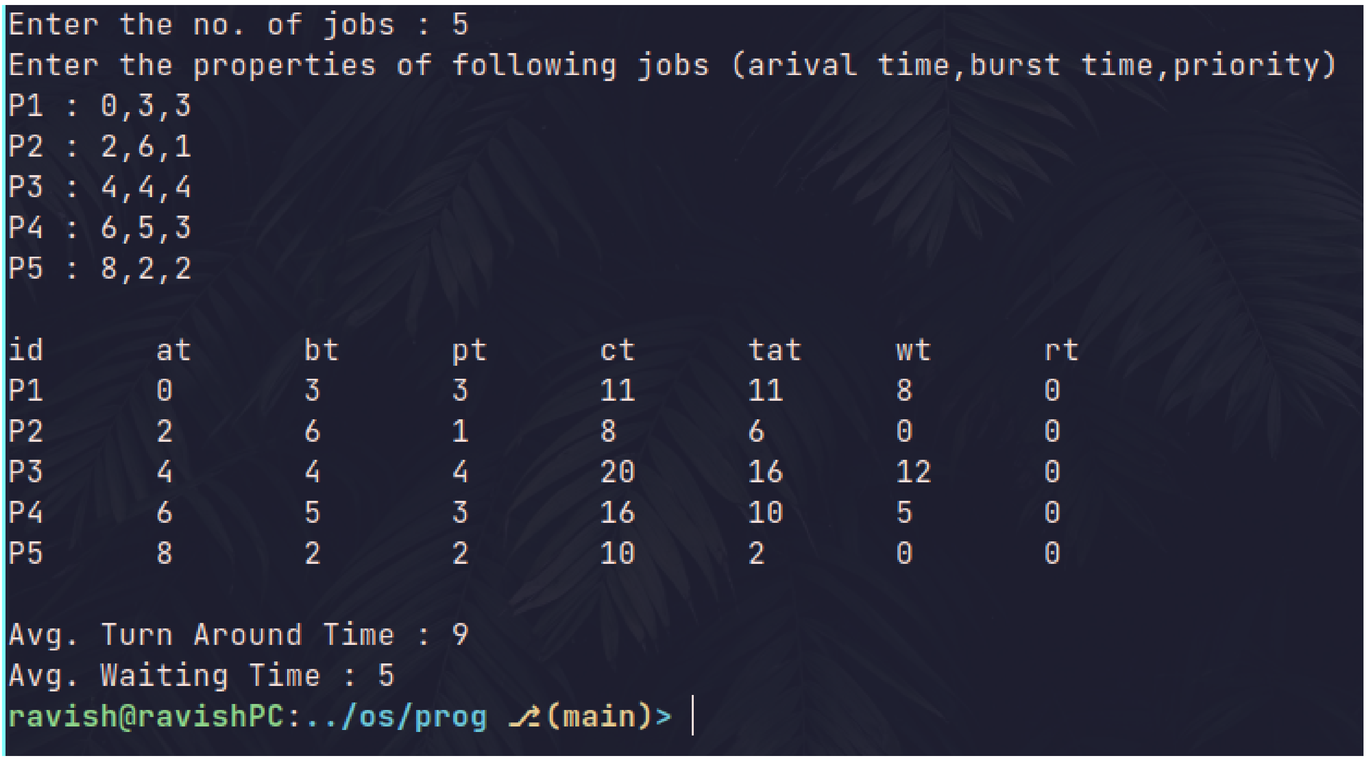
Assignment 5

```
Name : Ravish Ranjan
Course : MCA
Semester : 2nd semester
```

```
Q1: Pre-Emptive Priority Scheduling
Write a C/C++ program that:
1.Reads the number of processes (Sample input)
2. Accepts arrival time, burst time and priority for each process (Sample
input)
3. Schedules processes using Pre-Emptive Priority Scheduling
4. Calculates WT and TAT
5. Displays results in tabular form
6. Calculate and display Average WT and TAT
```

```cpp
#ifndef SCHEDULER_CPP
#define SCHEDULER_CPP

#include <iostream>
#include <vector>
#include <sstream>
#include <algorithm>


std::vector<std::string> split(std::string inp,char sep = ' '){
    std::stringstream ss(inp);
    std::string segment;
    std::vector<std::string> results;

    while (std::getline(ss, segment, sep)) results.push_back(segment);

    return results;
}

class Job{
    public:
        int id;
        int at;
        int bt;
        int pt;
        int ct = 0;
        int tat = 0;
        int wt = 0;
        int rt = 0;
        Job(int id,int bt,int at = 0,int pt =
0):id(id),bt(bt),at(at),pt(pt){}
};

class Scheduler{
    public:
        virtual void apply(std::vector<Job>& jobs) = 0;
        void printTable(const std::vector<Job>& jobs){
            std::cout << "\nid\tat\tbt\tpt\tct\ttat\twt\trt" << std::endl;
            for (Job job:jobs){

printf("P%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",job.id,job.at,job.bt,job.pt,job.
