```
//Gaurav Rai 1706019

#include<bits/stdc++.h>
using namespace std;
class PCB
{
    public :
    static float avgWt,avgTat,avgCt;
    int pid;
    int pri;
    bool vis;
    float bt,at,ct,tat,wt;
    bool operator ==(const PCB p) const
    {
        if(at!=p.at || bt!=p.bt ||pri!=p.pri)
        return false;
        return true;
    }
};
float PCB::avgWt =0;
float PCB::avgTat =0;
float PCB::avgCt =0;
class less_than_at
{
    public:
    inline bool operator() (const PCB& struct1, const PCB& struct2)
    {
        return (struct1.at < struct2.at);
    }
};

class less_than_bt
{
    public:
    inline bool operator() (const PCB& struct1, const PCB& struct2)
    {
        return (struct1.bt < struct2.bt);
    }
};
void SJF(deque<PCB> &P,deque<PCB> &ganttChart,int n)
{
    P[0].ct=P[0].at+P[0].bt;
    P[0].tat=P[0].ct-P[0].at;
    P[0].wt=0;
    P[0].vis=true;
    ganttChart.push_back(P[0]);
    int countt=1;
    P.pop_front();
    while(countt<n)
    {
        PCB temp=ganttChart.back();
        vector<PCB> t;bool flag=false;
        for(int i=0;i<P.size();i++)
        {
            if(P[i].at<=temp.ct)
            {  t.push_back(P[i]); flag=true; }
        }
        if(flag)
        {
            sort(t.begin(),t.end(),less_than_bt());
            int i=0;
            if(t[0].at<=temp.ct)
            {
                t[i].ct=temp.ct+t[i].bt;
                t[i].tat=t[i].ct-t[i].at;
                t[i].wt=t[i].tat-t[i].bt;
            }
            else
            {
                t[i].ct=t[i].at+t[i].bt;
                t[i].tat=t[i].ct-t[i].at;
                t[i].wt=t[i].tat-t[i].bt;
            }
            ganttChart.push_back(t[i]);
            for(i=0;i<P.size();i++)
            {
                if(t[0]==P[i])
                {
                    flag=false;
                    break;
                }
            }
            if(flag)
            cout<<"Error\n";
            P.erase(P.begin()+i);
        }
        else
        {
            int i=0;t.push_back(P[0]);
            t[i].ct=t[i].at+t[i].bt;
            t[i].tat=t[i].ct-t[i].at;
            t[i].wt=t[i].tat-t[i].bt;
            ganttChart.push_back(t[i]);
            P.pop_front();
        }
            countt++;
    }
}
int main()
{
    int n;
    cout<<"Enter the number of processes ...";cin>>n;
    deque<PCB> P(n);

    for(int i=0;i<n;i++)
    {
        cout<<"Process Id : ";cin>>P[i].pid;
        cout<<"Arrival Time : ";cin>>P[i].at;
        cout<<"Burst Time : ";cin>>P[i].bt;
        P[i].vis=false;
    }
    sort(P.begin(),P.end(),less_than_at());
    deque<PCB> ganttChart;
    SJF(P,ganttChart,n);
    P.clear();
    copy(ganttChart.begin(), ganttChart.end(), back_inserter(P));
    for(int i=0;i<P.size();i++)
    {
        PCB::avgWt +=P[i].wt;
        PCB::avgTat+=P[i].tat;
        PCB::avgCt+=P[i].ct;
    }
    PCB::avgWt/=P.size();
    PCB::avgTat/=P.size();
    PCB::avgCt/=P.size();
    cout<<"\tSJF(Non Preemptive) CPU SCHEDULING\n";
    cout<<"PID\tAT\tBT\tCT\tTAT\tWT\n";
    for(int i=0;i<n;i++)

cout<<P[i].pid<<"\t"<<P[i].at<<"\t"<<P[i].bt<<"\t"<<P[i].ct<<"\t"<<P[i].tat<<"\t"<<
P[i].wt<<endl;
    cout<<"\tAverage Waiting Time : "<<PCB::avgWt<<endl;
    cout<<"\tAverage TurnAround Time : "<<PCB::avgTat<<endl;
    cout<<"\tAverage Completion Time : "<<PCB::avgCt<<endl;

    return 0;
}
```