```cpp
#include <iostream>
#include <algorithm>
#include <iomanip>
#include <string.h>
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;
};

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;
    int is_completed[100];
    memset(is_completed,0,sizeof(is_completed));

    //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;
    }

    int current_time = 0;
    int completed = 0;
    int prev = 0;

    while(completed != n) {
        int idx = -1;
        int max= -1;
        for(int i = 0; i < n; i++) {
            if(p[i].arrival_time <= current_time && is_completed[i] == 0) {
                if(p[i].burst_time > max) {
                    max = p[i].burst_time;
                    idx = i;
                }
                if(p[i].burst_time == max) {
                    if(p[i].arrival_time < p[idx].arrival_time) {
                        max = p[i].burst_time;
                        idx = i;
                    }
                }
            }
        }
        if(idx != -1) {
            p[idx].start_time = current_time;
            p[idx].completion_time = p[idx].start_time + p[idx].burst_time;
            p[idx].turnaround_time = p[idx].completion_time -
p[idx].arrival_time;
            p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
            p[idx].response_time = p[idx].start_time - p[idx].arrival_time;

            total_turnaround_time += p[idx].turnaround_time;
            total_waiting_time += p[idx].waiting_time;
            total_response_time += p[idx].response_time;
            total_idle_time += p[idx].start_time - prev;

            is_completed[idx] = 1;
            completed++;
            current_time = p[idx].completion_time;
            prev = current_time;
        }
        else {
            current_time++;
        }

    }

    int min_arrival_time = 10000000;
    int max_completion_time = -1;
    for(int i = 0; i < n; i++) {
```

```
        min_arrival_time = min(min_arrival_time,p[i].arrival_time);
        max_completion_time = max(max_completion_time,p[i].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 = ((max_completion_time - total_idle_time) / (float)
max_completion_time )*100;
    throughput = float(n) / (max_completion_time - min_arrival_time);

    cout<<endl<<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].st
art_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;


}
```

OUTPUT:-

```
PS C:\Users\AJAY SHARMA\Desktop\os> cd "c:\Users\AJAY SHARMA\Desktop\os\" ;
ljf } ; if ($?)                    > cd "c:\Users\AJAY SHARMA\Desktop\os\" ;
ljf } ; if ($?) { .\ljf }
Enter the number of processes: 4
Enter arrival time of process 1: 1
Enter burst time of process 1: 2

Enter arrival time of process 2: 2
Enter burst time of process 2: 4

Enter arrival time of process 3: 3
Enter burst time of process 3: 6

Enter arrival time of process 4: 4
Enter burst time of process 4: 8
```

```
 #P      AT      BT      ST      CT      TAT     WT      RT

 1       1       2       1       3       2       0       0

 2       2       4       17      21      19      15      15

 3       3       6       3       9       6       0       0

 4       4       8       9       17      13      5       5

 Average Turnaround Time = 10
 Average Waiting Time = 5
 Average Response Time = 5
 CPU Utilization = 95.2381%
 Throughput = 0.2 process/unit time
 PS C:\Users\AJAY SHARMA\Desktop\os>
```