Name: Nguyễn Đức Luân

ID: 2250825

Class: IT007.O21.ANTN

OPERATING SYSTEM
LAB 4'S REPORT

SUMMARY

Task	Status	Page
1. Giải thuật SJF: Vẽ lưu đồ giải thuật -Trình bày tính đúng đắn của lưu đồ bằng cách chạy tay ít nhất 01 test case tối thiểu 05 tiến trình -Thực hiện code cho giải thuật -Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình	DONE	
2. giải thuật SRTF: -Vẽ lưu đồ giải thuật -Trình bày tính đúng đắn của lưu đồ bằng cách chạy tay ít nhất 01 test case tối thiểu 05 tiến trình -Thực hiện code cho giải thuật -Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình	DONE	
3. (Bonus) Thực hiện các yêu cầu trên với giải thuật còn lại.(ROUND	DONE	

ROBIN)		

Self-scrores: 9

Xem chi tiết code vào link: https://github.com/nauL1412/IT007_LAB4

TASK 01: GIẢI THUẬT SJF

1)Lưu đồ giải thuật

Bắt đầu

|

|__ Khai báo cấu trúc Process để lưu thông tin về mỗi tiến trình.

|

|__ Khai báo cấu trúc burst_priority và arr_priority để sắp xếp tiến trình theo thứ tự ưu tiên.

|

|__ Khai báo biến num_process để lưu số lượng tiến trình.

|

|__ Khai báo hàng đợi ưu tiên input_process để lưu trữ các tiến trình theo thứ tự đến.

|

|__ Khai báo hàng đợi ưu tiên queue_process để lưu trữ các tiến trình theo thứ tự burst time.

|

|__ Khai báo vector result_process để lưu kết quả cuối cùng.

|

|__ Nhập số lượng tiến trình từ người dùng và thông tin của từng tiến trình.

|

|__ Khởi tạo biến remain_process để theo dõi số tiến trình còn lại chưa hoàn thành.

|

|__ Khởi tạo biến time_start để theo dõi thời gian bắt đầu thực hiện.

|

|__ Trong khi còn tiến trình chưa hoàn thành hoặc hàng đợi queue_process không trống:

| |

| |__ Kiểm tra xem có tiến trình mới đến không:

| | |

| | |__ Nếu có, đưa tiến trình vào hàng đợi queue_process dựa trên thời gian đến.

| |

| |__ Nếu hàng đợi queue_process không trống:

| | |

| | |__ Lấy tiến trình có burst time nhỏ nhất ra khỏi hàng đợi.

| | |

| | |__ Tính thời gian phản hồi, thời gian kết thúc , thời gian hoàn thành và thời gian chờ cho tiến trình này.

| | |

| | |__ Cập nhật start_time là thời gian kết thúc của tiến trình trước đó.

| | |

| | |__ Lưu tiến trình vào danh sách kết quả.

| | |

| | |__ Giảm số tiến trình còn lại đi một đơn vị.

|

|__ Sắp xếp danh sách kết quả theo PID.

|

|__ Hiển thị

2) Kiểm tra tính đúng đắn cho giải thuật

Process	Arrival Time	Burst Time
P1	0	10
P2	1	3
P3	2	2
P4	3	1
P5	4	5

Đầu tiên đưa các thông tin của các tiến trình vào hàng đợi input_process=[P1,P2,P3,P4,P5]

Đặt biến time_start=0;

Đặt biến remain_process=5

Lúc này remain_process khác 0, kiểm tra xem trong input_process có tiến trình nào có arival_time <= time_start kay không, trong trường hợp này P1 thỏa mãn điều kiện nên đưa P1 vào trong queue_process để xử lý.

Khi này ta thấy trong queue_process chỉ có P1 là tiến trình có burst time nhỏ nhất nên thực hiện tính toán cho P1, cập nhật các giá trị time_start+= P1.burst_time= 10

Sau khi thực hiện xong đưa P1 ra khỏi queue_process và đưa vào result_process để in ra kết quả và tính các giá trị trung bình.

