U18CO018
Shubham Shekhaliya
Assignment – 5
Operating System

Write a c/Java program for simulation of

    (1). Shortest Job First (SJF)
    (2). Shortest Remaining Time First (SRTF) CPU scheduler.

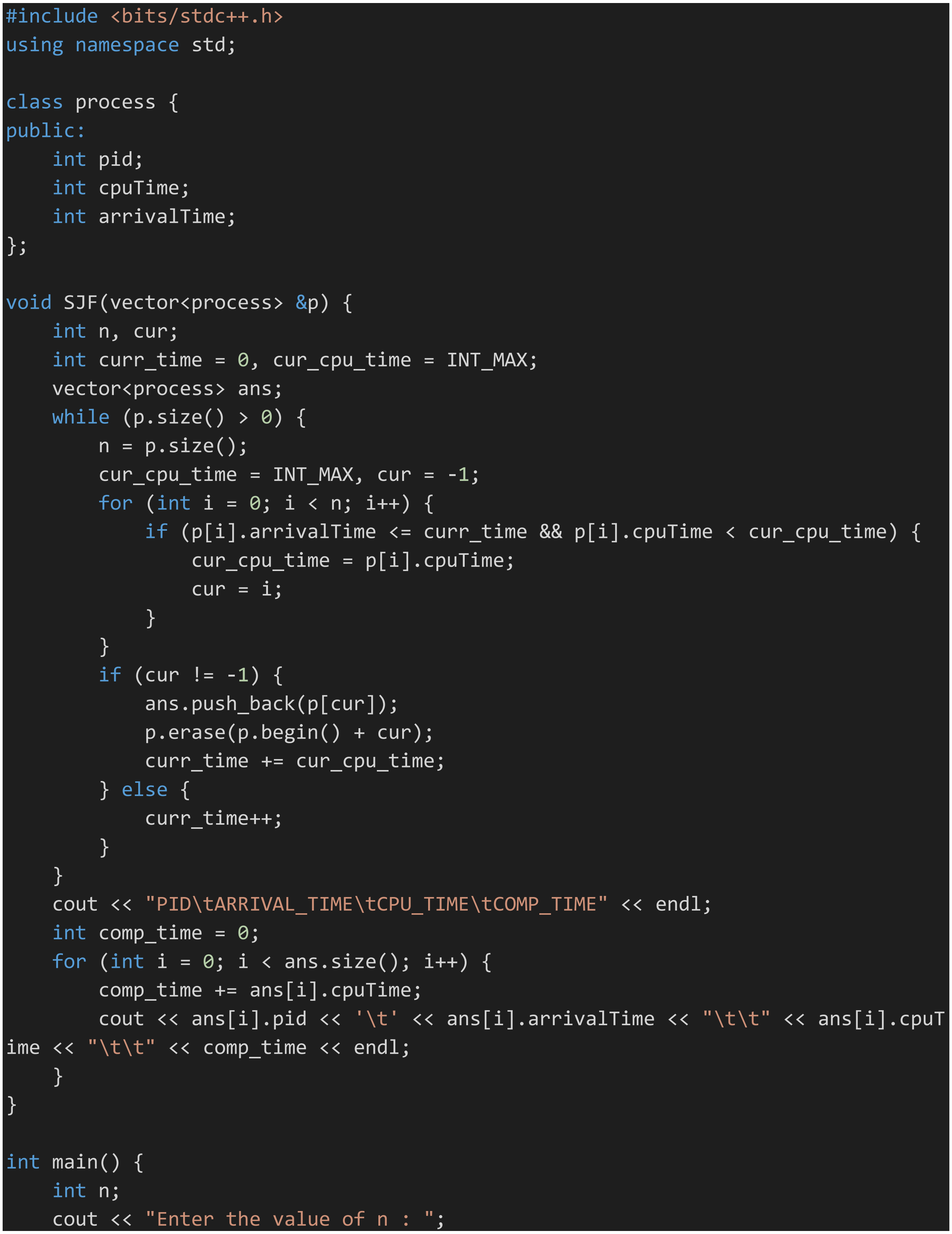
Program should maintain Ready_Q using process pointers. Each Process should have cpu_time and arrival_time . Cpu_time and arrival_time should be generated randomly. Demonstrate processes context switch according to SRTF Scheduling.

```
Struct process
{
Int pid;
Int cpu_time;
Int arrival_time;
Struct process * next;
}
```

```
Void main()
{
Initialize
/*
Generate n no of process.
Assign process_id serially to each process.
Randomly generate cpu_time and arrival_time for each process.
Create &amp; initialize Ready_Q with n no of processes.
*/
Untill Ready_Q <> empy
{
select Queue Ready_Q
Randomly generate current_time requirement out of total_time requirement for corresponding process present on the front of corresponding queue.
Call delay for current_time no of times.
Subtract current_time from total_cpu_time &amp; update total_cpu_time
Shift process to another queue.

displyQueue.
}
```