ct,job.tat,job.wt,job.rt);
            }
            std::cout << std::endl;
        }
        void getAvgs(const std::vector<Job>& jobs){
            double avgTat,avgWt,avgRt;
            for(Job job:jobs){
                avgTat += job.tat;
                avgWt += job.wt;
                avgRt += job.rt;
            }
            std::cout << "Avg. Turn Around Time : " <<
(double)avgTat/jobs.size() << std::endl;
            std::cout << "Avg. Waiting Time : " <<
(double)avgWt/jobs.size() << std::endl;
            // std::cout << "Avg. Response Time : " <<
(double)avgRt/jobs.size() << std::endl;
        }
};

#endif
```

```
#include "scheduler.cpp"
#include <queue>

class PPS:public Scheduler{
    public:
        void apply(std::vector<Job>& jobs){
            std::sort(jobs.begin(),jobs.end(),[](const Job& j1,const Job&
j2){
                return j1.at < j2.at;
            });

            std::vector<int> rem_bt(jobs.size());
            for (int i = 0; i < jobs.size(); i++){
                rem_bt[i] = jobs[i].bt;
            }
            int at = 0;
            int completed = 0;

            while (completed < jobs.size()){
                int min_idx = -1;
                int min_pt = __INT_MAX__;

                for (int i = 0; i < jobs.size(); i++){
                    if (jobs[i].at <= at && rem_bt[i] > 0 && jobs[i].pt <
min_pt){
                        min_pt = jobs[i].pt;
                        min_idx = i;
                    }
                }

                at++;

                if (min_pt > -1) {
                    rem_bt[min_idx]--;;
                    if (rem_bt[min_idx] <= 0){
                        jobs[min_idx].ct = at;
                        jobs[min_idx].tat = jobs[min_idx].ct -
jobs[min_idx].at;
                        jobs[min_idx].wt = jobs[min_idx].tat -
jobs[min_idx].bt;
                        completed++;
                    }
                }
            }
        }
};

int main(){
    int n = 0;
    std::string inp = "";
    std::cout << "Enter the no. of jobs : ";
    std::cin >> n;

    std::vector<Job> jobs_pps;
    std::cout << "Enter the properties of following jobs (arival time,burst
time,priority)" << std::endl;

    for (int i = 0; i < n; i++){
        std::cout << "P" << i+1 << " : ";
        std::cin >> inp;
        std::vector<std::string> parts = split(inp,',');


jobs_pps.push_back(Job(i+1,std::stoi(parts[1]),std::stoi(parts[0]),std::sto
i(parts[2])));
    }

    PPS pps;
    pps.apply(jobs_pps);
    pps.printTable(jobs_pps);
    pps.getAvgs(jobs_pps);

    return 0;
}
```



```
Enter the no. of jobs : 5
Enter the properties of following jobs (arival time,burst time,priority)
P1 : 0,3,3
P2 : 2,6,1
P3 : 4,4,4
P4 : 6,5,3
P5 : 8,2,2

id      at      bt      pt      ct      tat     wt      rt
P1      0       3       3       11      11      8       0
P2      2       6       1       8       6       0       0
P3      4       4       4       20      16      12      0
P4      6       5       3       16      10      5       0
P5      8       2       2       10      2       0       0

Avg. Turn Around Time : 9
Avg. Waiting Time : 5
ravish@ravishPC:../os/prog ⎇(main)>
```