Lab Report No:	10
Lab Report Name:	Implementation of Round Robin Scheduling Algorithm
ID:	IT-17037

Objective:- To understand the Round Robin scheduling algorithm technique in detail.

To simulate Round Robin Scheduling Algorithm using C/C++.

To understand the advantage and disadvantage of Round Robin scheduling algorithm technique.

Round Robin Scheduling:- Round Robin scheduling algorithm is a type of preemptive type of scheduling used by the operating system for scheduling the processes. In the Round Robin scheduling algorithm, a time quantum is decided which remains constant throughout the execution of all processes. Each process executes only for this much time. If within this time, the process completes its execution, then it is terminated. Else, it waits for its turn for again getting the processor for the same time quantum, and this process continues. Let us understand it further with the help of an example.

Suppose we have 4 processes: P1, P2, P3, and P4, with the following properties:

process	Arrival time	Burst time
P1	0	5
P2	2	4
P3	4	1
P4	5	6

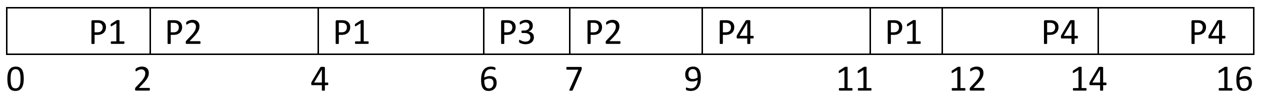
1. Completion Time: Time at which process completes its execution.

2. Turn Around Time: Time Difference between completion time and arrival time. Turn Around Time = Completion Time – Arrival Time
3. Waiting Time (W.T): Time Difference between turn-around time and burst time.
   Waiting Time = Turn Around Time – Burst Time

In this post, we have assumed arrival times as 0, so turn around and completion times are same.

And suppose that the time quantum decided is 2 units. So, the execution of these processes while following the Round Robin Scheduling will be as follows:

Gant Chart:



process	Arrival time	Burst time	Completion time	Turn-around time	Waiting time
P1	0	5	12	12	0
P2	2	4	9	7	3
P3	4	1	7	3	2
P4	5	6	16	11	5

Average waiting time = (0+3+2+5)/4

=2.5ms

Average turn-around time = (12+7+3+11)/4

=8.25ms

Code:-

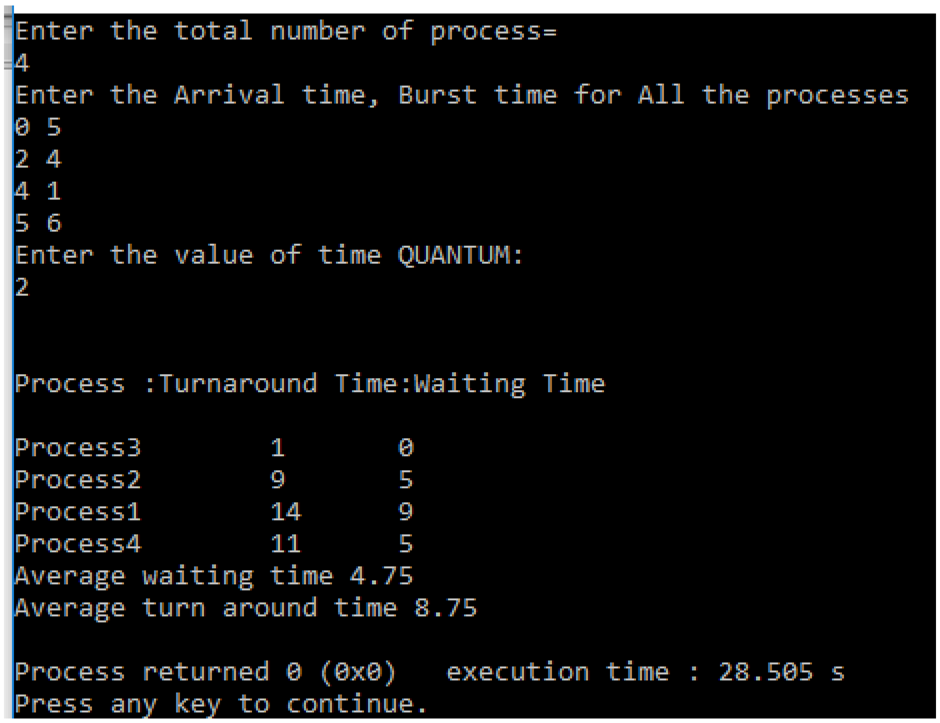
```
#include <iostream>
```

```
#include <vector>

using namespace std;

int main(){
        int i,n,time,remain,temps=0,time_quantum;

        int wt=0,tat=0;

        cout<<"Enter the total number of process="<<endl;
        cin>>n;

        remain=n;
        vector<int>at(n);
        vector<int>bt(n);
        vector<int>rt(n);

        cout<<"Enter the Arrival time, Burst time for All the processes"<<endl;
        for(i=0;i<n;i++)
        {
                cin>>at[i];
                cin>>bt[i];
                rt[i]=bt[i];
        }
```

```
cout<<"Enter the value of time QUANTUM:"<<endl;
cin>>time_quantum;

cout<<"\n\nProcess\t:Turnaround Time:Waiting Time\n\n";
for(time=0,i=0;remain!=0;)
{
        if(rt[i]<=time_quantum && rt[i]>0)
        {
                time += rt[i];
                rt[i]=0;
                temps=1;
        }

        else if(rt[i]>0)
        {
                rt[i] -= time_quantum;
                time += time_quantum;
        }

        if(rt[i]==0 && temps==1)
        {
                remain--;
```

```
                        cout<<"Process"<<i+1<<"\t"<<time-at[i]<<"\t"<<time-at[i]-
bt[i];
                        cout<<endl;

                        wt += time-at[i]-bt[i];
                        tat += time-at[i];
                        temps=0;
                }

                if(i == n-1)
                        i=0;
                else if(at[i+1] <= time)
                        i++;
                else
                        i=0;
        }

        cout<<"Average waiting time "<<wt*1.0/n<<endl;
        cout<<"Average turn around time "<<tat*1.0/n<<endl;;

        return 0;
}
```

Output:-



```
Enter the total number of process=
4
Enter the Arrival time, Burst time for All the processes
0 5
2 4
4 1
5 6
Enter the value of time QUANTUM:
2


Process :Turnaround Time:Waiting Time

Process3        1       0
Process2        9       5
Process1        14      9
Process4        11      5
Average waiting time 4.75
Average turn around time 8.75

Process returned 0 (0x0)   execution time : 28.505 s
Press any key to continue.
```

Conclusion:- A comparative study of Dynamic Quantum with Re-Adjusted Round Robin Scheduling algorithm, Optimized RR algorithm and proposed one is made. It is concluded that the proposed algorithm is superior to the other two algorithms as it has less waiting time, less turnaround time and usually less context switching thereby reducing the overhead and saving of memory space. Future work can be based on this algorithm modified and implemented for hard real time system where deadlines of the processes are to be taken into consideration.