SJF Algorithm :-



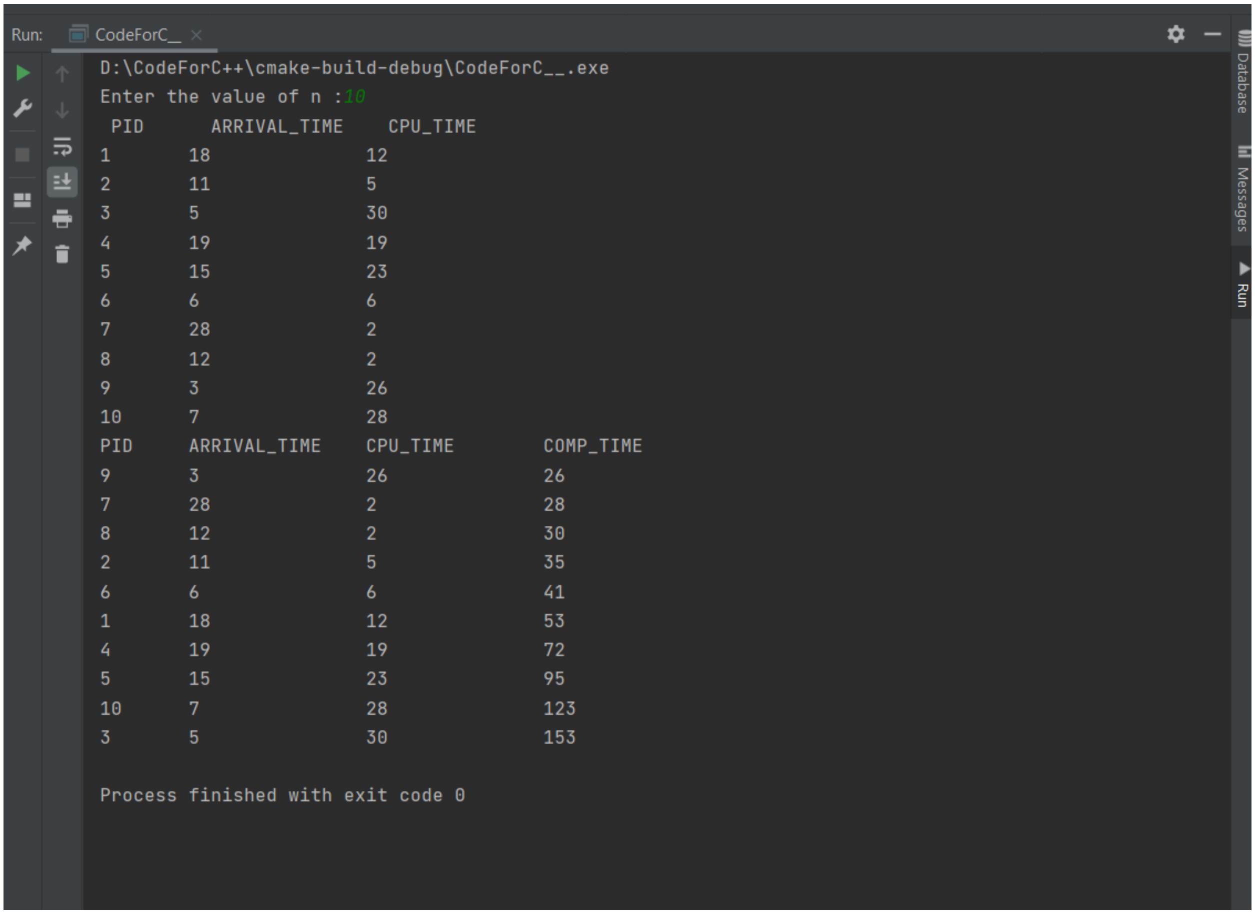
```
#include <bits/stdc++.h>
using namespace std;

class process {
public:
    int pid;
    int cpuTime;
    int arrivalTime;
};

void SJF(vector<process> &p) {
    int n, cur;
    int curr_time = 0, cur_cpu_time = INT_MAX;
    vector<process> ans;
    while (p.size() > 0) {
        n = p.size();
        cur_cpu_time = INT_MAX, cur = -1;
        for (int i = 0; i < n; i++) {
            if (p[i].arrivalTime <= curr_time && p[i].cpuTime < cur_cpu_time) {
                cur_cpu_time = p[i].cpuTime;
                cur = i;
            }
        }
        if (cur != -1) {
            ans.push_back(p[cur]);
            p.erase(p.begin() + cur);
            curr_time += cur_cpu_time;
        } else {
            curr_time++;
        }
    }
    cout << "PID\tARRIVAL_TIME\tCPU_TIME\tCOMP_TIME" << endl;
    int comp_time = 0;
    for (int i = 0; i < ans.size(); i++) {
        comp_time += ans[i].cpuTime;
        cout << ans[i].pid << '\t' << ans[i].arrivalTime << "\t\t" << ans[i].cpuT
ime << "\t\t" << comp_time << endl;
    }
}

int main() {
    int n;
    cout << "Enter the value of n : ";
    cin >> n;
    vector<process> init(n);
    cout << "PID\tARRIVAL_TIME\tCPU_TIME" << endl;
    for (int i = 0; i < n; i++) {
        init[i].pid = i + 1;
        init[i].cpuTime = rand() % (n * 3) + 1;
        init[i].arrivalTime = rand() % (n * 3) + 1;
        cout << init[i].pid << "\t" << init[i].arrivalTime << "\t\t" << init[i].cpuTime << endl;
    }
    SJF(init);
}
```

