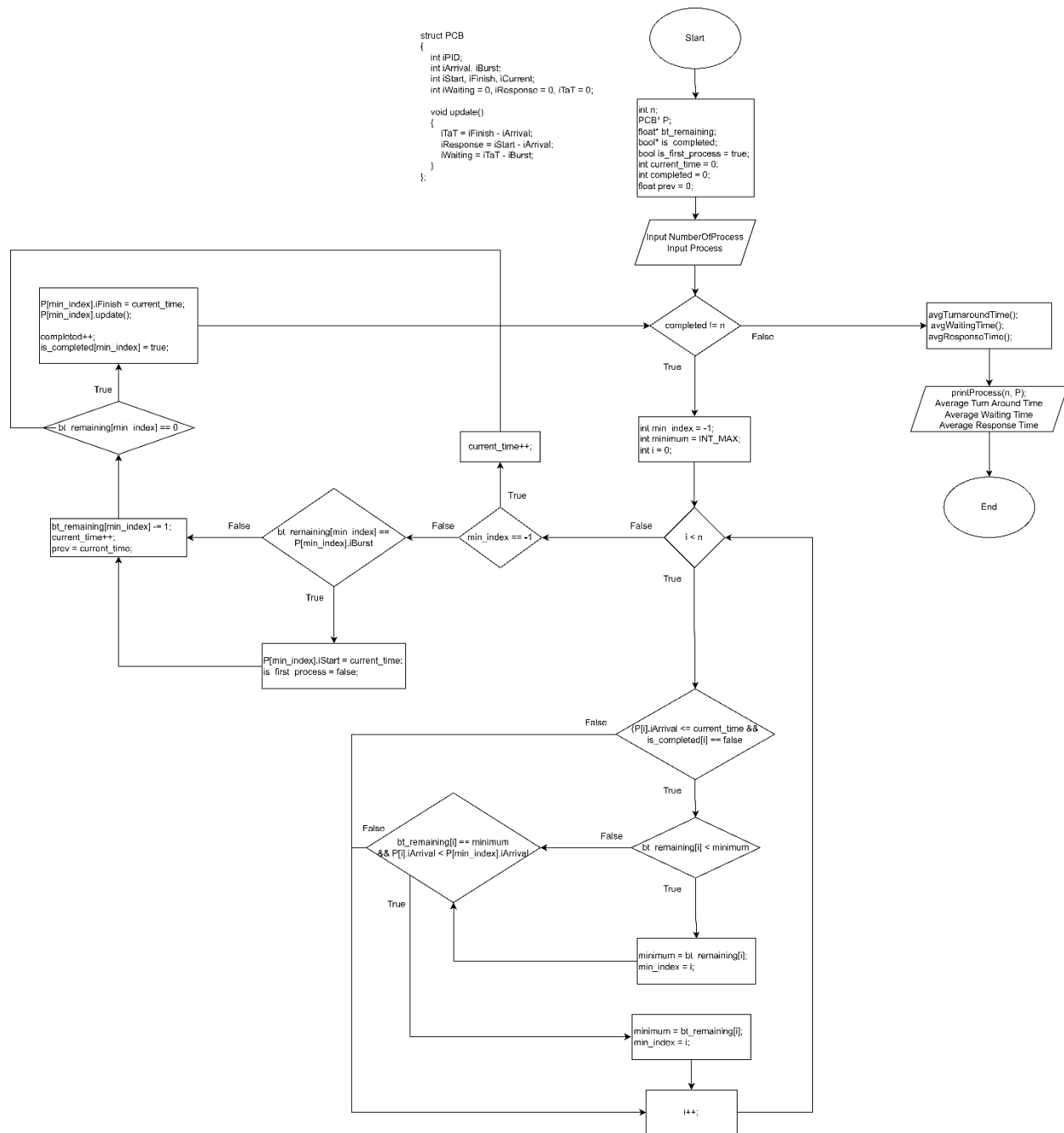
+ Thời gian đáp ứng trung bình là: 9.4

+ Thời gian hoàn thành trung bình: 19.