Giảm remain_process đi 1 còn 4

Lúc này remain_process vẫn khác 0, kiểm tra xem trong input_process có tiến trình nào có arival_time <= time_start(lúc này bằng 10) hay không, trong trường hợp này P2,P3,P4,P5 thỏa mãn điều kiện nên đưa lần lượt vào trong queue_process để xử lý sắp xếp theo ưu tiên burst time nhỏ nhất là P4,P2,P3,P5.

Tiếp tục thực hiện các bước như P1

3)Thực hiện code cho giải thuật

SJF.cpp > Process > Process(int, int, int)

```cpp
#include<iostream>
#include<algorithm>
#include<vector>
#include <queue>
using namespace std;
struct Process {
    int pid;
    int arr_time;
    int burst_time;
    int wait_time;
    int rp_time;
    int turound_time;
    int finish_time;
    int start_time;
    int time_remain;
    Process(int pid, int arr_t, int burst_t){
        this->pid= pid;
        this->arr_time= arr_t;
        this->burst_time=burst_t;
        this->wait_time=0;
        this->turound_time=0;
        this->rp_time=0;
        this->time_remain= burst_t;
        this->start_time=0;
        this->finish_time=0;
    }
};
```

```cpp
struct burst_priority{
    bool operator()(const Process& Process1, const Process& Process2) {

        return Process1.burst_time > Process2.burst_time;
    }
};
struct arr_priority{
    bool operator()(const Process& Process1, const Process& Process2) {

        return Process1.arr_time > Process2.arr_time;
    }
};


bool compare_pid (struct Process a, struct Process b){
    return a.pid < b.pid;
}
```

```
int num_process;
void show(vector<Process> P){

    printf(" ____________________________________________________________________________________________ \n");
    printf("|      PID       | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |\n");
    printf("|________________|______________|____________|_________________|_________________|______________|_______________|\n");

    float avg_waiting_time = 0;
    float avg_turnaround = 0;
    float avg_respone_time = 0;
    for(int i=0;i<num_process;i++){
        printf("|%-16d|%-14d|%-12d|%-17d|%-17d|%-14d|%-15d|\n", P[i].pid, P[i].arr_time, P[i].burst_time, P[i].finish_time, P[i].turound_time, P[i].wait_time, P[i].rp_time);

        avg_waiting_time += P[i].wait_time;
        avg_turnaround += P[i].turound_time;
        avg_respone_time += P[i].rp_time;
    }

    printf("|________________|______________|____________|_________________|_________________|______________|_______________|\n");
    avg_waiting_time /= num_process;
    avg_turnaround /= num_process;
    avg_respone_time /= num_process;

    printf("Average waiting time : %.2f\n", avg_waiting_time);
    printf("Average turn around time: %.2f\n", avg_turnaround);
    printf("Average respone time: %.2f\n",avg_respone_time);
}
```

```
int main(){
    //vector <Process> input_process;
    priority_queue<Process, vector<Process>, burst_priority> queue_process;
    priority_queue<Process, vector<Process>, arr_priority> input_process;
    vector <Process> result_process;
    cout<<"Enter number process: ";
    cin>> num_process;
    int arr_t,burst_t=0;
    for (int i=0; i<num_process;i++){

        cout<<"Process pid "<<i<<endl;
        cout<<"Enter arrival time: ";
        cin>>arr_t;
        cout<<"Enter busrt time: ";
        cin>> burst_t;
        Process token(i,arr_t,burst_t);
        input_process.push(token);


    }
```