Output:-

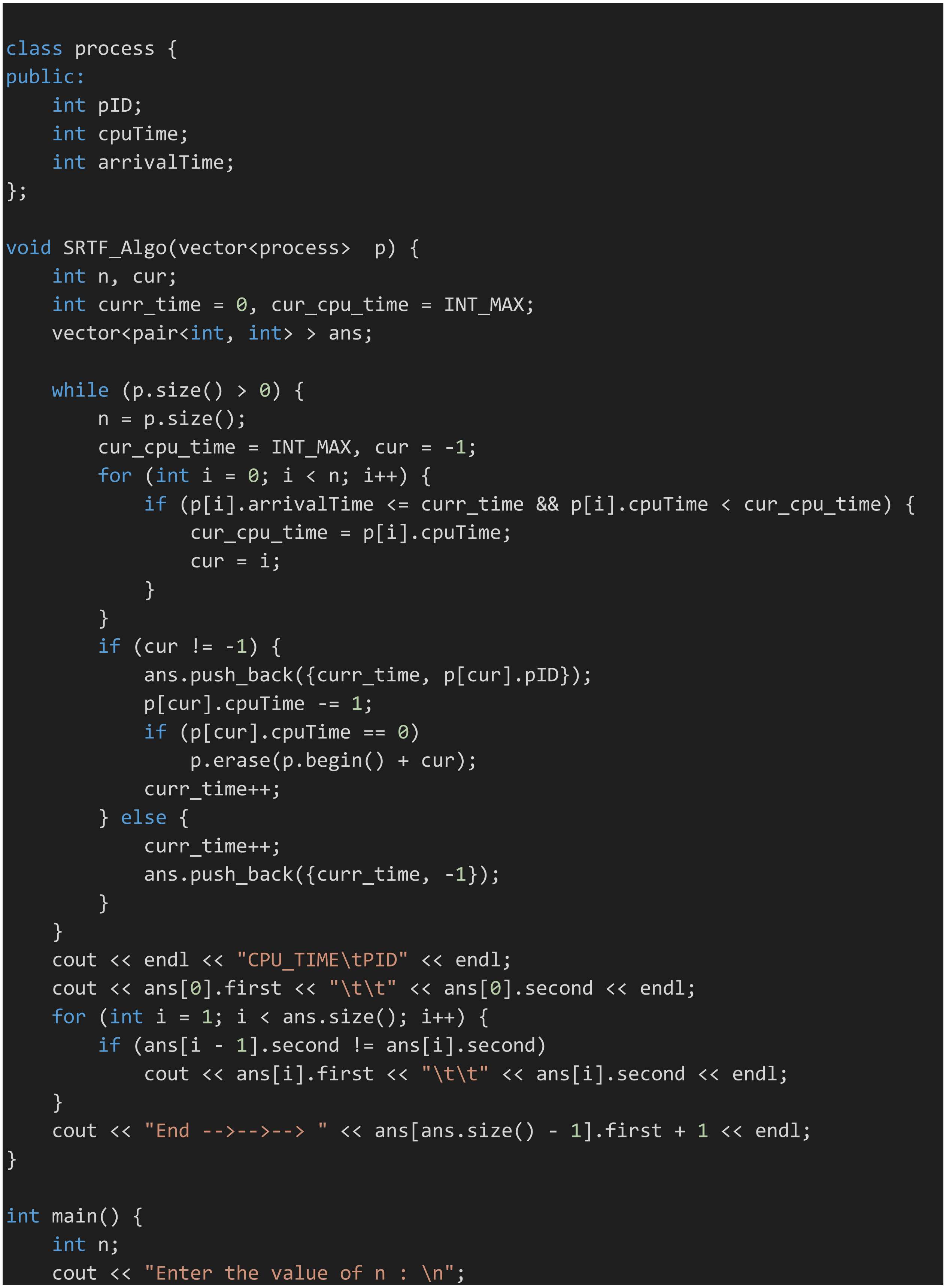


```
D:\CodeForC++\cmake-build-debug\CodeForC__.exe
Enter the value of n :10
 PID     ARRIVAL_TIME    CPU_TIME
1       18              12
2       11              5
3       5               30
4       19              19
5       15              23
6       6               6
7       28              2
8       12              2
9       3               26
10      7               28
PID     ARRIVAL_TIME    CPU_TIME        COMP_TIME
9       3               26              26
7       28              2               28
8       12              2               30
2       11              5               35
6       6               6               41
1       18              12              53
4       19              19              72
5       15              23              95
10      7               28              123
3       5               30              153

Process finished with exit code 0
```