Hình 6: Kết quả khi giải tay ví dụ 3 bằng giải thuật SJF

2. Giải thuật Shortest-Remaining-Time-First

2.1/ Lưu đồ giải thuật



2.2/ Chạy tay lưu đồ giải thuật

Process	Arrival Time	Burst Time	Waiting	Response	Turnaround
P1	2	6	7	1	13
P2	5	2	0	0	2
P3	1	8	14	14	22
P4	0	3	0	0	3
P5	4	4	2	0	6

Average Turn Around Time = 9.2

Average Waiting Time = 4.6

Average Response Time = 3



2.3/ Hiện thực code

File header

```
MyLibrary.h  SRT.cpp  Source.cpp
ProcessScheduling (Global Scope)

1  #pragma once
2
3  struct PCB
4  {
5      int iPID;
6      int iArrival, iBurst;
7      int iStart, iFinish, iCurrent;
8      int iWaiting = 0, iResponse = 0, iTaT = 0;
9
10     void update()
11     {
12         iTaT = iFinish - iArrival;
13         iResponse = iStart - iArrival;
14         iWaiting = iTaT - iBurst;
15     }
16 };
17
18 // ProcessSupport
19 void inputProcess(int n, PCB P[]);
20 void printProcess(int n, PCB P[]);
21
22 double avgTurnaroundTime(int n, PCB P[]);
23 double avgWaitingTime(int n, PCB P[]);
24 double avgResponseTime(int n, PCB P[]);
25
```

