OS LAB 2

NAME: Aditya Anand

ROLL NO.: 20124009

BRANCH: IT

S NO.	TITLE	DATE OF IMPLEMENTATION	REMARKS
1	Program to implement Shortest Job First Process of CPU Scheduling	02-02-2022	
2	Program to implement First Come First Serve Process of CPU Scheduling	02-02-2022	
3	Program to implement Priority based Scheduling Process of CPU Scheduling	02-02-2022	

SHORTEST JOB FIRST CPU SCHEDULING

CRITERIA: Burst Time

NOTE: In case of same burst time, process with lower arrival time is executed first.

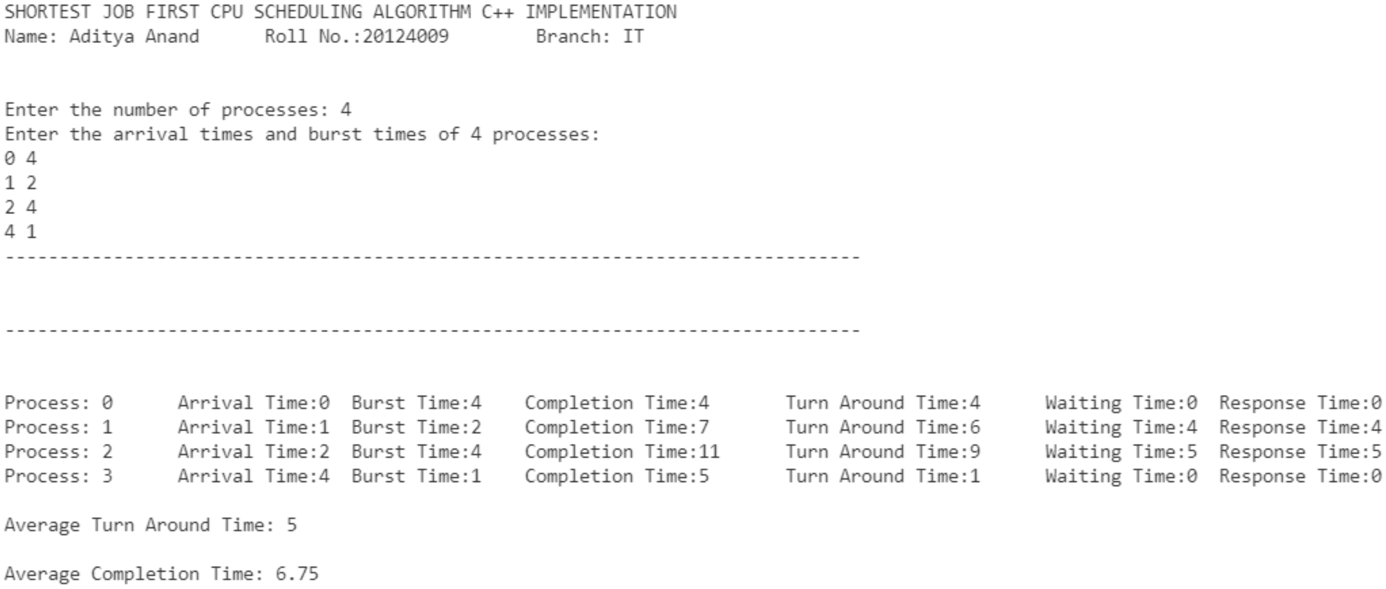
MODE: Non pre-emptive

GIVEN: List of processes with their arrival and burst time.