SRTF Algorithm:-

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



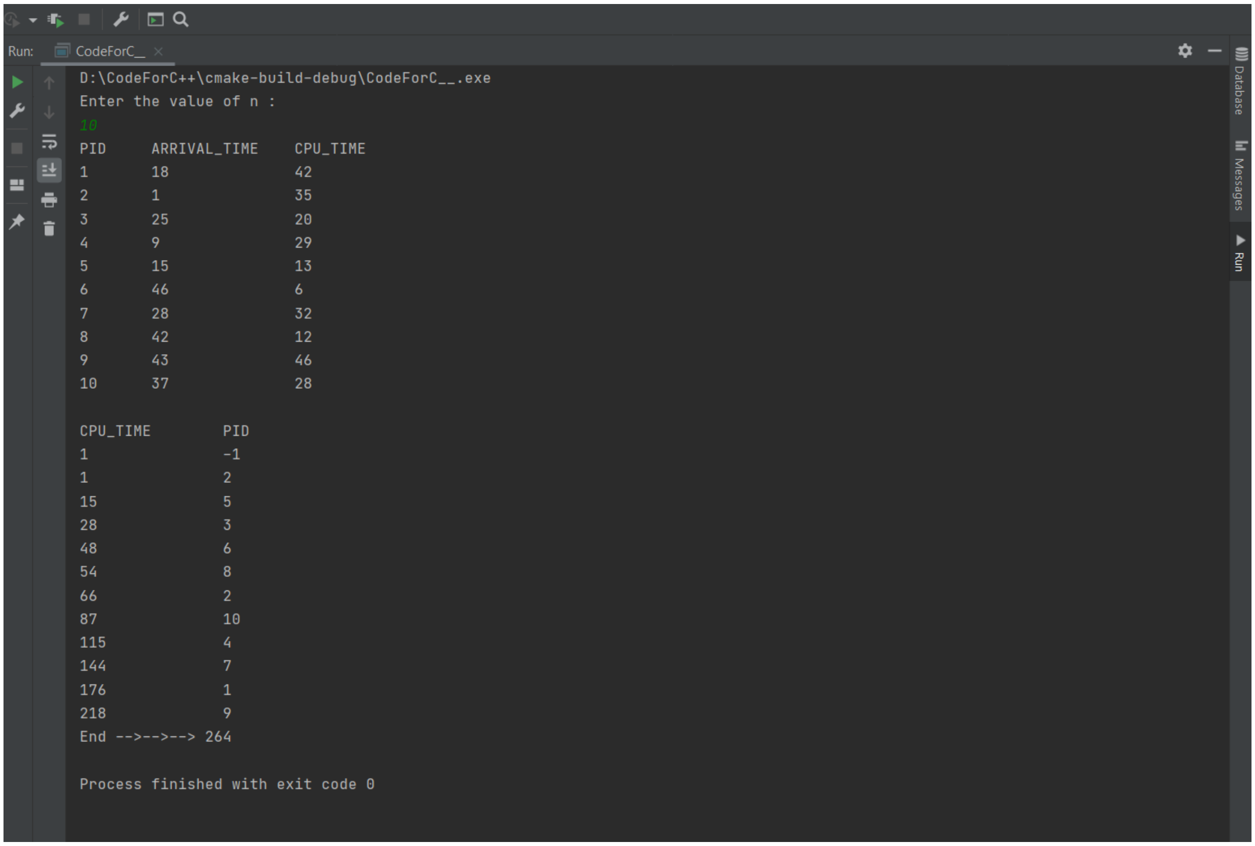
```
class process {
public:
    int pID;
    int cpuTime;
    int arrivalTime;
};

void SRTF_Algo(vector<process>  p) {
    int n, cur;
    int curr_time = 0, cur_cpu_time = INT_MAX;
    vector<pair<int, int> > ans;

    while (p.size() > 0) {
        n = p.size();
        cur_cpu_time = INT_MAX, cur = -1;
        for (int i = 0; i < n; i++) {
            if (p[i].arrivalTime <= curr_time && p[i].cpuTime < cur_cpu_time) {
                cur_cpu_time = p[i].cpuTime;
                cur = i;
            }
        }
        if (cur != -1) {
            ans.push_back({curr_time, p[cur].pID});
            p[cur].cpuTime -= 1;
            if (p[cur].cpuTime == 0)
                p.erase(p.begin() + cur);
            curr_time++;
        } else {
            curr_time++;
            ans.push_back({curr_time, -1});
        }
    }
    cout << endl << "CPU_TIME\tPID" << endl;
    cout << ans[0].first << "\t\t" << ans[0].second << endl;
    for (int i = 1; i < ans.size(); i++) {
        if (ans[i - 1].second != ans[i].second)
            cout << ans[i].first << "\t\t" << ans[i].second << endl;
    }
    cout << "End -->-->--> " << ans[ans.size() - 1].first + 1 << endl;
}

int main() {
    int n;
    cout << "Enter the value of n : \n";
    cin >> n;
    vector<process> init(n);
    cout << "PID\tARRIVAL_TIME\tCPU_TIME" << endl;
    for (int i = 0; i < n; i++) {
        init[i].pID = i + 1;
        init[i].cpuTime = rand() % (n * 5) + 1;
        init[i].arrivalTime = rand() % (n * 5) + 1;
        cout << init[i].pID << '\t' << init[i].arrivalTime << "\t\t" << init[i].cpuTime << endl;
    }
    SRTF_Algo(init);
}
```

Output:-



```
D:\CodeForC++\cmake-build-debug\CodeForC__.exe
Enter the value of n :
10
PID     ARRIVAL_TIME    CPU_TIME
1       18              42
2       1               35
3       25              20
4       9               29
5       15              13
6       46              6
7       28              32
8       42              12
9       43              46
10      37              28

CPU_TIME        PID
1               -1
1               2
15              5
28              3
48              6
54              8
66              2
87              10
115             4
144             7
176             1
218             9
End -->-->--> 264

Process finished with exit code 0
```