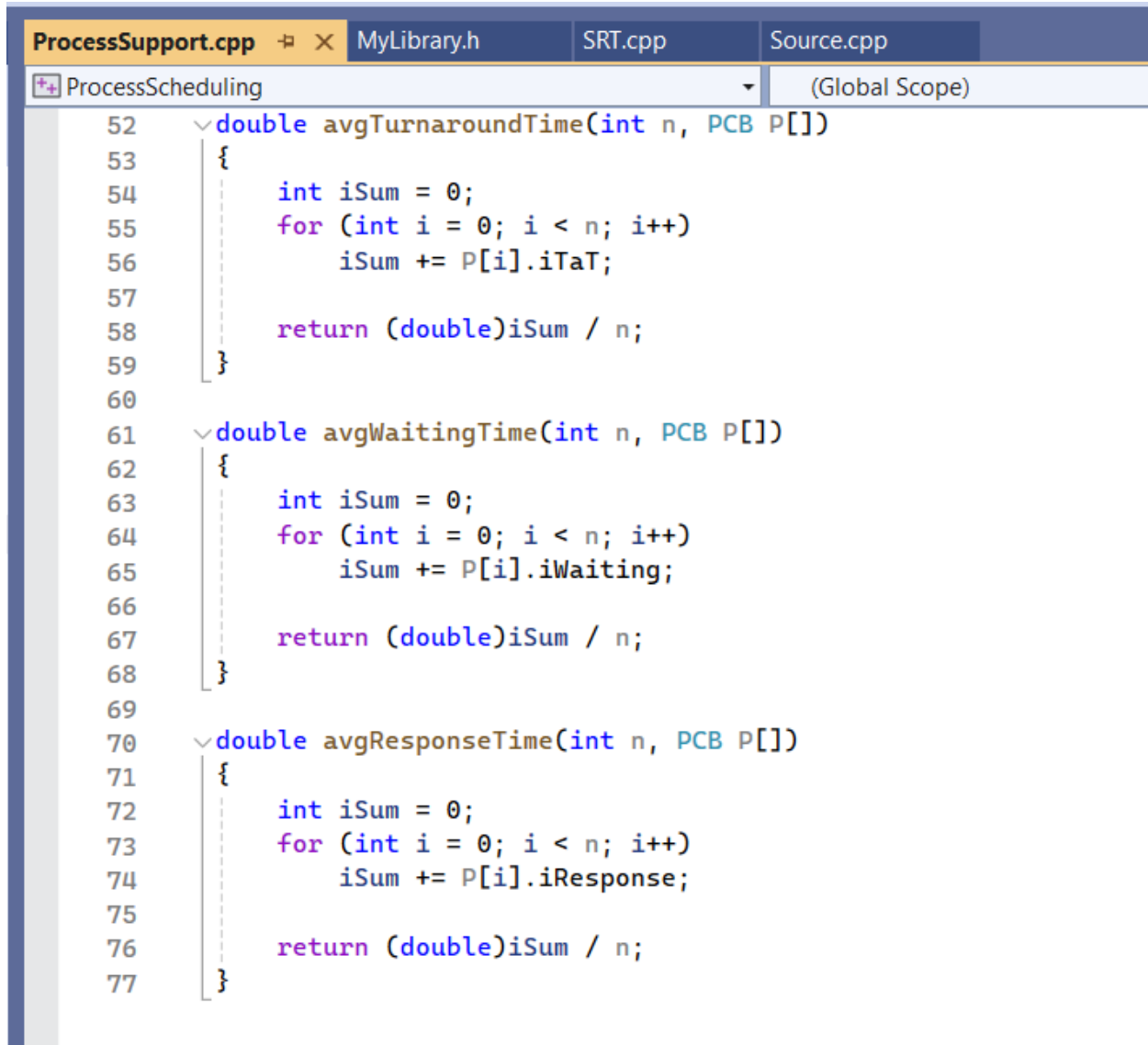
File Cpp

```
ProcessSupport.cpp  MyLibrary.h  SRT.cpp  Source.cpp
ProcessScheduling (Global Scope)

1  #include <iostream>
2  #include <string>
3  #include "MyLibrary.h"
4
5  using namespace std;
6
7  void inputProcess(int n, PCB P[])
8  {
9      for (int i = 0; i < n; i++)
10     {
11         cout << "Enter PID, Arrival Time, Burst Time: ";
12         cin >> P[i].iPID >> P[i].iArrival >> P[i].iBurst;
13     }
14     cout << "-----" << endl;
15 }
16
```

```
ProcessSupport.cpp  MyLibrary.h  SRT.cpp  Source.cpp
ProcessScheduling  (Global Scope)

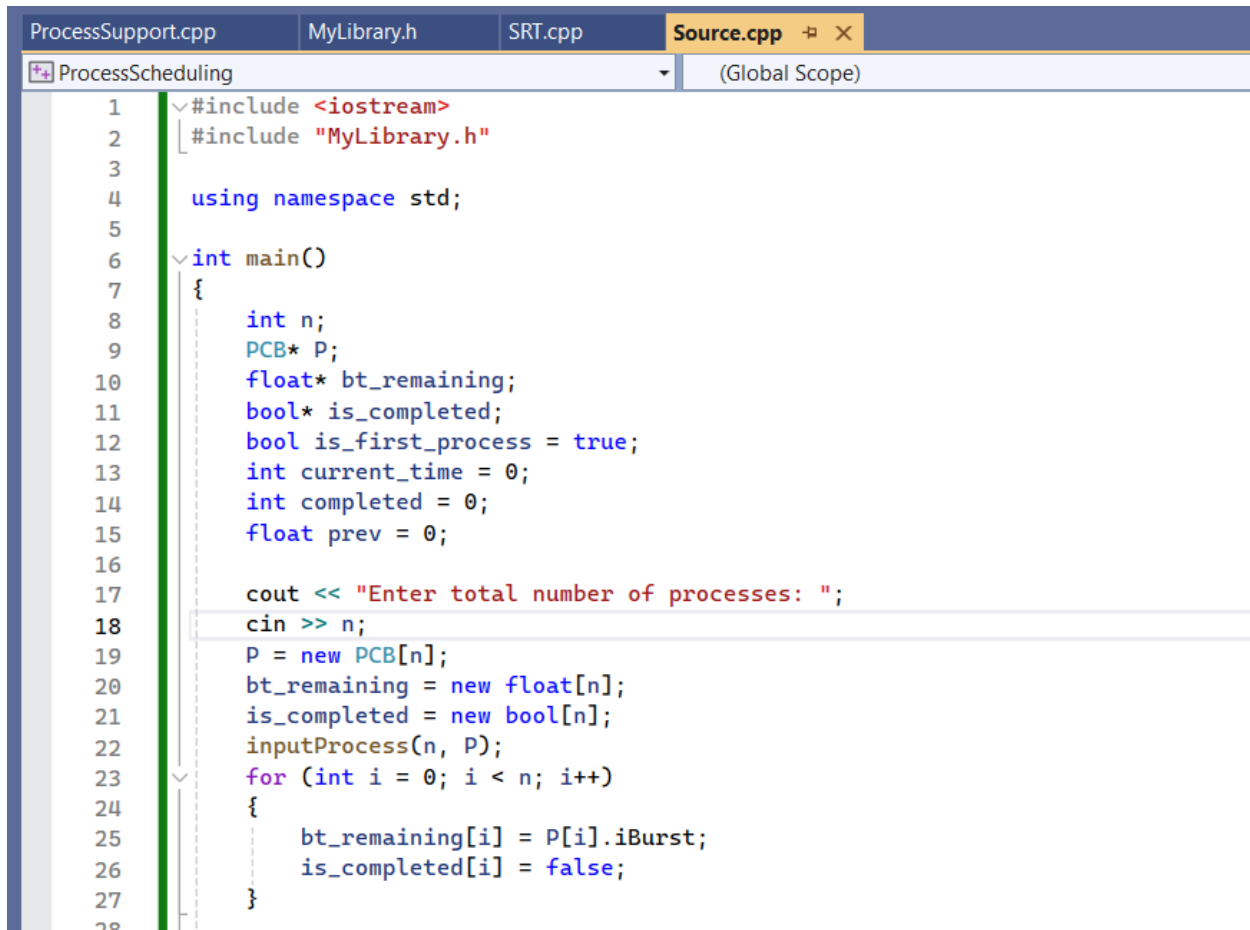
16
17 // print PID, Arrival Time, Burst Time, Waiting Time, Response Time, Turnaround Time
18 void printProcess(int n, PCB P[])
19 {
20     string sP = "PID", sArr = "Arrival Time", sBur = "Burst Time";
21     string sWait = "Waiting Time", sRes = "Response Time", sTaT = "Turnaround Time";
22
23     sP.resize(10, ' ');
24     sArr.resize(20, ' ');
25     sBur.resize(20, ' ');
26     sWait.resize(20, ' ');
27     sRes.resize(20, ' ');
28     sTaT.resize(20, ' ');
29     cout << sP << sArr << sBur << sWait << sRes << sTaT << endl;
30
31     for (int i = 0; i < n; i++)
32     {
33         sP = to_string(P[i].iPID);
34         sArr = to_string(P[i].iArrival);
35         sBur = to_string(P[i].iBurst);
36         sWait = to_string(P[i].iWaiting);
37         sRes = to_string(P[i].iResponse);
38         sTaT = to_string(P[i].iTaT);
39
40         sP.resize(10, ' ');
41         sArr.resize(20, ' ');
42         sBur.resize(20, ' ');
43         sWait.resize(20, ' ');
44         sRes.resize(20, ' ');
45         sTaT.resize(20, ' ');
46
47         cout << sP << sArr << sBur << sWait << sRes << sTaT << endl;
48     }
49     cout << "-----" << endl;
50 };
51
```



```
ProcessSupport.cpp  MyLibrary.h  SRT.cpp  Source.cpp
ProcessScheduling (Global Scope)

52  double avgTurnaroundTime(int n, PCB P[])
53  {
54      int iSum = 0;
55      for (int i = 0; i < n; i++)
56          iSum += P[i].iTAT;
57
58      return (double)iSum / n;
59  }
60
61  double avgWaitingTime(int n, PCB P[])
62  {
63      int iSum = 0;
64      for (int i = 0; i < n; i++)
65          iSum += P[i].iWaiting;
66
67      return (double)iSum / n;
68  }
69
70  double avgResponseTime(int n, PCB P[])
71  {
72      int iSum = 0;
73      for (int i = 0; i < n; i++)
74          iSum += P[i].iResponse;
75
76      return (double)iSum / n;
77  }
```