CODE:

```
#include <bits/stdc++.h>
using namespace ::std;

class process{
    public:
        int priority;
        int id;
        int arrivalTime;
        int burstTime;
        bool ready;
        int completionTime;
        int TAT;
        int WT;
        int RT;
};

struct comp{
    bool operator()(process const &p1, process const &p2){
        return p1.burstTime > p2.burstTime;
    }
};

void SJF(vector<process> &v){
    priority_queue<process, vector<process>, comp> p;
    int cur_time = INT_MAX;
    int n=v.size();
    for(int i=0; i<n; i++){
        cur_time = min(cur_time, v[i].arrivalTime);
    }

    int count = 0;
    while(true){
        for(int i=0; i<n; i++){
            if(v[i].arrivalTime<=cur_time && !v[i].ready){
                v[i].ready = true;
                p.push(v[i]);
                count++;
            }
        }
        if(count<n && p.empty()){
            cout<<"CPU empty from "<<cur_time<<" to "<<cur_time+1<<"\n";
            cur_time++;
            continue;
        }
        if(p.empty()){
            break;
        }
        process cur_process = p.top();
        p.pop();
        v[cur_process.id].RT = cur_time-cur_process.arrivalTime;
        cur_time+=cur_process.burstTime;
        v[cur_process.id].completionTime=cur_time;
    }
}

int main(){
    cout<<"SHORTEST JOB FIRST CPU SCHEDULING ALGORITHM C++ IMPLEMENTATION\n";
    cout<<"Name: Aditya Anand\tRoll No.:20124009\t Branch: IT\n\n\n";

    int n=0;
    cout<<"Enter the number of processes: ";
    cin>>n;

    cout<<"Enter the arrival times and burst times of "<<n<<" processes: \n";

    vector<process> v(n);
    for(int i=0; i<n; i++){
        cin>>v[i].arrivalTime>>v[i].burstTime;
        v[i].id = i;
        v[i].ready = false;
    }

    cout<<"-------------------------------------------------------------------------------\n";
    cout<<"\n";
    SJF(v);
    cout<<"\n";
    cout<<"-------------------------------------------------------------------------------\n";
    cout<<"\n\n";

    int t_TAT=0;
    int t_CT=0;
    for(int i=0; i<n; i++){
        v[i].TAT = v[i].completionTime-v[i].arrivalTime;
        v[i].WT = v[i].TAT-v[i].burstTime;
        t_TAT+=v[i].TAT;
        t_CT+=v[i].completionTime;
    }

    for(auto p:v){
        cout<<"Process: "<<p.id<<"\tArrival Time:"<<p.arrivalTime<<"\tBurst
Time:"<<p.burstTime<<"\tCompletion Time:"<<p.completionTime;
        cout<<"\tTurn Around Time:"<<p.TAT<<"\tWaiting Time:"<<p.WT<<"\tResponse Time:"<<p.RT<<"\n";
    }

    cout<<"\nAverage Turn Around Time: "<<(float)((1.0*t_TAT)/(1.0*n))<<"\n";
    cout<<"\nAverage Completion Time: "<<(float)((1.0*t_CT)/(1.0*n))<<"\n";
    return 0;
}
```

RESULT:



```
SHORTEST JOB FIRST CPU SCHEDULING ALGORITHM C++ IMPLEMENTATION
Name: Aditya Anand      Roll No.:20124009        Branch: IT


Enter the number of processes: 4
Enter the arrival times and burst times of 4 processes:
0 4
1 2
2 4
4 1
-------------------------------------------------------------------------------



-------------------------------------------------------------------------------


Process: 0      Arrival Time:0  Burst Time:4    Completion Time:4       Turn Around Time:4      Waiting Time:0  Response Time:0
Process: 1      Arrival Time:1  Burst Time:2    Completion Time:7       Turn Around Time:6      Waiting Time:4  Response Time:4
Process: 2      Arrival Time:2  Burst Time:4    Completion Time:11      Turn Around Time:9      Waiting Time:5  Response Time:5
Process: 3      Arrival Time:4  Burst Time:1    Completion Time:5       Turn Around Time:1      Waiting Time:0  Response Time:0

Average Turn Around Time: 5

Average Completion Time: 6.75
```

