```cpp
#include <iostream>
#include <algorithm>
#include <iomanip>
#include <string.h>
#include <queue>
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 compare1(process p1, process p2)
{
    return p1.arrival_time < p2.arrival_time;
}

bool compare2(process p1, process p2)
{
    return p1.pid < p2.pid;
}

int main() {

    int n;
    int tq;
    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 burst_remaining[100];
    int idx;

    cout << setprecision(2) << fixed;

    cout<<"Enter the number of processes: ";
    cin>>n;
    cout<<"Enter time quantum: ";
    cin>>tq;
```

```
    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;
        burst_remaining[i] = p[i].burst_time;
        p[i].pid = i+1;
        cout<<endl;
    }

    sort(p,p+n,compare1);

    queue<int> q;
    int current_time = 0;
    q.push(0);
    int completed = 0;
    int mark[100];
    memset(mark,0,sizeof(mark));
    mark[0] = 1;

    while(completed != n) {
        idx = q.front();
        q.pop();

        if(burst_remaining[idx] == p[idx].burst_time) {
            p[idx].start_time = max(current_time,p[idx].arrival_time);
            total_idle_time += p[idx].start_time - current_time;
            current_time = p[idx].start_time;
        }

        if(burst_remaining[idx]-tq > 0) {
            burst_remaining[idx] -= tq;
            current_time += tq;
        }
        else {
            current_time += burst_remaining[idx];
            burst_remaining[idx] = 0;
            completed++;

            p[idx].completion_time = current_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;
        }
```

```
        for(int i = 1; i < n; i++) {
            if(burst_remaining[i] > 0 && p[i].arrival_time <= current_time &&
mark[i] == 0) {
                q.push(i);
                mark[i] = 1;
            }
        }
        if(burst_remaining[idx] > 0) {
            q.push(idx);
        }

        if(q.empty()) {
            for(int i = 1; i < n; i++) {
                if(burst_remaining[i] > 0) {
                    q.push(i);
                    mark[i] = 1;
                    break;
                }
            }
        }


    }

    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,compare2);

    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].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\"
.cpp -o round_robin } ; if ($?) { .\round_robin }
Enter the number of processes: 4
Enter time quantum: 2
Enter arrival time of process 1: 0
Enter burst time of process 1: 5

Enter arrival time of process 2: 1
Enter burst time of process 2: 4

Enter arrival time of process 3: 2
Enter burst time of process 3: 2

Enter arrival time of process 4: 4
Enter burst time of process 4: 1
```

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

1       0       5       0       12      12      7       0

2       1       4       2       11      10      6       1

3       2       2       4       6       4       2       2

4       4       1       8       9       5       4       4

Average Turnaround Time = 7.75
Average Waiting Time = 4.75
Average Response Time = 1.75
CPU Utilization = 100.00%
Throughput = 0.44 process/unit time
PS C:\Users\AJAY SHARMA\Desktop\os>
```