File main



```
ProcessSupport.cpp | MyLibrary.h | SRT.cpp | Source.cpp [X]
ProcessScheduling (Global Scope)
1  #include <iostream>
2  #include "MyLibrary.h"
3
4  using namespace std;
5
6  int main()
7  {
8      int n;
9      PCB* P;
10     float* bt_remaining;
11     bool* is_completed;
12     bool is_first_process = true;
13     int current_time = 0;
14     int completed = 0;
15     float prev = 0;
16
17     cout << "Enter total number of processes: ";
18     cin >> n;
19     P = new PCB[n];
20     bt_remaining = new float[n];
21     is_completed = new bool[n];
22     inputProcess(n, P);
23     for (int i = 0; i < n; i++)
24     {
25         bt_remaining[i] = P[i].iBurst;
26         is_completed[i] = false;
27     }
28 }
```

```
ProcessSupport.cpp  MyLibrary.h  SRT.cpp  Source.cpp
ProcessScheduling (Global Scope)

28 while (completed != n)
29 {
30     //find process with min. burst time in ready queue at current time
31     int min_index = -1;
32     int minimum = INT_MAX;
33     for (int i = 0; i < n; i++)
34     {
35         if (P[i].iArrival <= current_time && is_completed[i] == false)
36         {
37             if (bt_remaining[i] < minimum)
38             {
39                 minimum = bt_remaining[i];
40                 min_index = i;
41             }
42             if (bt_remaining[i] == minimum)
43                 if (P[i].iArrival < P[min_index].iArrival)
44                 {
45                     minimum = bt_remaining[i];
46                     min_index = i;
47                 }
48         }
49     }
50
51     if (min_index == -1)
52         current_time++;
53     else
54     {
55         if (bt_remaining[min_index] == P[min_index].iBurst)
56         {
57             P[min_index].iStart = current_time;
58             is_first_process = false;
59         }
60         bt_remaining[min_index] -= 1;
61         current_time++;
62         prev = current_time;
63
64         if (bt_remaining[min_index] == 0)
65         {
66             P[min_index].iFinish = current_time;
67             P[min_index].update();
68
69             completed++;
70             is_completed[min_index] = true;
71         }
72     }
73 }
74
75
76 //Output
77 printProcess(n, P);
78
79 cout << "\nAverage Turn Around Time = " << avgTurnaroundTime(n, P);
80 cout << "\nAverage Waiting Time = " << avgWaitingTime(n, P);
81 cout << "\nAverage Response Time = " << avgResponseTime(n, P) << endl;
82
83 delete[]P;
84 delete[]bt_remaining;
85 delete[]is_completed;
86
87 return 0;
88 }
```