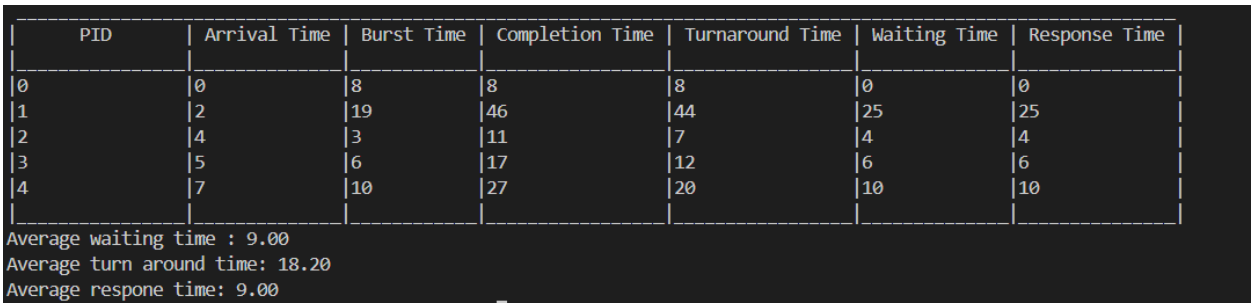
```
int main(){
    //vector <Process> input_process;
    priority_queue<Process, vector<Process>, burst_priority> queue_process;
    priority_queue<Process, vector<Process>, arr_priority> input_process;
    vector <Process> result_process;
    cout<<"Enter number process: ";
    cin>> num_process;
    int arr_t,burst_t=0;
    for (int i=0; i<num_process;i++){

        cout<<"Process pid "<<i<<endl;
        cout<<"Enter arrival time: ";
        cin>>arr_t;
        cout<<"Enter busrt time: ";
        cin>> burst_t;
        Process token(i,arr_t,burst_t);
        input_process.push(token);


    }
```

4) Kiểm tra tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình:

Test case 1:

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

Kết quả chạy code:



PID	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
0	0	8	8	8	0	0
1	2	19	46	44	25	25
2	4	3	11	7	4	4
3	5	6	17	12	6	6
4	7	10	27	20	10	10

Average waiting time : 9.00
Average turn around time: 18.20
Average respone time: 9.00

Test case 2:

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

Kết quả chạy code:

PID	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
0	0	13	13	13	0	0
1	4	9	26	22	13	13
2	6	4	17	11	7	7
3	7	20	56	49	29	29
4	12	10	36	24	14	14

```
Average waiting time : 12.60
Average turn around time: 23.80
Average respone time: 12.60
```

Test case 3:

Process	Arrival Time	Burst Time
P1	0	10
P2	1	3
P3	2	2
P4	3	1
P5	4	5

Kết quả chạy code:

PID	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
0	0	10	10	10	0	0
1	1	3	16	15	12	12
2	2	2	13	11	9	9
3	3	1	11	8	7	7
4	4	5	21	17	12	12

```
Average waiting time : 8.00
Average turn around time: 12.20
Average respone time: 8.00
```

TASK 02: GIẢI THUẬT SRTF

1)Lưu đồ giải thuật

Bắt đầu

|

|__ Khai báo cấu trúc Process để lưu thông tin về mỗi tiến trình.

|

|__ Khai báo cấu trúc remain_burst_priority và arr_priority để sắp xếp tiến trình theo thứ tự ưu tiên.

|

|__ Khai báo biến num_process để lưu số lượng tiến trình.

|

|__ Khai báo hàng đợi ưu tiên input_process để lưu trữ các tiến trình theo thứ tự đến.

|

|__ Khai báo hàng đợi ưu tiên queue_process để lưu trữ các tiến trình theo thứ tự thời gian còn lại ngắn nhất.

|

|__ Khai báo vector result_process để lưu kết quả cuối cùng.

|

|__ Nhập số lượng tiến trình từ người dùng và thông tin của từng tiến trình.

|

|__ Khởi tạo biến remain_process để theo dõi số tiến trình còn lại chưa hoàn thành.

|

|__ Khởi tạo biến time_start để theo dõi thời gian bắt đầu thực hiện.

|

|__ Trong khi còn tiến trình chưa hoàn thành hoặc hàng đợi queue_process không trống:

| |

| |__ Kiểm tra xem có tiến trình mới đến không:

| | |

| | |__ Nếu có và thời gian đến của tiến trình mới nhỏ hơn hoặc bằng thời gian hiện tại, đưa tiến trình vào hàng đợi queue_process.

| |

| |__ Nếu hàng đợi queue_process không trống:

| | |

| | |__ Lấy tiến trình có thời gian còn lại ngắn nhất ra khỏi hàng đợi.