FIRST COME FIRST SERVE CPU SCHEDULING

CRITERIA: Arrival Time

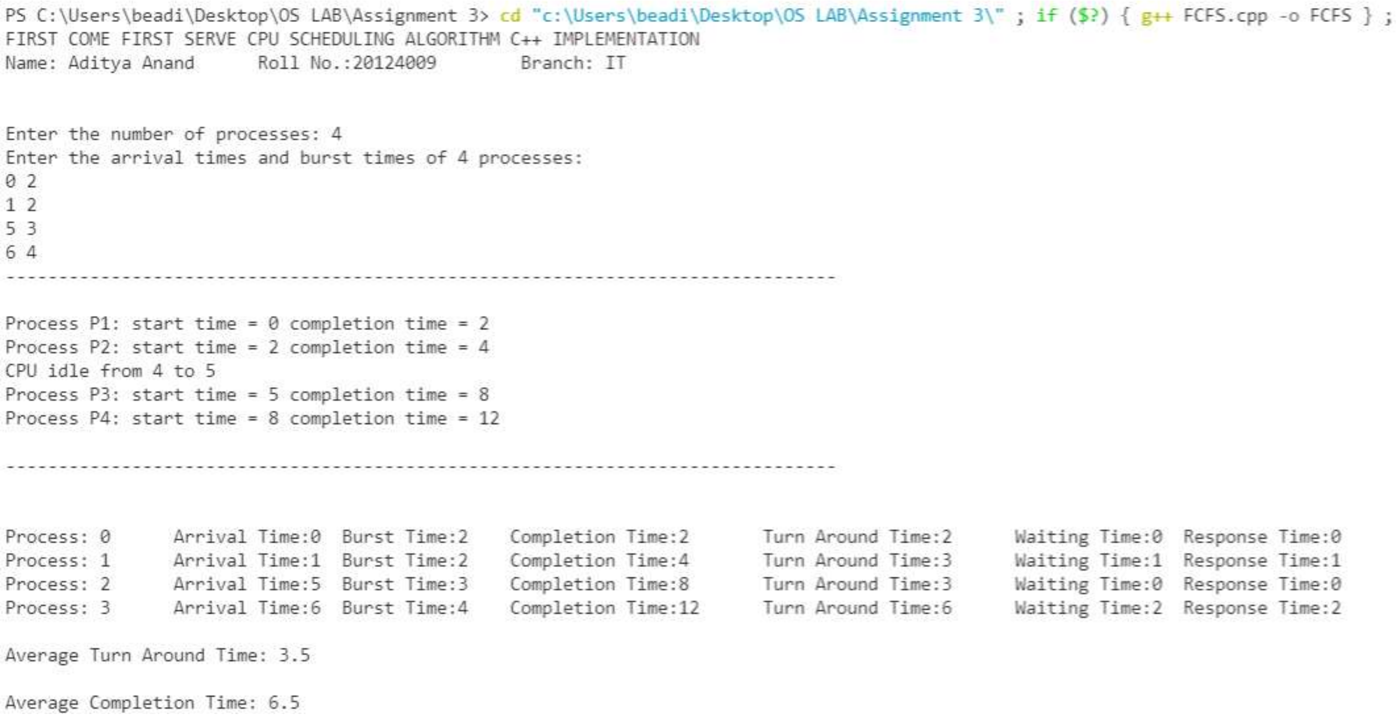
MODE: Non pre-emptive

GIVEN: List of processes with their arrival and burst time.

CODE:

```
#include<bits/stdc++.h>
using namespace :: std;

class process{
    public:
        int id;
        int arrivalTime;
        int burstTime;
        int completionTime;
        int TAT;
        int WT;
        int RT;
};

void FCFS(vector<process> &v){
    int cur_time = 0;
    int id = 0;
    for(int i=0; i<v.size(); i++){
        if(cur_time<v[i].arrivalTime){
            cout<<"CPU idle from "<<cur_time<<" to "<<v[i].arrivalTime<<endl;
            cur_time = v[i].arrivalTime;
        }
        v[i].completionTime = cur_time+v[i].burstTime;
        v[i].RT = cur_time-v[i].arrivalTime;

        cout<<"Process P"<<v[i].id+1<<": start time = "<<cur_time<<" completion time = 
"<<v[i].completionTime<<endl;
        cur_time+=v[i].burstTime;
    }
}

int main(){

    cout<<"FIRST COME FIRST SERVE CPU SCHEDULING ALGORITHM C++ IMPLEMENTATION\n";
    cout<<"Name: Aditya Anand\tRoll No.:20124009\t Branch: IT\n\n\n";



    int n=0;
    cout<<"Enter the number of processes: ";
    cin>>n;

    cout<<"Enter the arrival times and burst times of "<<n<<" processes: \n";

    vector<process> v(n);
    for(int i=0; i<n; i++){
        cin>>v[i].arrivalTime>>v[i].burstTime;
        v[i].id = i;
    }

    cout<<"-------------------------------------------------------------------------------\n";
    cout<<"\n";
    FCFS(v);
    cout<<"\n";
    cout<<"-------------------------------------------------------------------------------\n";
    cout<<"\n\n";

    int t_TAT=0;
    int t_CT=0;
    for(int i=0; i<n; i++){
        v[i].TAT = v[i].completionTime-v[i].arrivalTime;
        v[i].WT = v[i].TAT-v[i].burstTime;
        t_TAT+=v[i].TAT;
        t_CT+=v[i].completionTime;
    }

    for(auto p:v){
        cout<<"Process: "<<p.id<<"\tArrival Time:"<<p.arrivalTime<<"\tBurst
Time:"<<p.burstTime<<"\tCompletion Time:"<<p.completionTime;
        cout<<"\tTurn Around Time:"<<p.TAT<<"\tWaiting Time:"<<p.WT<<"\tResponse Time:"<<p.RT<<"\n";
    }

    cout<<"\nAverage Turn Around Time: "<<(float)((1.0*t_TAT)/(1.0*n))<<"\n";
    cout<<"\nAverage Completion Time: "<<(float)((1.0*t_CT)/(1.0*n))<<"\n";

    return 0;
}
```

