LAB 04 – GROUP 5

CLASS : IT007.019.1 && IT007.019.2

Members:

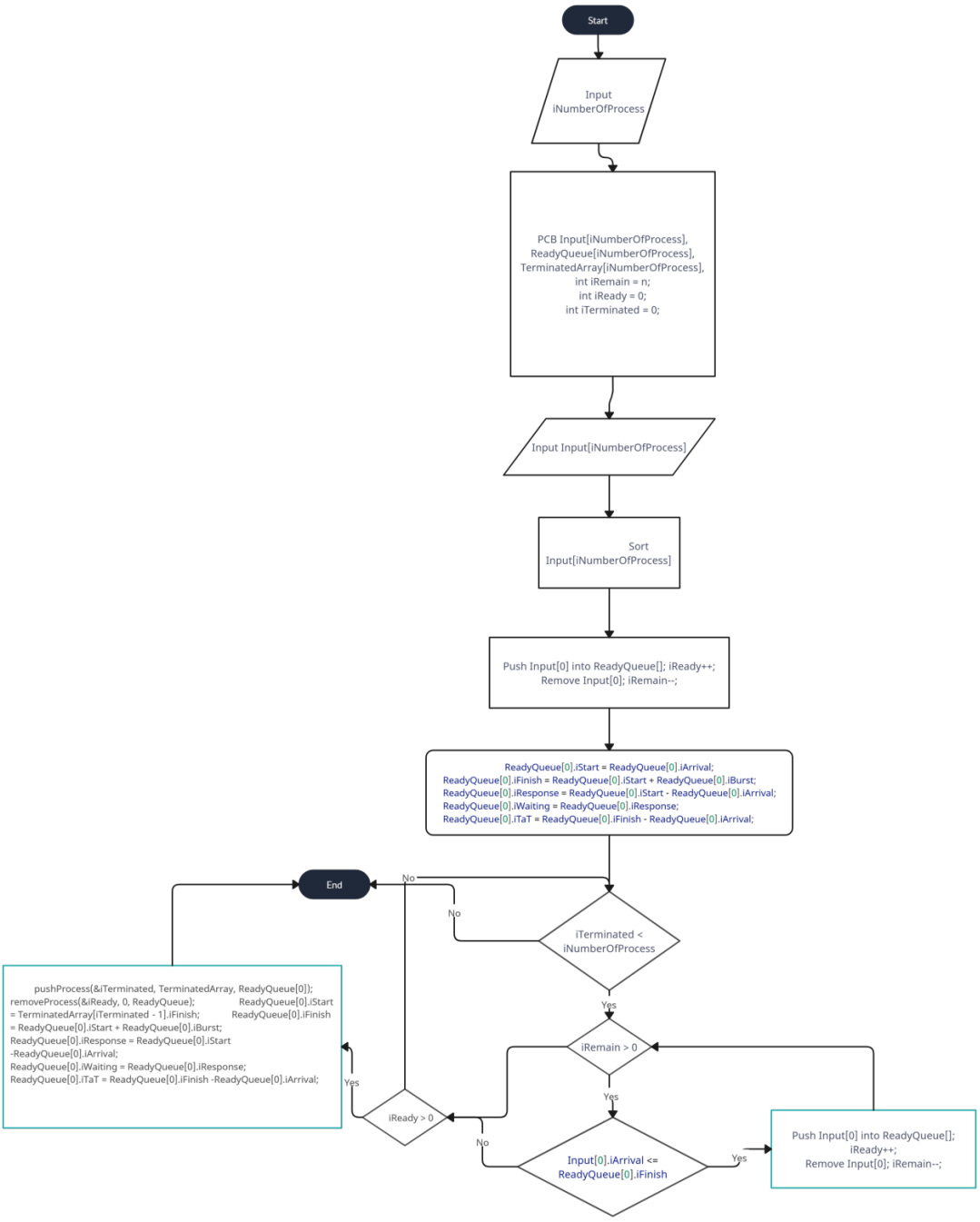
1. Hà Huy Hoàng | 22520460
2. Nguyễn Duy Hoàng | 22520467
3. Nguyễn Hoàng Hiệp | 22520452
4. Nguyễn Hoàng Phúc | 22521129

SUMMARY

Task		Status	Members
Thực hành	FCFS	Hoàn thành	3
	SJF	Hoàn thành	2
	SRTF	Hoàn thành	1
	RR	Hoàn thành	4

A. FCFS:

Lưu đồ giải thuật:



Code

```
#include <stdio.h>
#include <stdlib.h>
#define SORT_BY_ARRIVAL 0
#define SORT_BY_PID 1
```

```c
#define SORT_BY_BURST 2
#define SORT_BY_START 3
typedef struct{
 int iPID;
 int iArrival, iBurst;
 int iStart, iFinish, iWaiting, iResponse, iTaT;
} PCB;
void inputProcess(int n, PCB P[]) {
    for (int i = 0; i < n; ++i) {
        printf("Enter details for Process %d:\n", i + 1);
        P[i].iPID = i + 1;
        printf("Arrival Time: ");
        scanf("%d", &P[i].iArrival);
        printf("Burst Time: ");
        scanf("%d", &P[i].iBurst);
    }
}

void printProcess(int n, PCB P[]) {
  printf("\nPID | Arrival | Burst | Start | Finish | Waiting | Response | TaT\n");
  for (int i = 0; i < n; i++) {
    printf("%-4d | %-7d | %-6d | %-6d | %-6d | %-7d | %-8d | %-4d\n", P[i].iPID,
P[i].iArrival, P[i].iBurst, P[i].iStart, P[i].iFinish, P[i].iWaiting, P[i].iResponse,
P[i].iTaT);
  }
}

void exportGanttChart(int n, PCB P[]) {
  printf("\nGantt Chart:\n\n");
  for (int i = 0; i < n; i++) {
    //printf("0 ");
    printf("P%d | ", P[i].iPID);
  }
  printf("\n");
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < P[i].iBurst; j++) {
      printf("-");
    }
  }
  printf("\n");
  printf("0  | ");
  for (int i = 0; i < n; i++) {
    if(i == n - 1) {
        printf("%d", P[i].iFinish);
        break;
    }
```

```
    printf("%d | ", P[i].iFinish);
  }
  printf("\n");
}

void pushProcess(int *n, PCB P[], PCB Q) {
  P[*n] = Q;
  (*n)++;
}

void removeProcess(int *n, int index, PCB P[]) {
  for (int i = index; i < (*n) - 1; i++) {
    P[i] = P[i + 1];
  }
  (*n)--;
}

int swapProcess(PCB *P, PCB *Q) {
  PCB temp = *P;
  *P = *Q;
  *Q = temp;
  return 0;
}

int partition(PCB P[], int low, int high, int iCriteria) {
  int pivot = P[high].iArrival;
  int i = low - 1;
  for (int j = low; j < high; j++) {
    switch (iCriteria) {
      case SORT_BY_ARRIVAL:
        if (P[j].iArrival <= pivot) {
          i++;
          swapProcess(&P[i], &P[j]);
        }
        break;
      case SORT_BY_PID:
        if (P[j].iPID <= pivot) {
          i++;
          swapProcess(&P[i], &P[j]);
        }
        break;
      case SORT_BY_BURST:
        if (P[j].iBurst <= pivot) {
          i++;
          swapProcess(&P[i], &P[j]);
        }
```

```
        break;
      case SORT_BY_START:
        if (P[j].iStart <= pivot) {
          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 sumWaitingTime = 0;
    for (int i = 0; i < n; ++i) {
        sumWaitingTime += P[i].iWaiting;
    }
    float avgWaitingTime = (float)sumWaitingTime / n;
    printf("Average Waiting Time: %.2f\n", avgWaitingTime);
}

void calculateATaT(int n, PCB P[]) {
    int sumTurnaroundTime = 0;
    for (int i = 0; i < n; ++i) {
        sumTurnaroundTime += P[i].iTaT;
    }
    float avgTurnaroundTime = (float)sumTurnaroundTime / n;
    printf("Average Turnaround Time: %.2f\n", avgTurnaroundTime);
}

int main() {
    PCB Input[10];
    PCB ReadyQueue[10];
    PCB TerminatedArray[10];
    int iNumberOfProcess;

    printf("Please input number of Process: ");
```

```
    scanf("%d", &iNumberOfProcess);
    int iRemain = iNumberOfProcess, iReady = 0, iTerminated = 0;
    inputProcess(iNumberOfProcess, Input);

    quickSort(Input, 0, iNumberOfProcess - 1, SORT_BY_ARRIVAL);
    pushProcess(&iReady, ReadyQueue, Input[0]);
    removeProcess(&iRemain, 0, Input);
    ReadyQueue[0].iStart = ReadyQueue[0].iArrival;
    ReadyQueue[0].iFinish = ReadyQueue[0].iStart + ReadyQueue[0].iBurst;
    ReadyQueue[0].iResponse = ReadyQueue[0].iStart - ReadyQueue[0].iArrival;
    ReadyQueue[0].iWaiting = ReadyQueue[0].iResponse;
    ReadyQueue[0].iTaT = ReadyQueue[0].iFinish - ReadyQueue[0].iArrival;
    printf("\nReady Queue: ");
    printProcess(1, ReadyQueue);
    while (iTerminated < iNumberOfProcess) {
        while (iRemain > 0) {
            if (Input[0].iArrival <= ReadyQueue[0].iFinish) {
                pushProcess(&iReady, ReadyQueue, Input[0]);
                removeProcess(&iRemain, 0, Input);
                continue;
            }
            else
                break;
        }
        if (iReady > 0) {
            pushProcess(&iTerminated, TerminatedArray, ReadyQueue[0]);
            removeProcess(&iReady, 0, ReadyQueue);

            ReadyQueue[0].iStart = TerminatedArray[iTerminated - 1].iFinish;
            ReadyQueue[0].iFinish = ReadyQueue[0].iStart + ReadyQueue[0].iBurst;
            ReadyQueue[0].iResponse = ReadyQueue[0].iStart - ReadyQueue[0].iArrival;
            ReadyQueue[0].iWaiting = ReadyQueue[0].iResponse;
            ReadyQueue[0].iTaT = ReadyQueue[0].iFinish -ReadyQueue[0].iArrival;
        }
    }
    printf("\n===== FCFS Scheduling =====\n");
    exportGanttChart(iTerminated, TerminatedArray);
    quickSort(TerminatedArray, 0, iTerminated - 1, SORT_BY_PID);
    calculateAWT(iTerminated, TerminatedArray);
    calculateATaT(iTerminated, TerminatedArray);

    return 0;
}
```

Test case

P1	P2	P3	P4	P5
0	6	13	21	24	30

Waiting time = (0 + 4 + 8 + 12 + 12)/5 = 7.2

Turnaround time = (6 + 11 +16 + 15 + 18) / 5 = 13.2

Process	Arrival	Burst
p1	0	6
p2	2	7
p3	5	8
p4	9	3
p5	12	6

FCFS.cpp | FCFS.c

FCFS.c > exportGanttChart(int, PCB [])