2.4/ Chạy code và kiểm chứng

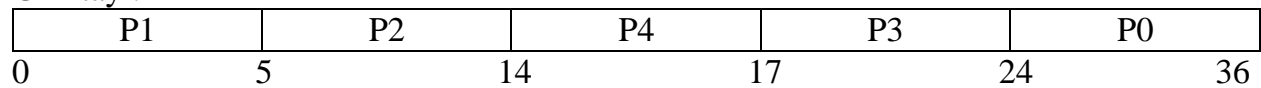
Test 1

```
Microsoft Visual Studio Debug Console
Enter total number of processes: 5
Enter PID, Arrival Time, Burst Time: 0 7 12
Enter PID, Arrival Time, Burst Time: 1 0 5
Enter PID, Arrival Time, Burst Time: 2 3 9
Enter PID, Arrival Time, Burst Time: 4 11 3
Enter PID, Arrival Time, Burst Time: 3 13 7

-----
PID      Arrival Time    Burst Time    Waiting Time    Response Time    Turnaround Time
0         7                12            17              17              29
1         0                5             0               0               5
2         3                9             2               2              11
4        11                3             3               3               6
3        13                7             4               4              11
-----

Average Turn Around Time = 12.4
Average Waiting Time = 5.2
Average Response Time = 5.2
```

Giải tay :



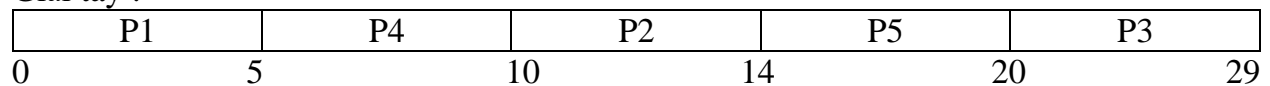
Test 2

```
Microsoft Visual Studio Debug Console
Enter total number of processes: 5
Enter PID, Arrival Time, Burst Time: 1 0 5
Enter PID, Arrival Time, Burst Time: 2 7 4
Enter PID, Arrival Time, Burst Time: 3 11 9
Enter PID, Arrival Time, Burst Time: 4 5 5
Enter PID, Arrival Time, Burst Time: 5 3 6

-----
PID      Arrival Time    Burst Time    Waiting Time    Response Time    Turnaround Time
1         0                5             0               0               5
2         7                4             3               3               7
3        11                9             9               9              18
4         5                5             0               0               5
5         3                6            11              11              17
-----

Average Turn Around Time = 10.4
Average Waiting Time = 4.6
Average Response Time = 4.6
```

Giải tay :