| | |

| | |__ Kiểm tra xem tiến trình đã được đánh dấu bắt đầu chưa. Nếu chưa, đánh dấu thời gian bắt đầu và đánh dấu là đã bắt đầu.

| | |

| | |__ Giảm thời gian còn lại của tiến trình đi một đơn vị.

| | |

| | |__ Tăng thời gian bắt đầu lên một đơn vị.

| | |

| | |__ Nếu thời gian còn lại của tiến trình vẫn lớn hơn 0, đưa tiến trình trở lại hàng đợi.

| | |

| | |__ Nếu thời gian còn lại của tiến trình là 0:

| | | |

| | | |__ Tính thời gian phản hồi, thời gian hoàn thành và thời gian chờ cho tiến trình này.

| | | |

| | | |__ Lưu tiến trình vào danh sách kết quả.

| | | |

| | | |__ Giảm số tiến trình còn lại đi một đơn vị.

| |

| |__ Nếu hàng đợi SRTF trống và vẫn còn tiến trình chưa hoàn thành:

| | |

| | |__ Tăng thời gian bắt đầu lên một đơn vị.

|

|__ Sắp xếp danh sách kết quả theo PID.

|

|__ Hiển thị

2)Thực hiện code cho giải thuật:

```cpp
#include<iostream>
#include<algorithm>
#include<vector>
#include <queue>
using namespace std;
struct Process {
    int pid;
    int arr_time;
    int burst_time;
    int wait_time;
    int rp_time;
    int turound_time;
    int finish_time;
    int start_time;
    int time_remain;
    bool flag;
    Process(int pid, int arr_t, int burst_t){
        this->pid= pid;
        this->arr_time= arr_t;
        this->burst_time=burst_t;
        this->wait_time=0;
        this->turound_time=0;
        this->rp_time=0;
        this->time_remain= burst_t;
        this->start_time=0;
        this->finish_time=0;
        this->flag=false;
    }
};

```

```cpp
struct remain_burst_priority{
    bool operator()(const Process& Process1, const Process& Process2) {

        return Process1.time_remain > Process2.time_remain;
    }
};
struct arr_priority{
    bool operator()(const Process& Process1, const Process& Process2) {

        return Process1.arr_time > Process2.arr_time;
    }
};


bool compare_pid (struct Process a, struct Process b){
    return a.pid < b.pid;
}
```

```cpp
int num_process;
void show(vector<Process> P){

    printf(" ____________________________________________________________________________________________ \n");
    printf("|      PID       | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |\n");
    printf("|________________|______________|____________|_________________|_________________|______________|_______________|\n");

    float avg_waiting_time = 0;
    float avg_turnaround = 0;
    float avg_respone_time = 0;
    for(int i=0;i<num_process;i++){
        printf("|%-16d|%-14d|%-12d|%-17d|%-17d|%-14d|%-15d|\n", P[i].pid, P[i].arr_time, P[i].burst_time, P[i].finish_time, P[i].turound_time, P[i].wait_time, P[i].rp_

        avg_waiting_time += P[i].wait_time;
        avg_turnaround += P[i].turound_time;
        avg_respone_time += P[i].rp_time;
    }

    printf("|________________|______________|____________|_________________|_________________|______________|_______________|\n");
    avg_waiting_time /= num_process;
    avg_turnaround /= num_process;
    avg_respone_time /= num_process;

    printf("Average waiting time : %.2f\n", avg_waiting_time);
    printf("Average turn around time: %.2f\n", avg_turnaround);
    printf("Average respone time: %.2f\n",avg_respone_time);
}
```

```cpp
int main(){
    //vector <Process> input_process;
    priority_queue<Process, vector<Process>, remain_burst_priority> queue_process;
    priority_queue<Process, vector<Process>, arr_priority> input_process;
    vector <Process> result_process;
    cout<<"Enter number process: ";
    cin>> num_process;
    int arr_t,burst_t=0;
    for (int i=0; i<num_process;i++){

        cout<<"Process pid "<<i<<endl;
        cout<<"Enter arrival time: ";
        cin>>arr_t;
        cout<<"Enter busrt time: ";
        cin>> burst_t;
        Process token(i,arr_t,burst_t);
        input_process.push(token);


    }
```

```cpp
 int remain_process= num_process;
 int time_start=0;
 while(remain_process !=0 || !queue_process.empty()){
    bool exist=true;
    do{

        if(!input_process.empty()&&input_process.top().arr_time <= time_start){
            Process tmp =input_process.top();
            queue_process.push(tmp);
            input_process.pop();
        }else exist=false;
    }
    while (exist);
    if(queue_process.empty() && remain_process !=0){
        time_start++;
        continue;
    }
    Process current_p = queue_process.top();
    if(!current_p.flag){
        current_p.start_time= time_start;
        current_p.flag=true;
    }
    queue_process.pop();
    time_start++;
    current_p.time_remain--;
```

```
    if(current_p.time_remain>0){
        queue_process.push(current_p);
    }
    else {
        current_p.finish_time=time_start;
        current_p.rp_time= current_p.start_time - current_p.arr_time;
        current_p.turound_time= current_p.finish_time - current_p.arr_time;
        current_p.wait_time= current_p.turound_time- current_p.burst_time;
        result_process.push_back(current_p);
        remain_process--;
    }

}
sort(result_process.begin(), result_process.end(),compare_pid);
show(result_process);
```