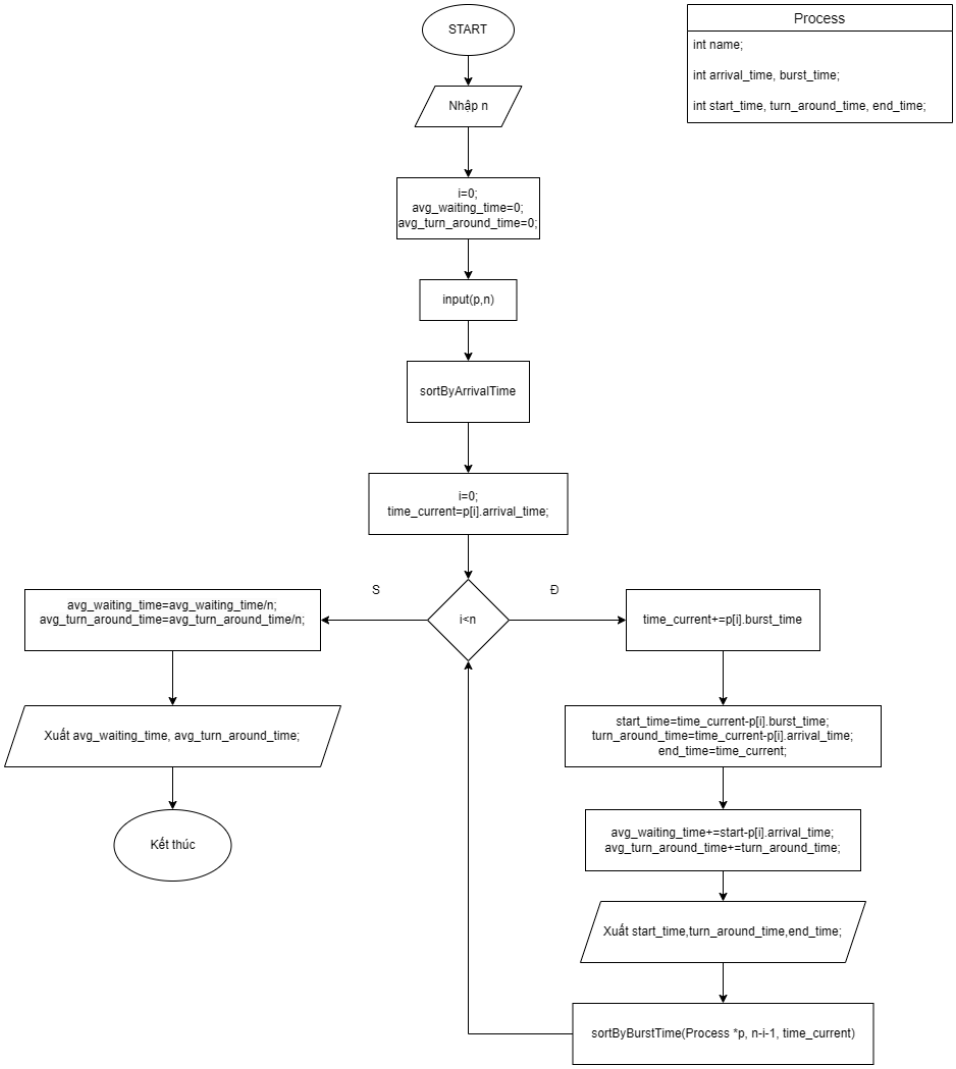
```
41    }
42    printf("\n");
43    printf("0  | ");
44    for (int i = 0; i < n; i++) {
45      if(i == n - 1) {
46          printf("%d", P[i].iFinish);
47          break;
48      }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Please input number of Process: 5
Enter details for Process 1:
Arrival Time: 0
Burst Time: 6
Enter details for Process 2:
Arrival Time: 2
Burst Time: 7
Enter details for Process 3:
Arrival Time: 5
Burst Time: 8
Enter details for Process 4:
Arrival Time: 9
Burst Time: 3
Enter details for Process 5:
Arrival Time: 12
Burst Time: 6

Ready Queue:
PID | Arrival | Burst | Start | Finish | Waiting | Response | TaT
1    | 0       | 6     | 0     | 6      | 0       | 0        | 6

===== FCFS Scheduling =====

Gantt Chart:

P1 | P2 | P3 | P4 | P5 |
------------------------------
0  | 6 | 13 | 21 | 24 | 30
Average Waiting Time: 7.20
Average Turnaround Time: 13.20
nguyenhoanghiep-22520452@HiepNH:~$
```

B. SJF:

Lưu đồ giải thuật



Code

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

// Cấu trúc biểu diễn tiến trình
typedef struct {
    int iPID;
    int arrival_time;
    int burst_time;
} Process;

// Hàm so sánh cho sắp xếp theo thời gian đến
int compareByArrivalTime(const void *a, const void *b) {
```

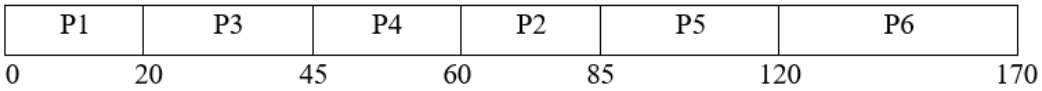
```
    return ((Process*)a)->arrival_time - ((Process*)b)->arrival_time;
}

// Hàm so sánh cho sắp xếp theo burst time
int compareByBurstTime(const void *a, const void *b) {
    return ((Process*)a)->burst_time - ((Process*)b)->burst_time;
}

int main() {
    int n;
    printf("Nhập số lượng tiến trình: ");
    scanf("%d", &n);

    Process processes[n];

    // Nhập thông tin cho từng tiến trình
    for (int i = 0; i < n; i++) {
        processes[i].iPID = i + 1;
        printf("Nhập thời gian đến của tiến trình %d: ", i + 1);
        scanf("%d", &processes[i].arrival_time);
        printf("Nhập thời gian thực hiện của tiến trình %d: ", i + 1);
        scanf("%d", &processes[i].burst_time);
    }

    // Sắp xếp tiến trình theo thời gian đến tăng dần
    qsort(processes, n, sizeof(Process), compareByArrivalTime);

    int time_current = 0;
    float avg_waiting_time = 0;
    float avg_turnaround_time = 0;

    for (int i = 0; i < n;) {
        // Kiểm tra xem có tiến trình nào đến trong khoảng thời gian hiện tại không
        if (processes[i].arrival_time <= time_current) {
            int queue_size = 0; // Kích thước của ready queue
            int j = i;
            // Tính kích thước của ready queue
            while (j < n && processes[j].arrival_time <= time_current) {
                queue_size++;
                j++;
            }

            // Sắp xếp các tiến trình trong ready queue theo burst time
            qsort(processes + i, queue_size, sizeof(Process), compareByBurstTime);

            // Chọn tiến trình có burst time nhỏ nhất để thực hiện
```

```
            Process selected_process = processes[i];

            // Thực hiện tiến trình
            printf("Thực hiện tiến trình %d từ thời điểm %d đến %d\n",
                   selected_process.iPID, time_current, time_current +
selected_process.burst_time);

            // Cập nhật thời gian hiện tại
            time_current += selected_process.burst_time;

            // Tính thời gian chờ đợi cho tiến trình
            int waiting_time = time_current - selected_process.arrival_time -
selected_process.burst_time;

            // Cộng vào avg_waiting_time
            avg_waiting_time += waiting_time;

            // Tính thời gian quay vòng cho tiến trình
            int turnaround_time = time_current - selected_process.arrival_time;

            // Cộng vào avg_turnaround_time
            avg_turnaround_time += turnaround_time;

            // In thời gian chờ và thời gian hoàn thành của tiến trình
            printf("Tiến trình %d: Thời gian chờ: %d, Thời gian hoàn thành: %d\n",
                   selected_process.iPID, waiting_time, turnaround_time);

            // Tiến trình đã được thực hiện, tăng biến đếm
            i++;
        } else {
            // Chưa có tiến trình nào đến, tăng thời gian hiện tại
            time_current++;
        }
    }

    // Tính trung bình thời gian chờ đợi và thời gian quay vòng
    avg_waiting_time /= n;
    avg_turnaround_time /= n;