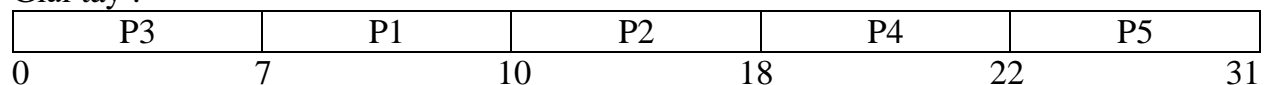
Test 3

```
Microsoft Visual Studio Debug Console
Enter total number of processes: 5
Enter PID, Arrival Time, Burst Time: 1 6 3
Enter PID, Arrival Time, Burst Time: 2 2 8
Enter PID, Arrival Time, Burst Time: 3 0 7
Enter PID, Arrival Time, Burst Time: 4 15 4
Enter PID, Arrival Time, Burst Time: 5 9 9

-----
PID      Arrival Time    Burst Time    Waiting Time    Response Time    Turnaround Time
1         6                3             1               1               4
2         2                8             8               8              16
3         0                7             0               0               7
4        15                4             3               3               7
5         9                9            13              13              22
-----

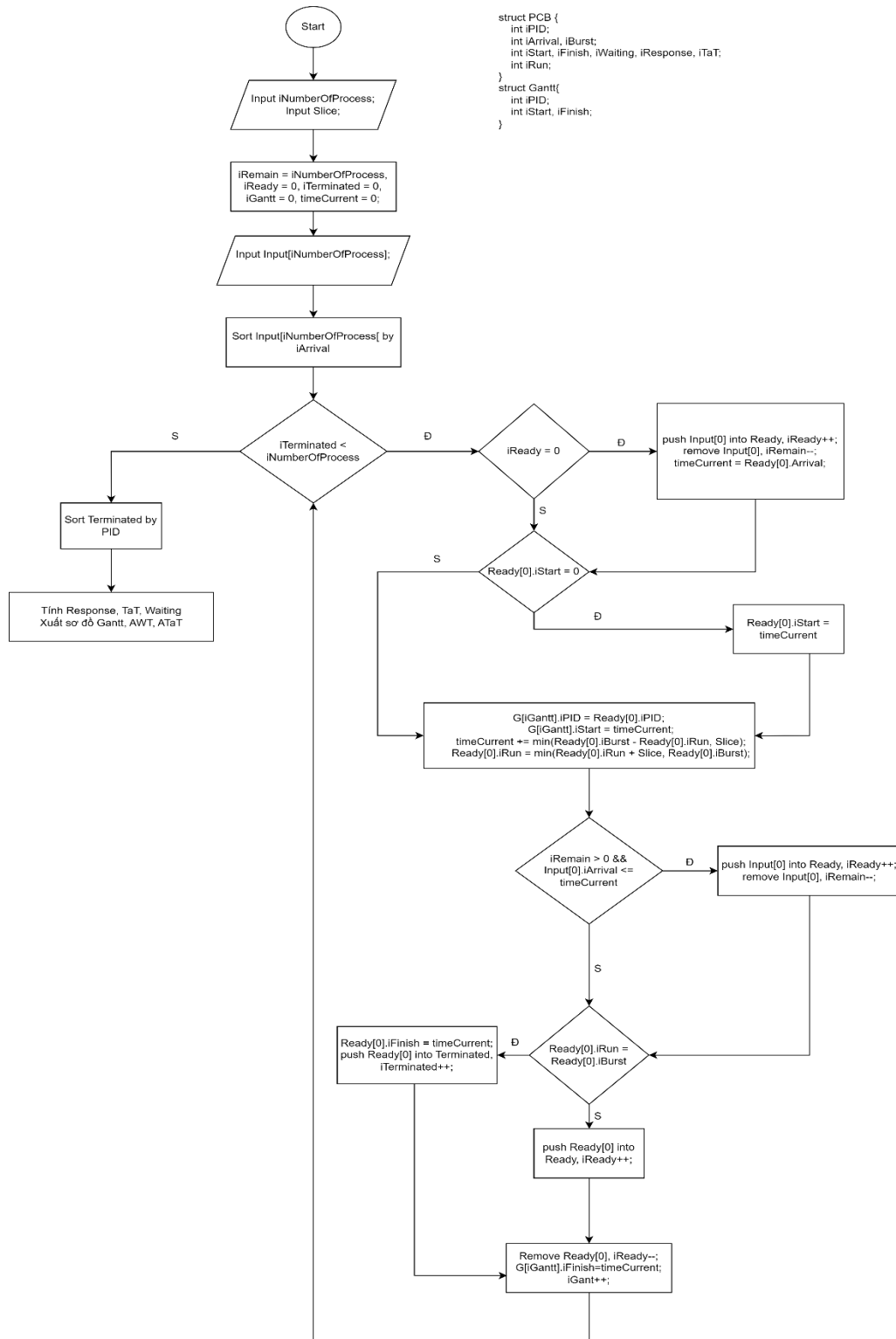
Average Turn Around Time = 11.2
Average Waiting Time = 5
Average Response Time = 5
```

Giải tay :



3. Giải thuật Round Robin

3.1/ Lưu đồ giải thuật

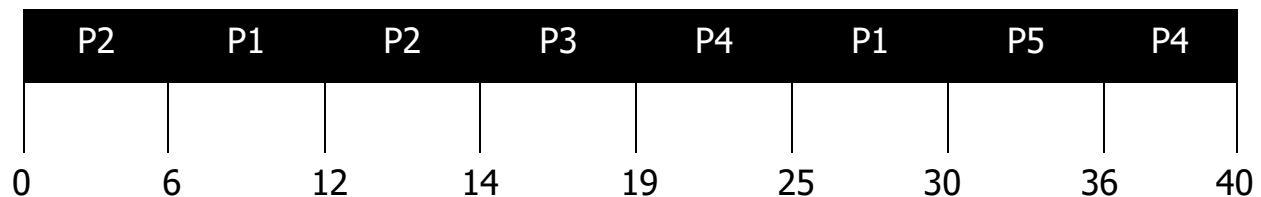


3.2/ Chạy tay lưu đồ

Process	Arrival Time	Burst Time
P1	2	11
P2	0	8
P3	7	5
P4	9	10
P5	13	6

Với quantum time = 6

Kết quả chạy tay:



Thời gian chờ trung bình: $(17+6+7+21+17)/5 = 13.6$

Thời gian hoàn thành trung bình: $(28+14+12+31+23)/5 = 108/5 = 21.6$

3.3/ Code giải thuật

```
#include <stdio.h>
#include <stdlib.h>
#define SORT_BY_ARRIVAL 0
#define SORT_BY_PID 1
#define SORT_BY_BURST 2
#define SORT_BY_START 3

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

typedef struct Gantt{
    int iPID;
    int iStart, iFinish;
```



```
} Gantt;
void inputProcess(int n, PCB P[])
{
    for(int i=0; i<n; i++)
    {
        scanf("%d %d %d", &P[i].iPID, &P[i].iArrival, &P[i].iBurst);
        P[i].iRun = 0;
    }
}
void printProcess(int n, PCB P[])
{
    for(int i=0; i<n; i++)
        printf("%d %d %d\n", P[i].iPID, P[i].iArrival, P[i].iBurst);
}
void exportGanttChart(int n, Gantt P[])
{
    for(int i=0; i<n; i++)
    {
        printf("%d->%d: P%d\n", P[i].iStart, P[i].iFinish, P[i].iPID);
    }
}
void pushProcess(int *n, PCB P[], PCB Q)
{
    P[*n]=Q;
    (*n)++;
}
void removeProcess(int *n, int index, PCB P[])
{
    for(int i=index+1; i < (*n); i++)
        P[i-1]=P[i];
    (*n)--;
}
void swapProcess(PCB *P, PCB *Q)
{
    PCB temp=*P;
    *P=*Q;
    *Q=temp;
}
int partition(PCB P[], int low, int high, int iCriteria)
{
    int pivot=high, i=low - 1;
    for(int j=low; j<=high; j++)
    {
        switch (iCriteria)

```

```
    {
        case 0:
            if(P[j].iArrival < P[pivot].iArrival)
            {
                i++;
                swapProcess(&P[i], &P[j]);
            }
            break;
        case 1:
            if(P[j].iPID < P[pivot].iPID)
            {
                i++;
                swapProcess(&P[i], &P[j]);
            }
            break;
        }
    }
    swapProcess(&P[i+1], &P[high]);
    return (i+1);
}

void quickSort(PCB P[], int low, int high, int iCriteria)
{
    if(low < high)
    {
        int pi=partition(P, low, high, iCriteria);
        quickSort(P, low, pi-1, iCriteria);
        quickSort(P, pi+1, high, iCriteria);
    }
}

void calculateAWT(int n, PCB P[])
{
    int sum=0;
    float avg;
    for(int i=0; i<n; i++)
    {
        sum+=P[i].iWaiting;
    }
    avg=sum/(n*1.0);
    printf("Average waiting time: %f \n", avg);
}

void calculateATaT(int n, PCB P[])
{
    int sum=0;
    float avg;
```

```
        for(int i=0; i<n; i++)
        {
            sum+=P[i].iTAT;
        }
        avg=sum/(n*1.0);
        printf("Average turn around time: %f \n", avg);
    }
void calculateTime(int n, PCB P[])
{
    for(int i = 0; i < n; i++)
    {
        P[i].iResponse = P[i].iStart - P[i].iArrival;
        P[i].iTAT = P[i].iFinish - P[i].iArrival;
        P[i].iWaiting = P[i].iTAT - P[i].iBurst;
    }
}
int min(int x, int y)
{
    return(x < y? x : y);
}
int main()
{
    PCB Input[10];
    PCB Ready[10];
    PCB Terminated[10];
    Gantt G[100];
    int iNumberOfProcess, Slice;
    printf("Input number of Process: ");
    scanf("%d", &iNumberOfProcess);
    printf("Input slice time: ");
    scanf("%d", &Slice);

    int iRemain = iNumberOfProcess, iReady = 0, iTerminated = 0, iGantt = 0,
timeCurrent = 0;
    inputProcess(iNumberOfProcess, Input);
    quickSort(Input, 0, iNumberOfProcess-1, SORT_BY_ARRIVAL);
    while (iTerminated < iNumberOfProcess)
    {
        if(iReady == 0)
        {
            pushProcess(&iReady, Ready, Input[0]);
            removeProcess(&iRemain, 0, Input);
            timeCurrent = Ready[0].iArrival;
        }
    }
}
```

```
if(Ready[0].iStart == 0) Ready[0].iStart = timeCurrent;
G[iGantt].iPID = Ready[0].iPID;
G[iGantt].iStart = timeCurrent;
timeCurrent += min(Ready[0].iBurst - Ready[0].iRun, Slice);
Ready[0].iRun = min(Ready[0].iRun + Slice, Ready[0].iBurst);
while(iRemain > 0 && Input[0].iArrival <= timeCurrent)
{
    pushProcess(&iReady, Ready, Input[0]);
    removeProcess(&iRemain, 0, Input);
}
if(Ready[0].iRun == Ready[0].iBurst)
{
    Ready[0].iFinish = timeCurrent;
    pushProcess(&iTerminated, Terminated, Ready[0]);
}
else pushProcess(&iReady, Ready, Ready[0]);
removeProcess(&iReady, 0, Ready);
G[iGantt].iFinish = timeCurrent;
iGantt++;
}

quickSort(Terminated, 0, iTerminated - 1, SORT_BY_PID);
calculateTime(iTerminated, Terminated);
printf("\n==== RR Scheduling =====\n");
exportGanttChart(iGantt, G);

calculateAWT(iTerminated, Terminated);
calculateATaT(iTerminated, Terminated);
return 0;
}
```