3) Kiểm tra tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình:

Test case 1:

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

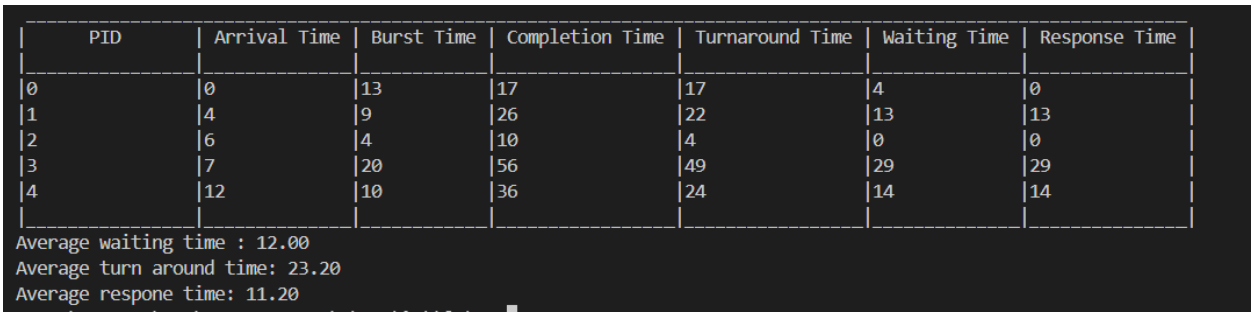
Kết quả chạy code:

```
|      PID       | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |
|0               |0             |8           |11               |11               |3             |0              |
|1               |2             |19          |46               |44               |25            |25             |
|2               |4             |3           |7                |3                |0             |0              |
|3               |5             |6           |17               |12               |6             |6              |
|4               |7             |10          |27               |20               |10            |10             |
Average waiting time : 8.80
Average turn around time: 18.00
Average respone time: 8.20
```

Test case 2:

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

Kết quả chạy code:



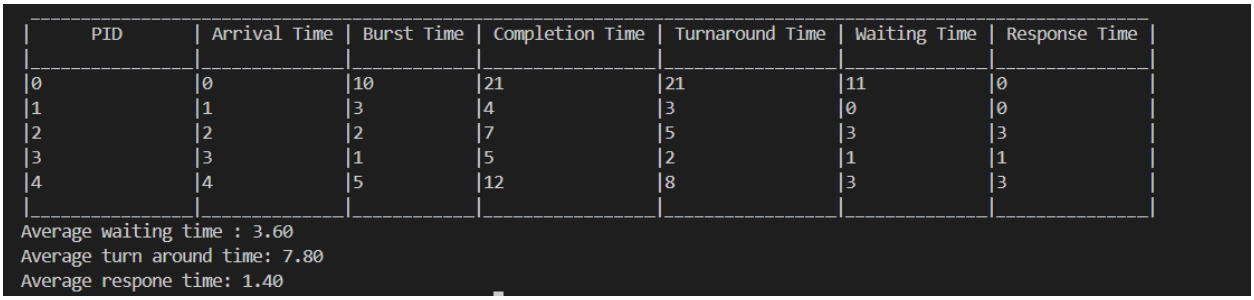
PID	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
0	0	13	17	17	4	0
1	4	9	26	22	13	13
2	6	4	10	4	0	0
3	7	20	56	49	29	29
4	12	10	36	24	14	14