    // In kết quả
    printf("Thời gian chờ đợi trung bình: %.2f\n", avg_waiting_time);
    printf("Thời gian quay vòng trung bình: %.2f\n", avg_turnaround_time);

    return 0;
}
```

Test case 1:

Process	Arrival Time	Burst Time	Priority
P1	0	20	20
P2	25	25	30
P3	20	25	15
P4	35	15	35
P5	10	35	5
P6	15	50	10

- SJF:

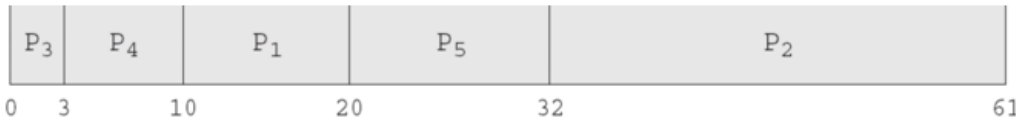


- Thời gian chờ trung bình: (0+35+0+10+75+105)/6 = 37.5
- Thời gian đáp ứng trung bình: (0+35+0+10+75+105)/6 = 37.5
- Thời gian hoàn thành trung bình: (20+60+25+25+110+155)/6= 65.83

```
nguyenduyhoang-22520467@HOANGND04:~$ gcc sjf.c -o test
nguyenduyhoang-22520467@HOANGND04:~$ ./test
Nhập số lượng tiến trình: 6
Nhập thời gian đến của tiến trình 1: 0
Nhập thời gian thực hiện của tiến trình 1: 20
Nhập thời gian đến của tiến trình 2: 25
Nhập thời gian thực hiện của tiến trình 2: 25
Nhập thời gian đến của tiến trình 3: 20
Nhập thời gian thực hiện của tiến trình 3: 25
Nhập thời gian đến của tiến trình 4: 35
Nhập thời gian thực hiện của tiến trình 4: 15
Nhập thời gian đến của tiến trình 5: 10
Nhập thời gian thực hiện của tiến trình 5: 35
Nhập thời gian đến của tiến trình 6: 15
Nhập thời gian thực hiện của tiến trình 6: 50
Thực hiện tiến trình 1 từ thời điểm 0 đến 20
Tiến trình 1: Thời gian chờ: 0, Thời gian hoàn thành: 20
Thực hiện tiến trình 3 từ thời điểm 20 đến 45
Tiến trình 3: Thời gian chờ: 0, Thời gian hoàn thành: 25
Thực hiện tiến trình 4 từ thời điểm 45 đến 60
Tiến trình 4: Thời gian chờ: 10, Thời gian hoàn thành: 25
Thực hiện tiến trình 2 từ thời điểm 60 đến 85
Tiến trình 2: Thời gian chờ: 35, Thời gian hoàn thành: 60
Thực hiện tiến trình 5 từ thời điểm 85 đến 120
Tiến trình 5: Thời gian chờ: 75, Thời gian hoàn thành: 110
Thực hiện tiến trình 6 từ thời điểm 120 đến 170
Tiến trình 6: Thời gian chờ: 105, Thời gian hoàn thành: 155
Thời gian chờ đợi trung bình: 37.50
Thời gian quay vòng trung bình: 65.83_
```

Test case 2

Process	Arrival Time	CPU Burst Time
P1	0	8
P2	2	19
P3	4	3
P4	5	6
P5	7	10



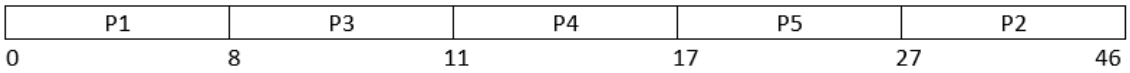
Thời gian đợi trung bình: (10 + 32 + 0 + 3 + 20)/5 = 13

Thời gian lưu lại trong hệ thống trung bình: (20 + 61 + 3 + 10 + 32)/5 = 25.2