3.4/ Kiểm chứng code

VD 1: ví dụ đã chạy bằng lưu đồ ở trên

Kết quả khi chạy code:

```
● hovantri-23521637@VanTri:~/LAB04$ ./RR
Input number of Process: 5
Input slice time: 6
1 2 11
2 0 8
3 7 5
4 9 10
5 13 6

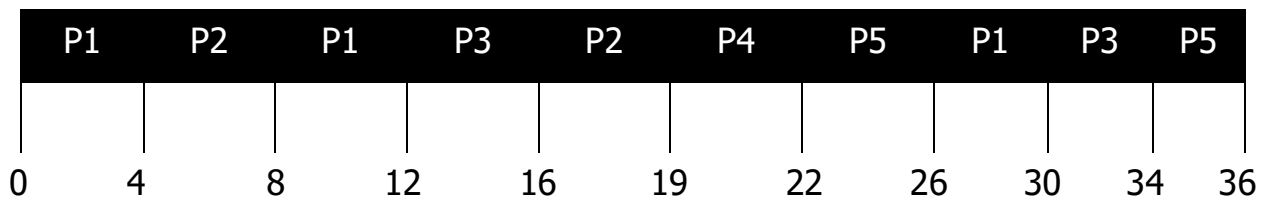
===== RR Scheduling =====
0->6: 2
6->12: 1
12->14: 2
14->19: 3
19->25: 4
25->30: 1
30->36: 5
36->40: 4
Average waiting time: 13.600000
Average turn around time: 21.600000
○ hovantri-23521637@VanTri:~/LAB04$
```

VD 2:

Process	Arrival Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

Với quantum time = 4

Kết quả chạy tay:



Thời gian chờ trung bình: $(18+10+21+10+18)/5 = 15.4$

Thời gian hoàn thành trung bình: $(40+17+29+13+24)/5 = 22.6$

Kết quả chạy code:

```

Average turn around time: 22.600000
● hovantri-23521637@VanTri:~/LAB04$ ./RR
Input number of Process: 5
Input slice time: 4
1    0    12
2    2    7
3    5    8
4    9    3
5   12    6

===== RR Scheduling =====
0->4: 1
4->8: 2
8->12: 1
12->16: 3
16->19: 2
19->22: 4
22->26: 5
26->30: 1
30->34: 3
34->36: 5
Average waiting time: 15.400000
Average turn around time: 22.600000
○ hovantri-23521637@VanTri:~/LAB04$

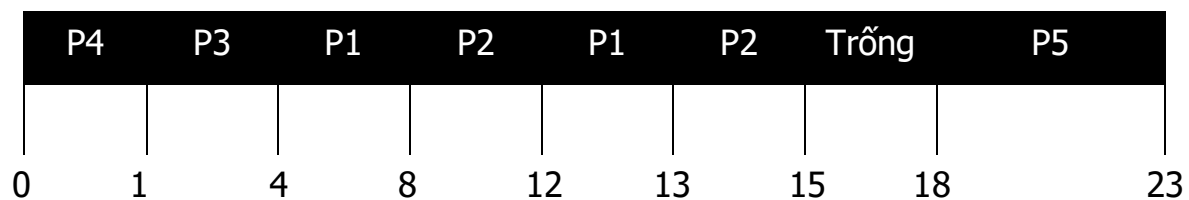
```

VD 3:

Process	Arrival Time	Burst Time
P1	3	5
P2	4	6
P3	1	3
P4	0	1
P5	18	5

Với quantum time = 4

Kết quả chạy tay:



Thời gian chờ trung bình: $(5+5+0+0+0)/5 = 2$

Thời gian hoàn thành trung bình: $(10+11+3+1+5)/5 = 6$

Kết quả chạy code:

```
hovantri-23521637@VanTri:~/LAB04$ ./RR
Input number of Process: 5
Input slice time: 4
1      3      5
2      4      6
3      1      3
4      0      1
5      18     5

===== RR Scheduling =====
0->1: 4
1->4: 3
4->8: 1
8->12: 2
12->13: 1
13->15: 2
18->22: 5
22->23: 5
Average waiting time: 2.000000
Average turn around time: 6.000000
hovantri-23521637@VanTri:~/LAB04$
```