RESULT:



```
PS C:\Users\beadi\Desktop\OS LAB\Assignment 3> cd "c:\Users\beadi\Desktop\OS LAB\Assignment 3\" ; if ($?) { g++ FCFS.cpp -o FCFS } ;
FIRST COME FIRST SERVE CPU SCHEDULING ALGORITHM C++ IMPLEMENTATION
Name: Aditya Anand      Roll No.:20124009        Branch: IT


Enter the number of processes: 4
Enter the arrival times and burst times of 4 processes:
0 2
1 2
5 3
6 4
-------------------------------------------------------------------------------

Process P1: start time = 0 completion time = 2
Process P2: start time = 2 completion time = 4
CPU idle from 4 to 5
Process P3: start time = 5 completion time = 8
Process P4: start time = 8 completion time = 12

-------------------------------------------------------------------------------


Process: 0      Arrival Time:0  Burst Time:2    Completion Time:2       Turn Around Time:2      Waiting Time:0  Response Time:0
Process: 1      Arrival Time:1  Burst Time:2    Completion Time:4       Turn Around Time:3      Waiting Time:1  Response Time:1
Process: 2      Arrival Time:5  Burst Time:3    Completion Time:8       Turn Around Time:3      Waiting Time:0  Response Time:0
Process: 3      Arrival Time:6  Burst Time:4    Completion Time:12      Turn Around Time:6      Waiting Time:2  Response Time:2

Average Turn Around Time: 3.5

Average Completion Time: 6.5
```

PRIORITY BASED CPU SCHEDULING

CRITERIA: Priority (higher the value, greater the priority)

NOTE: In case of same priority, process with lower arrival time is executed first.

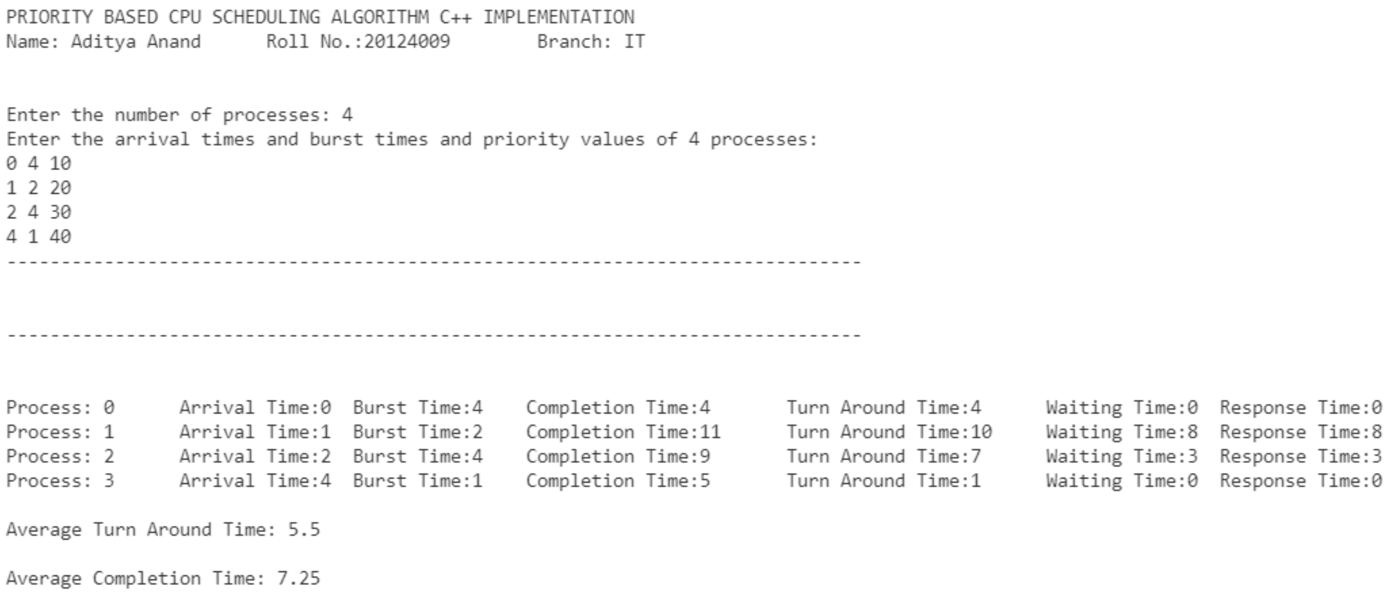
MODE: Non pre-emptive

GIVEN: List of processes with their arrival and burst time.