Average waiting time : 12.00
Average turn around time: 23.20
Average respone time: 11.20

Test case 3:

Process	Arrival Time	Burst Time
P1	0	10
P2	1	3
P3	2	2
P4	3	1
P5	4	5

Kết quả chạy code:



PID	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
0	0	10	21	21	11	0
1	1	3	4	3	0	0
2	2	2	7	5	3	3
3	3	1	5	2	1	1
4	4	5	12	8	3	3

Average waiting time : 3.60
Average turn around time: 7.80
Average respone time: 1.40

TASK 03: GIẢI THUẬT ROUND ROBIN

1)Lưu đồ giải thuật

Bắt đầu

|

|__ Nhập số lượng tiến trình (num_process) và thời gian lượng tử (quantum_time)

|

|__ Khởi tạo hàng đợi ưu tiên (priority_queue) input_process để lưu trữ các tiến trình theo thứ tự đến.

|

|__ Vòng lặp để nhập thông tin của mỗi tiến trình:

| |

| |__ Nhập thời gian đến (arr_t) và thời gian burst (burst_t) cho mỗi tiến trình.

| |

| |__ Tạo một đối tượng tiến trình với thông tin vừa nhập và đưa vào hàng đợi ưu tiên input_process.

|

|__ Khởi tạo biến remain_process để theo dõi số tiến trình còn lại chưa hoàn thành.

|

|__ Khởi tạo biến time_start để theo dõi thời gian bắt đầu thực hiện.

|

|__ Trong khi còn tiến trình chưa hoàn thành hoặc hàng đợi không trống:

| |

| |__ Kiểm tra xem có tiến trình mới đến không:

| | |

| | |__ Nếu có, đưa tiến trình vào hàng đợi của Round Robin.

| |

| |__ Lấy tiến trình đầu tiên từ hàng đợi Round Robin.

| |

| |__ Nếu tiến trình chưa được đánh dấu bắt đầu, đánh dấu thời gian bắt đầu và đánh dấu đã bắt đầu.

| |

| |__ Nếu chưa có thời gian lượt chạy, thiết lập thời gian lượt chạy bằng quantum time

| |

| |__ Tăng thời gian bắt đầu lên một đơn vị.

| |

| |__ Giảm thời gian còn lại của tiến trình hiện tại đi một đơn vị.

| |

| |__ Giảm thời gian lượt chạy của tiến trình hiện tại đi một đơn vị.

| |

| |__ Nếu thời gian còn lại của tiến trình hiện tại vẫn còn:

| | |

| | |__ Nếu không còn thời gian lượt chạy:

| | | |

| | | |__ Lấy tiến trình ra khỏi hàng đợi Round Robin.

| | | |

| | | |__ Nếu có tiến trình mới đến, đưa tiến trình đó vào hàng đợi.

| | | |

| | | |__ Đưa tiến trình hiện tại vào cuối hàng đợi.

| | |

| | |__ Nếu còn thời gian lượt chạy, cập nhật lại tiến trình hiện tại trong hàng đợi.

| |

| |__ Nếu thời gian còn lại của tiến trình hiện tại là 0:

| | |

| | |__ Đánh dấu thời gian hoàn thành cho tiến trình hiện tại.

| | |

| | |__ Tính thời gian phản hồi, thời gian hoàn thành và thời gian chờ cho tiến trình hiện tại.