```
nguyenduyhoang-22520467@HOANGND04:~$ ./test
Nhập số lượng tiến trình: 5
Nhập thời gian đến của tiến trình 1: 0
Nhập thời gian thực hiện của tiến trình 1: 8
Nhập thời gian đến của tiến trình 2: 2
Nhập thời gian thực hiện của tiến trình 2: 19
Nhập thời gian đến của tiến trình 3: 4
Nhập thời gian thực hiện của tiến trình 3: 3
Nhập thời gian đến của tiến trình 4: 5
Nhập thời gian thực hiện của tiến trình 4: 6
Nhập thời gian đến của tiến trình 5: 7
Nhập thời gian thực hiện của tiến trình 5: 10
Thực hiện tiến trình 1 từ thời điểm 0 đến 8
Tiến trình 1: Thời gian chờ: 0, Thời gian hoàn thành: 8
Thực hiện tiến trình 3 từ thời điểm 8 đến 11
Tiến trình 3: Thời gian chờ: 4, Thời gian hoàn thành: 7
Thực hiện tiến trình 4 từ thời điểm 11 đến 17
Tiến trình 4: Thời gian chờ: 6, Thời gian hoàn thành: 12
Thực hiện tiến trình 5 từ thời điểm 17 đến 27
Tiến trình 5: Thời gian chờ: 10, Thời gian hoàn thành: 20
Thực hiện tiến trình 2 từ thời điểm 27 đến 46
Tiến trình 2: Thời gian chờ: 25, Thời gian hoàn thành: 44
Thời gian chờ đợi trung bình: 9.00
Thời gian quay vòng trung bình: 18.20
```

Test case 3

Process	Arrival Time	Burst Time
P1	0	10
P2	0	29
P3	0	3
P4	0	7
P5	0	12



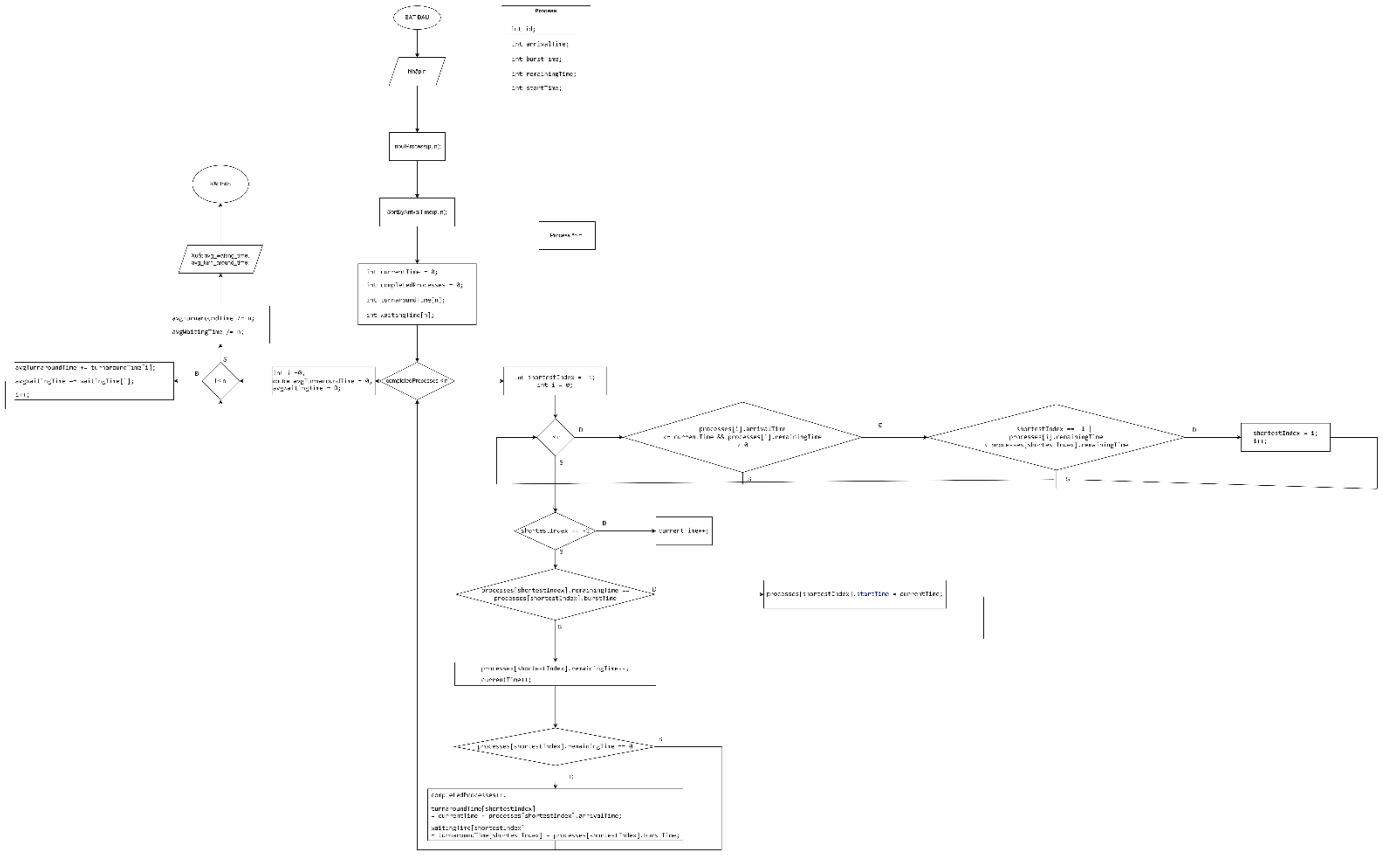
THỜI GIAN ĐỢI TRUNG BÌNH : (0 + 25 + 4 + 6 + 10) / 5 = 9

THỜI GIAN HOÀN THÀNH TRUNG BÌNH: (8 + 44 + 7 + 12 + 20)/5 = 18.2

```
nguyenduyhoang-22520467@HOANGND04:~$ ./test
Nhập số lượng tiến trình: 5
Nhập thời gian đến của tiến trình 1: 0
Nhập thời gian thực hiện của tiến trình 1: 10
Nhập thời gian đến của tiến trình 2: 0
Nhập thời gian thực hiện của tiến trình 2: 29
Nhập thời gian đến của tiến trình 3: 0
Nhập thời gian thực hiện của tiến trình 3: 3
Nhập thời gian đến của tiến trình 4: 0
Nhập thời gian thực hiện của tiến trình 4: 7
Nhập thời gian đến của tiến trình 5: 0
Nhập thời gian thực hiện của tiến trình 5: 12
Thực hiện tiến trình 3 từ thời điểm 0 đến 3
Tiến trình 3: Thời gian chờ: 0, Thời gian hoàn thành: 3
Thực hiện tiến trình 4 từ thời điểm 3 đến 10
Tiến trình 4: Thời gian chờ: 3, Thời gian hoàn thành: 10
Thực hiện tiến trình 1 từ thời điểm 10 đến 20
Tiến trình 1: Thời gian chờ: 10, Thời gian hoàn thành: 20
Thực hiện tiến trình 5 từ thời điểm 20 đến 32
Tiến trình 5: Thời gian chờ: 20, Thời gian hoàn thành: 32
Thực hiện tiến trình 2 từ thời điểm 32 đến 61
Tiến trình 2: Thời gian chờ: 32, Thời gian hoàn thành: 61
Thời gian chờ đợi trung bình: 13.00
Thời gian quay vòng trung bình: 25.20
nguyenduyhoang-22520467@HOANGND04:~$
```

C. SRTF:

Lưu đồ giải thuật



Code

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

struct Process {
    int id;
    int arrivalTime;
    int burstTime;
    int remainingTime;
    int startTime;  // Thời gian bắt đầu thực hiện
};

void SRTF(struct Process *processes, int n) {
    // Sắp xếp các tiến trình theo thời gian đến
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (processes[j].arrivalTime > processes[j + 1].arrivalTime) {
```

```
                // Hoán đổi vị trí nếu thời gian đến của tiến trình sau lớn hơn tiến
trình trước
                struct Process temp = processes[j];
                processes[j] = processes[j + 1];
                processes[j + 1] = temp;
            }
        }
    }

    int currentTime = 0;
    int completedProcesses = 0;
    int turnaroundTime[n];
    int waitingTime[n];

    while (completedProcesses < n) {
        int shortestIndex = -1;
        for (int i = 0; i < n; ++i) {
            if (processes[i].arrivalTime <= currentTime && processes[i].remainingTime
> 0) {
                if (shortestIndex == -1 || processes[i].remainingTime <
processes[shortestIndex].remainingTime) {
                    shortestIndex = i;
                }
            }
        }

        if (shortestIndex == -1) {
            currentTime++;
        } else {
            if (processes[shortestIndex].remainingTime ==
processes[shortestIndex].burstTime) {
                processes[shortestIndex].startTime = currentTime;
            }

            processes[shortestIndex].remainingTime--;
            currentTime++;

            if (processes[shortestIndex].remainingTime == 0) {
                completedProcesses++;
                turnaroundTime[shortestIndex] = currentTime -
processes[shortestIndex].arrivalTime;
                waitingTime[shortestIndex] = turnaroundTime[shortestIndex] -
processes[shortestIndex].burstTime;
            }
        }
    }
```