CODE:

```cpp
#include <bits/stdc++.h>
using namespace ::std;

class process{
    public:
        int priority;
        int id;
        int arrivalTime;
        int burstTime;
        bool ready;
        int completionTime;
        int TAT;
        int WT;
        int RT;
};

struct comp{
    bool operator()(process const &p1, process const &p2){
        return p1.priority < p2.priority;
    }
};

void PriorityBasedScheduling(vector<process> &v){
    priority_queue<process, vector<process>, comp> p;
    int cur_time = INT_MAX;
    int n=v.size();
    for(int i=0; i<n; i++){
        cur_time = min(cur_time, v[i].arrivalTime);
    }

    int count = 0;
    while(true){
        for(int i=0; i<n; i++){
            if(v[i].arrivalTime<=cur_time && !v[i].ready){
                v[i].ready = true;
                p.push(v[i]);
                count++;
            }
        }
        if(count<n && p.empty()){
            cout<<"CPU empty from "<<cur_time<<" to "<<cur_time+1<<"\n";
            cur_time++;
            continue;
        }
        if(p.empty()){
            break;
        }

        process cur_process = p.top();
        p.pop();

        v[cur_process.id].RT = cur_time-cur_process.arrivalTime;
        cur_time+=cur_process.burstTime;
        v[cur_process.id].completionTime=cur_time;
    }
}

int main(){

    cout << "PRIORITY BASED CPU SCHEDULING ALGORITHM C++ IMPLEMENTATION\n";
    cout << "Name: Aditya Anand\tRoll No.:20124009\t Branch: IT\n\n\n";

    int n = 0;
    cout << "Enter the number of processes: ";
    cin >> n;

    cout << "Enter the arrival times and burst times and priority values of " << n << " processes: \n";

    vector<process> v(n);
    for (int i = 0; i < n; i++){
        cin >> v[i].arrivalTime >> v[i].burstTime >> v[i].priority;
        v[i].ready = false;
        v[i].id = i;
    }

    cout << "------------------------------------------------------------------------------\n";
    cout << "\n";
    PriorityBasedScheduling(v);
    cout << "\n";
    cout << "------------------------------------------------------------------------------\n";
    cout << "\n\n";

    int t_TAT = 0;
    int t_CT = 0;
    for (int i = 0; i < n; i++){
        v[i].TAT = v[i].completionTime - v[i].arrivalTime;
        v[i].WT = v[i].TAT - v[i].burstTime;
        t_TAT += v[i].TAT;
        t_CT += v[i].completionTime;
    }

    for (auto p : v){
        cout << "Process: " << p.id << "\tArrival Time:" << p.arrivalTime << "\tBurst Time:" <<
p.burstTime << "\tCompletion Time:" << p.completionTime;
        cout << "\tTurn Around Time:" << p.TAT << "\tWaiting Time:" << p.WT << "\tResponse Time:" << p.RT
<< "\n";
    }
    cout << "\nAverage Turn Around Time: " << (float)((1.0 * t_TAT) / (1.0 * n)) << "\n";
    cout << "\nAverage Completion Time: " << (float)((1.0 * t_CT) / (1.0 * n)) << "\n";

    return 0;
}
```

RESULT:



```
PRIORITY BASED CPU SCHEDULING ALGORITHM C++ IMPLEMENTATION
Name: Aditya Anand      Roll No.:20124009        Branch: IT


Enter the number of processes: 4
Enter the arrival times and burst times and priority values of 4 processes:
0 4 10
1 2 20
2 4 30
4 1 40
-------------------------------------------------------------------------------



-------------------------------------------------------------------------------



Process: 0      Arrival Time:0  Burst Time:4    Completion Time:4       Turn Around Time:4      Waiting Time:0  Response Time:0
Process: 1      Arrival Time:1  Burst Time:2    Completion Time:11      Turn Around Time:10     Waiting Time:8  Response Time:8
Process: 2      Arrival Time:2  Burst Time:4    Completion Time:9       Turn Around Time:7      Waiting Time:3  Response Time:3
Process: 3      Arrival Time:4  Burst Time:1    Completion Time:5       Turn Around Time:1      Waiting Time:0  Response Time:0

Average Turn Around Time: 5.5

Average Completion Time: 7.25
```