| | |

| | |__ Lưu tiến trình hiện tại vào danh sách kết quả.
| | |
| | |__ Loại bỏ tiến trình hiện tại khỏi hàng đợi Round Robin.
| | |
| | |__ Giảm số tiến trình còn lại đi một đơn vị.
|
|__ Sắp xếp danh sách kết quả theo PID.
|
|__ Hiển thị bảng kết quả.
Kết thúc

2)Thực hiện code cho giải thuật:

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
using namespace std;
struct Process
{
    int pid;
    int arr_time;
    int burst_time;
    int wait_time;
    int rp_time;
    int turound_time;
    int finish_time;
    int start_time;
    int time_remain;
    int turn_time;
    bool flag;
    Process(int pid, int arr_t, int burst_t)
    {
        this->pid = pid;
        this->arr_time = arr_t;
        this->burst_time = burst_t;
        this->wait_time = 0;
        this->turound_time = 0;
        this->rp_time = 0;
        this->time_remain = burst_t;
        this->start_time = 0;
        this->finish_time = 0;
        this->turn_time = 0;
        this->flag = false;
    }
};
```

```cpp
struct arr_priority
{
    bool operator()(const Process &Process1, const Process &Process2)
    {

        return Process1.arr_time > Process2.arr_time;
    }
};

bool compare_pid(struct Process a, struct Process b)
{
    return a.pid < b.pid;
}
```

```cpp
int num_process;
void show(vector<Process> P)
{

    printf(" ____________________________________________________________________________________________ \n");
    printf("|      PID       | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |\n");
    printf("|________________|______________|____________|_________________|_________________|______________|_______________|\n");

    float avg_waiting_time = 0;
    float avg_turnaround = 0;
    float avg_respone_time = 0;
    for (int i = 0; i < num_process; i++)
    {
        printf("|%-16d|%-14d|%-12d|%-17d|%-17d|%-14d|%-15d|\n", P[i].pid, P[i].arr_time, P[i].burst_time, P[i].finish_time, P[i].turound_time, P[i].wait_time, P[i

        avg_waiting_time += P[i].wait_time;
        avg_turnaround += P[i].turound_time;
        avg_respone_time += P[i].rp_time;
    }

    printf("|________________|______________|____________|_________________|_________________|______________|_______________|\n");
    avg_waiting_time /= num_process;
    avg_turnaround /= num_process;
    avg_respone_time /= num_process;

    printf("Average waiting time : %.2f\n", avg_waiting_time);
    printf("Average turn around time: %.2f\n", avg_turnaround);
    printf("Average respone time: %.2f\n", avg_respone_time);
}
```

```cpp
int main()
{

    queue<Process> queue_process;
    priority_queue<Process, vector<Process>, arr_priority> input_process;
    vector<Process> result_process;
    int quantum_time;
    cout << "Enter quantum time: ";
    cin >> quantum_time;
    cout << "Enter number process: ";
    cin >> num_process;
    int arr_t, burst_t = 0;
    for (int i = 0; i < num_process; i++)
    {

        cout << "Process pid " << i << endl;
        cout << "Enter arrival time: ";
        cin >> arr_t;
        cout << "Enter busrt time: ";
        cin >> burst_t;
        Process token(i, arr_t, burst_t);
        input_process.push(token);
    }
```

```
int remain_process = num_process;
int time_start = 0;
while (remain_process != 0 || !queue_process.empty())
{
    bool exist = true;
    do
    {

        if (!input_process.empty() && input_process.top().arr_time == time_start)
        {
            Process tmp = input_process.top();
            queue_process.push(tmp);
            input_process.pop();
        }
        else
            exist = false;
    } while (exist);
    if (queue_process.empty() && remain_process != 0)
    {
        time_start++;
        continue;
    }
    Process current_p = queue_process.front();
    if (!current_p.flag)
    {
        current_p.start_time = time_start;
        current_p.flag = true;
    }
    if (!current_p.turn_time)
    {
        current_p.turn_time = quantum_time;
    }
```