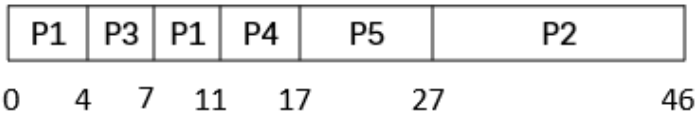
```
    // In kết quả
    printf("Process\tTurnaround Time\tWaiting Time\n");
    for (int i = 0; i < n; ++i) {
        printf("%d\t%d\t\t%d\n", processes[i].id, turnaroundTime[i], waitingTime[i]);
    }

    // Tính toán và in thời gian chờ trung bình và thời gian hoàn thành trung bình
    double avgTurnaroundTime = 0, avgWaitingTime = 0;
    for (int i = 0; i < n; ++i) {
        avgTurnaroundTime += turnaroundTime[i];
        avgWaitingTime += waitingTime[i];
    }
    avgTurnaroundTime /= n;
    avgWaitingTime /= n;
    printf("\nAverage Turnaround Time: %.2f\n", avgTurnaroundTime);
    printf("Average Waiting Time: %.2f\n", avgWaitingTime);
}

int main() {
    // Nhập số lượng tiến trình
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("----------------------------------------\n");

    struct Process *processes = (struct Process *)malloc(n * sizeof(struct Process));

    // Nhập thông tin cho từng tiến trình
    for (int i = 0; i < n; ++i) {
        printf("Enter process id for process: ");
        scanf("%d", &processes[i].id);
        printf("Enter arrival time for process: ");
        scanf("%d", &processes[i].arrivalTime);
        printf("Enter burst time for process: ");
        scanf("%d", &processes[i].burstTime);
        processes[i].remainingTime = processes[i].burstTime;
        printf("----------------------------------------\n");
    }

    // Chạy giải thuật SRTF
    SRTF(processes, n);

    free(processes);
    return 0;
}
```

Test case 1 :

Process	Arrival Time	CPU Burst Time
P1	0	8
P2	2	19
P3	4	3
P4	5	6
P5	7	10

- *Giải tay*

Giản đồ Gantt



Thời gian chờ trung bình : (3+ 25 + 0 + 6 + 10)/5 = 8,8

Thời gian lưu lại trong hệ thống: (11 + 44 + 3 + 12 + 20)/5 = 18

- *Chạy code*

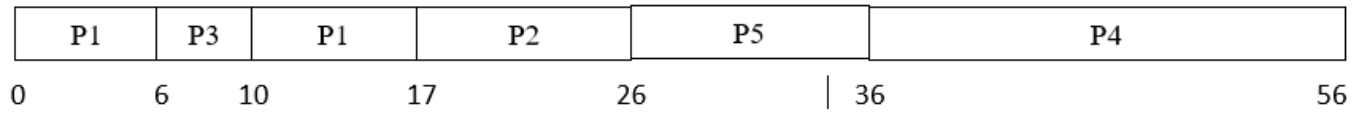
```
Enter the number of processes: 5
----------------------------------------
Enter process id for process: 1
Enter arrival time for process: 0
Enter burst time for process: 8
----------------------------------------
Enter process id for process: 2
Enter arrival time for process: 2
Enter burst time for process: 19
----------------------------------------
Enter process id for process: 3
Enter arrival time for process: 4
Enter burst time for process: 3
----------------------------------------
Enter process id for process: 4
Enter arrival time for process: 5
Enter burst time for process: 6
----------------------------------------
Enter process id for process: 5
Enter arrival time for process: 7
Enter burst time for process: 10
----------------------------------------
Process Turnaround Time Waiting Time
1       11              3
2       44              25
3       3               0
4       12              6
5       20              10

Average Turnaround Time: 18.00
Average Waiting Time: 8.80
hoang-22520460@DESKTOP-EMKNK1L:~/Lab4$
```

Test case 2:

Process	Arrival Time	Burst Time
P1	0	13
P2	4	9
P3	6	4
P4	7	20
P5	12	10

- *Giải tay*

Giản đồ Gantt:



Thời gian đợi trung bình: (4 + 13 + 0 + 29 + 14)/5 = 12

Thời gian hoàn thành trung bình: (17 + 22 + 4 + 49 + 24)/5 = 23.2

- *Chạy code*

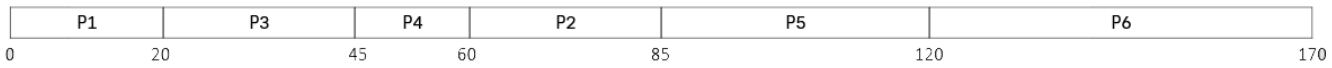
```
hoang-22520460@DESKTOP-EMKNK1L:~/Lab4$ ./a
Enter the number of processes: 5
--------------------------------------
Enter process id for process: 1
Enter arrival time for process: 0
Enter burst time for process: 13
--------------------------------------
Enter process id for process: 2
Enter arrival time for process: 4
Enter burst time for process: 9
--------------------------------------
Enter process id for process: 3
Enter arrival time for process: 6
Enter burst time for process: 4
--------------------------------------
Enter process id for process: 4
Enter arrival time for process: 7
Enter burst time for process: 20
--------------------------------------
Enter process id for process: 5
Enter arrival time for process: 12
Enter burst time for process: 10
--------------------------------------
Process Turnaround Time Waiting Time
1       17              4
2       22              13
3       4               0
4       49              29
5       24              14

Average Turnaround Time: 23.20
Average Waiting Time: 12.00
```

Test case 3:

Process	Arrival Time	Burst Time
P1	0	20
P2	25	25
P3	20	25
P4	35	15
P5	10	35
P6	15	50

