```cpp
#include <iostream>
#include <algorithm>
#include <iomanip>
using namespace std;

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


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

bool compareID(process p1, process p2)
{
    return p1.pid < p2.pid;
}
```

```
int main() {

    int n;
    struct process p[100];
    float avg_turnaround_time;
    float avg_waiting_time;
    float avg_response_time;
    float cpu_utilisation;
    int total_turnaround_time = 0;
    int total_waiting_time = 0;
    int total_response_time = 0;
    int total_idle_time = 0;
    float throughput;

    cout << setprecision(2) << fixed;


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

    for(int i = 0; i < n; i++) {
        cout<<"Enter arrival time of process "<<i+1<<": ";
        cin>>p[i].arrival_time;
        cout<<"Enter burst time of process "<<i+1<<": ";
        cin>>p[i].burst_time;
        p[i].pid = i+1;
        cout<<endl;
    }

    sort(p,p+n,compareArrival);

for(int i = 0; i < n; i++) {
    p[i].start_time = (i == 0)?p[i].arrival_time:max(p[i-1].completion_time,p[i].arrival_time);
    p[i].completion_time = p[i].start_time + p[i].burst_time;
    p[i].turnaround_time = p[i].completion_time - p[i].arrival_time;
    p[i].waiting_time = p[i].turnaround_time - p[i].burst_time;
    p[i].response_time = p[i].start_time - p[i].arrival_time;

    total_turnaround_time += p[i].turnaround_time;
    total_waiting_time += p[i].waiting_time;
    total_response_time += p[i].response_time;
    total_idle_time += (i == 0)?(p[i].arrival_time):(p[i].start_time - p[i-1].completion_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;

    cpu_utilisation = ((p[n-1].completion_time - total_idle_time) / (float) p[n-1].completion_time)*100;

    throughput = float(n) / (p[n-1].completion_time - p[0].arrival_time);


    sort(p,p+n,compareID);


    cout<<endl;

    cout<<"#P₩t"<<"AT₩t"<<"BT₩t"<<"ST₩t"<<"CT₩t"<<"TAT₩t"<<"WT₩t"<<"RT₩t"<<"₩n"<<endl;


    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].completion_time<<"₩t"<<p[i].turnaround_time<<"₩t"<<p[i].waiting_time<<"₩t"<<p[i].response_time<<"₩t"<<"₩n"<<endl;

    }
     cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;
     cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;
     cout<<"Average Response Time = "<<avg_response_time<<endl;
     cout<<"CPU Utilization = "<<cpu_utilisation<<"%"<<endl;
     cout<<"Throughput = "<<throughput<<" process/unit time"<<endl;


}

/*

AT - Arrival Time of the process
BT - Burst time of the process
ST - Start time of the process
CT - Completion time of the process
TAT - Turnaround time of the process
WT - Waiting time of the process
RT - Response time of the process

Formulas used:

TAT = CT - AT
WT = TAT - BT
RT = ST - AT

*/
```