```
        time_start++;
        current_p.time_remain--;
        current_p.turn_time--;
        if (current_p.time_remain > 0)
        {
            if (!current_p.turn_time)
            {
                queue_process.pop();
                if (!input_process.empty() && input_process.top().arr_time == time_start)
                {
                    Process tmp = input_process.top();
                    queue_process.push(tmp);
                    input_process.pop();
                }
                queue_process.push(current_p);
            }
            else
            {
                queue_process.front() = current_p;
                continue;
            }
        }
        else
        {
            current_p.finish_time = time_start;
            current_p.rp_time = current_p.start_time - current_p.arr_time;
            current_p.turound_time = current_p.finish_time - current_p.arr_time;
            current_p.wait_time = current_p.turound_time - current_p.burst_time;
            result_process.push_back(current_p);
            queue_process.pop();
            remain_process--;
        }
    }
    sort(result_process.begin(), result_process.end(), compare_pid);
    show(result_process);
}
```

3) Kiểm tra tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình

Test case 1:

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

Kết quả chạy code:(Quantum time =2)

```
 _____________________________________________________________________________________________
|      PID      | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |
|_______________|______________|____________|_________________|_________________|______________|_______________|
|0              |0             |8           |23               |23               |15            |0              |
|1              |2             |19          |46               |44               |25            |0              |
|2              |4             |3           |17               |13               |10            |2              |
|3              |5             |6           |29               |24               |18            |5              |
|4              |7             |10          |39               |32               |22            |7              |
|_______________|______________|____________|_________________|_________________|______________|_______________|
Average waiting time : 18.00
Average turn around time: 27.20
Average respone time: 2.80
```

Test case 2:

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

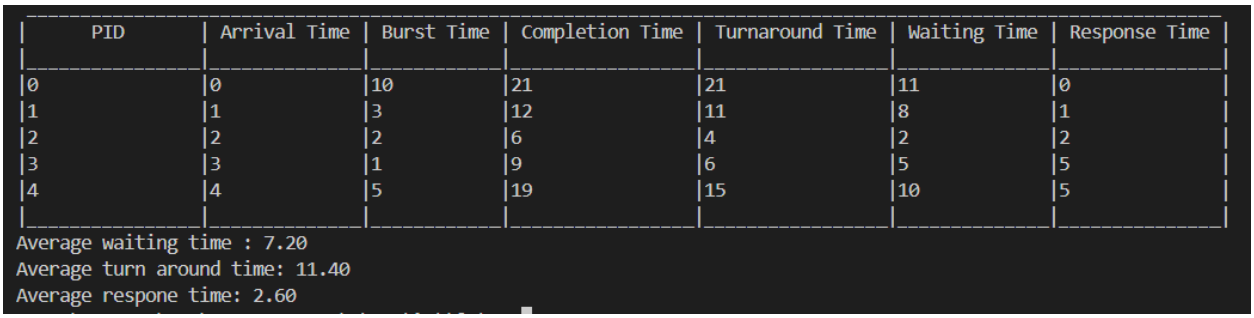
Kết quả chạy code:(Quantum time =2)

```
 _____________________________________________________________________________________________
|      PID      | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |
|_______________|______________|____________|_________________|_________________|______________|_______________|
|0              |0             |13          |40               |40               |27            |0              |
|1              |4             |9           |37               |33               |24            |0              |
|2              |6             |4           |18               |12               |8             |2              |
|3              |7             |20          |56               |49               |29            |5              |
|4              |12            |10          |46               |34               |24            |6              |
|_______________|______________|____________|_________________|_________________|______________|_______________|
Average waiting time : 22.40
Average turn around time: 33.60
Average respone time: 2.60
```

Test case 3:

Process	Arrival Time	Burst Time
P1	0	10
P2	1	3
P3	2	2
P4	3	1
P5	4	5

Kết quả chạy code:(Quantum time =2)



PID	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
0	0	10	21	21	11	0
1	1	3	12	11	8	1
2	2	2	6	4	2	2
3	3	1	9	6	5	5
4	4	5	19	15	10	5

Average waiting time : 7.20
Average turn around time: 11.40
Average respone time: 2.60