- *Giải tay*

Giản đồ Gantt



Thời gian đợi trung bình: (0 + 35 + 0 + 10 + 75 + 105)/6 = 37,5

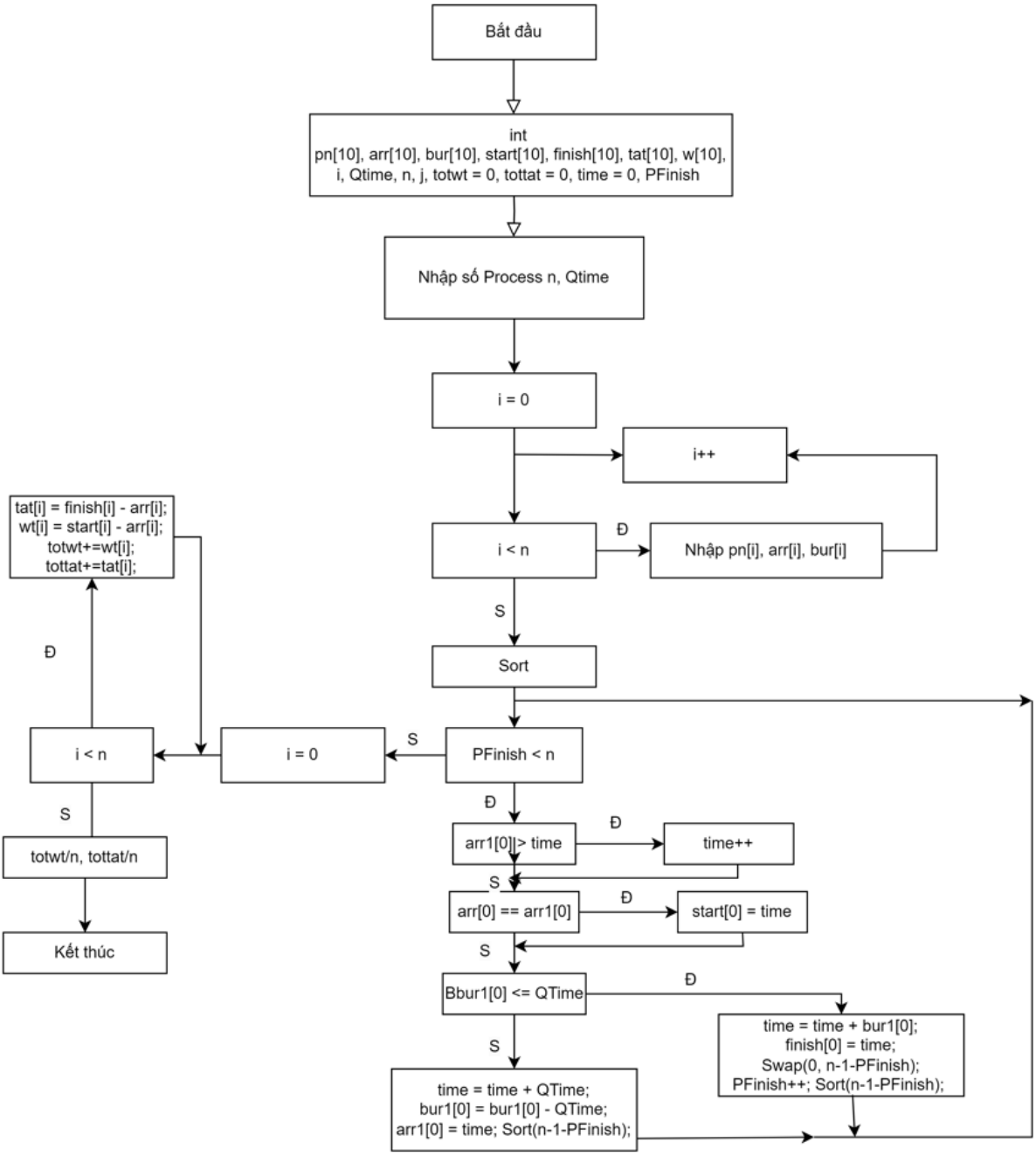
Thời gian hoàn thành trung bình: (20 + 60 + 25 + 25 + 110 + 155)/6 = 65,833

- *Chạy code*

```
Enter the number of processes: 6
----------------------------------------
Enter process id for process: 1
Enter arrival time for process: 0
Enter burst time for process: 20
----------------------------------------
Enter process id for process: 2
Enter arrival time for process: 25
Enter burst time for process: 25
----------------------------------------
Enter process id for process: 3
Enter arrival time for process: 20
Enter burst time for process: 25
----------------------------------------
Enter process id for process: 4
Enter arrival time for process: 35
Enter burst time for process: 15
----------------------------------------
Enter process id for process: 5
Enter arrival time for process: 10
Enter burst time for process: 35
----------------------------------------
Enter process id for process: 6
Enter arrival time for process: 15
Enter burst time for process: 50
----------------------------------------
Process Turnaround Time Waiting Time
1       20              0
5       110             75
6       155             105
3       25              0
2       60              35
4       25              10

Average Turnaround Time: 65.83
Average Waiting Time: 37.50
hoang-22520460@DESKTOP-EMKNK1L:~/Lab4$
```

D. RR:

Lưu đồ giải thuật



Code

```
#include<stdio.h>
int pn[10], arr[10], arr1[10], bur[10], bur1[10], start[10], finish[10],
tat[10], wt[10];
void hv(int a, int b)
{
```

```
    int tmp = pn[a];
    pn[a] = pn[b];
    pn[b] = tmp;
    tmp = arr[a];
    arr[a] = arr[b];
    arr[b] = tmp;
    tmp = arr1[a];
    arr1[a] = arr1[b];
    arr1[b] = tmp;
    tmp = bur[a];
    bur[a] = bur[b];
    bur[b] = tmp;
    tmp = bur1[a];
    bur1[a] = bur1[b];
    bur1[b] = tmp;
    tmp = start[a];
    start[a] = start[b];
    start[b] = tmp;
    tmp = finish[a];
    finish[a] = finish[b];
    finish[b] = tmp;
}
void Sort(int b)
{
    int i, j;
    for(i = 0; i < b; i++)
        for(j = i + 1; j <= b; j++)
            if(arr1[i] > arr1[j])
                hv(i, j);
            else if((arr1[i] == arr1[j]) && (bur1[i] > bur1[j]))
                hv(i, j);
    for(i = b; i >= 1; i--)
        if((arr1[i] == arr1[i-1]) && (arr[i] == arr1[i]) && (arr[i-1] == arr1[i
- 1]))
            hv(i, i - 1);
}
int main()
{
    int totwt = 0, tottat = 0, i, n, j, QTime;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter QTime: ");
    scanf("%d", &QTime);
    for (i = 0; i < n; i++)
    {
        printf("Enter the Process Name, Arrival Time, Burst Time: ");
```

```
        scanf("%d%d%d", &pn[i], &arr[i], &bur[i]);
        arr1[i] = arr[i];
        bur1[i] = bur[i];
    }
    for(i = 0; i < n -1; i++)
        for(j = i + 1; j < n; j++)
        {
            if(arr[i] > arr[j])
                hv(i, j);
            else if((arr[i] == arr[j]) && (bur[i] < bur[j]))
                    hv(i, j);
        }
    int PFinish = 0, time = 0;
    while(PFinish < n)
    {
        while(arr1[0] > time)
            time++;
        if(arr[0] == arr1[0])
            start[0] = time;
        if(bur1[0] <= QTime)
        {
            time = time + bur1[0];
            finish[0] = time;
            hv(0, n-1-PFinish);
            PFinish++;
            Sort(n-1-PFinish);
        }
        else
        {
            time = time + QTime;
            bur1[0] = bur1[0] - QTime;
            arr1[0] = time;
            Sort(n-1-PFinish);
        }
    }

    printf("\nPName Arrtime Burtime Start Tat Finish Wt");
    for(i = 0; i < n; i++)
    {
        tat[i] = finish[i] - arr[i];
        wt[i] = tat[i] - bur[i];
        printf("\nP%d\t%6d\t%6d\t%6d\t%6d\t%6d\t%6d", pn[i], arr[i], bur[i],
start[i], tat[i], finish[i], wt[i]);
        totwt += wt[i];
        tottat += tat[i];
    }
```

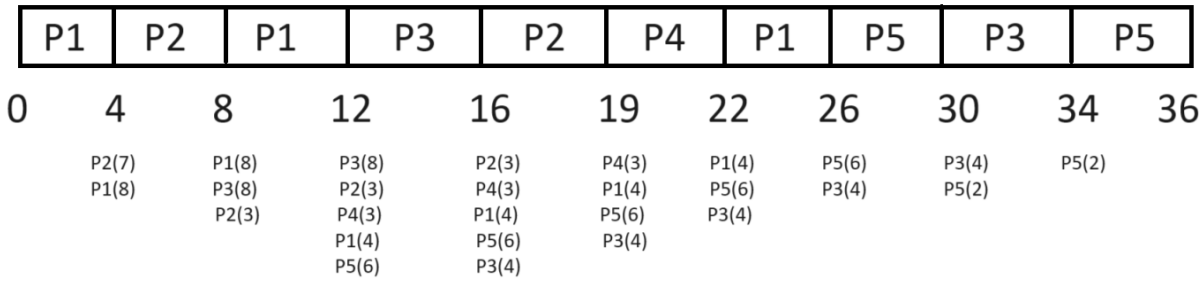
```
    printf("\nAvgwt= %f\nAvgtat= %f\n", (float)totwt / n, (float)tottat / n);
return 0;
```

Test case 1

RR(quantum time = 4)

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

Kết quả giản đồ gantt:



Thời gian chờ trung bình: (14 + 10 + 21 + 10 + 18)/5 = 14,6

Thời gian chờ trung bình: (26 + 17 + 29 + 13 + 24)/5 = 21,8

```
Enter the Process Name, Arrival Time, Burst Time: 1 0 12
Enter the Process Name, Arrival Time, Burst Time: 2 2 7
Enter the Process Name, Arrival Time, Burst Time: 3 5 8
Enter the Process Name, Arrival Time, Burst Time: 4 9 3
Enter the Process Name, Arrival Time, Burst Time: 5 12 6

PName Arrtime Burtime Start Tat Finish Wt
P5          12       6      26      24      36      18
P3           5       8      12      29      34      21
P1           0      12       0      26      26      14
P4           9       3      19      13      22      10
P2           2       7       4      17      19      10
Avgwt= 14.600000
Avgtat= 21.799999
```

Test case 2

RR (quantum time = 10)

Process	Arrival Time	Burst Time
P1	0	10
P2	2	29
P3	4	3
P4	5	7
P5	7	12

Kết quả giản đồ gantt:

0 10 20 23 30 40 50 52 61

Thời gian chờ trung bình: (0 + 30 + 16 + 18 + 33)/5 = 19,4

Thời gian chờ trung bình: (10 + 59 + 19 + 25 + 45)/5 = 31,6

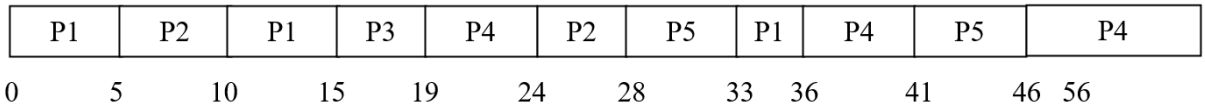
```
Enter the number of processes: 5
Enter QTime: 10
Enter the Process Name, Arrival Time, Burst Time: 1 0 10
Enter the Process Name, Arrival Time, Burst Time: 2 2 29
Enter the Process Name, Arrival Time, Burst Time: 3 4 3
Enter the Process Name, Arrival Time, Burst Time: 4 5 7
Enter the Process Name, Arrival Time, Burst Time: 5 7 12

PName Arrtime Burtime Start Tat Finish Wt
P2          2      29      10      59      61      30
P5          7      12      30      45      52      33
P4          5       7      23      25      30      18
P3          4       3      20      19      23      16
P1          0      10       0      10      10       0
Avgwt= 19.400000
Avgtat= 31.600000
```

Test case 3

RR (quantum time = 5)

Process	Arrival Time	Burst Time
P1	0	13
P2	4	9
P3	6	4
P4	7	20
P5	12	10

Kết quả giản đồ gantt:



Thời gian đợi trung bình: ((5 + 18) + (1 + 14) + 9 + (12 + 12 + 5) + (16 + 8))/5 = 20

Thời gian hoàn thành trung bình: (36 + 24 + 13 + 49 + 34)/5 = 31.2

```
Enter the number of processes: 5
Enter QTime: 5
Enter the Process Name, Arrival Time, Burst Time: 1 0 13
Enter the Process Name, Arrival Time, Burst Time: 2 4 9
Enter the Process Name, Arrival Time, Burst Time: 3 6 4
Enter the Process Name, Arrival Time, Burst Time: 4 7 20
Enter the Process Name, Arrival Time, Burst Time: 5 12 10

PName Arrtime Burtime Start Tat Finish Wt
P4           7      20      19      49      56      29
P5          12      10      28      34      46      24
P1           0      13       0      36      36      23
P2           4       9       5      24      28      15
P3           6       4      15      13      19       9
Avgwt= 20.000000
Avgtat= 31.200001
```