Write program to implement Round Robin scheduling algorithm

CODE

```
#include <iostream>
using namespace std;
//Ready Queue

void queueUpdation(int queue[], int timer, int arrival[], int n, int
maxProccessIndex)
{
    int zeroIndex;
    for (int i = 0; i < n; i++)
    {
        if (queue[i] == 0)
        {
            zeroIndex = i;
            break;
        }
    }
    queue[zeroIndex] = maxProccessIndex + 1;
}
//queue maintained
void queueMaintainence(int queue[], int n)
{
    for (int i = 0; (i < n - 1) && (queue[i + 1] != 0); i++)
    {
        int temp = queue[i];
        queue[i] = queue[i + 1];
        queue[i + 1] = temp;
    }
}

void checkNewArrival(int timer, int arrival[], int n, int maxProccessIndex, int
queue[])
{
    if (timer <= arrival[n - 1])
    {
        bool newArrival = false;
```

```
        for (int j = (maxProccessIndex + 1); j < n; j++)
        {
            if (arrival[j] <= timer)
            {
                if (maxProccessIndex < j)
                {
                    maxProccessIndex = j;
                    newArrival = true;
                }
            }
        }
        //adds the incoming process to the ready queue

        if (newArrival)
        {
            queueUpdation(queue, timer, arrival, n, maxProccessIndex);
        }
    }
}

int main()
{
    int n, tq, timer = 0, maxProccessIndex = 0;
    float avgWait = 0, avgTT = 0;
    cout << "\nPlease enter the Time Quantum : ";
    cin >> tq;
    cout << "\nPlease enter the number of processes : ";
    cin >> n;
    int arrival[n], burst[n], wait[n], turn[n], queue[n], temp_burst[n];
    bool complete[n];

    cout << "\nPlease enter the Arrival Time (in ascending order) : ";
    for (int i = 0; i < n; i++)
    {
        cin >> arrival[i];
    }

    cout << "\nPlease enter the CPU Burst Time of the processes : ";
    for (int i = 0; i < n; i++)
    {
        cin >> burst[i];
```

```
        temp_burst[i] = burst[i];
    }

    for (int i = 0; i < n; i++)
    { //Initializing the queue and complete array
        complete[i] = false;
        queue[i] = 0;
    }

    while (timer < arrival[0]) //Incrementing Timer until the first process
arrives
    {
        timer++;
    }
    queue[0] = 1;

    while (true)
    {
        bool flag = true;
        for (int i = 0; i < n; i++)
        {
            if (temp_burst[i] != 0)
            {
                flag = false;
                break;
            }
        }
        if (flag)
        {
            break;
        }

        for (int i = 0; (i < n) && (queue[i] != 0); i++)
        {
            int ctr = 0;
            while ((ctr < tq) && (temp_burst[queue[0] - 1] > 0))
            {
                temp_burst[queue[0] - 1] -= 1;
                timer += 1;
                ctr++;
```

```
            //Checking and Updating the ready queue until all the processes
arrive
            checkNewArrival(timer, arrival, n, maxProccessIndex, queue);
        }

        if ((temp_burst[queue[0] - 1] == 0) && (complete[queue[0] - 1] ==
false))
        {
            //turn array currently stores the completion time
            turn[queue[0] - 1] = timer;
            complete[queue[0] - 1] = true;
        }

        bool idle = true;
        if (queue[n - 1] == 0)
        {
            for (int i = 0; i < n && queue[i] != 0; i++)
            {
                if (complete[queue[i] - 1] == false)
                {
                    idle = false;
                }
            }
        }
        else
        {
            idle = false;
        }
        if (idle)
        {
            timer++;
            checkNewArrival(timer, arrival, n, maxProccessIndex, queue);
        }

        //Maintaining the entries of processes
        //after each premption in the ready Queue
        queueMaintainence(queue, n);
    }
}

for (int i = 0; i < n; i++)
```

```
    {
        turn[i] = turn[i] - arrival[i];
        wait[i] = turn[i] - burst[i];
    }

    cout << "\nProcesses\tArrival Time\tCPU Burst Time\tWaiting Time\tTurnaround
Time" << endl;
    for (int i = 0; i < n; i++)
    {
        cout << i + 1 << "\t\t" << arrival[i] << "\t\t"
             << burst[i] << "\t\t" << wait[i] << "\t\t" << turn[i] << endl;
    }
    for (int i = 0; i < n; i++)
    {
        avgWait += wait[i];
        avgTT += turn[i];
    }
    cout << "\nAverage Waiting Time : " << (avgWait / n)
         << "\nAverage Turn Around Time : " << (avgTT / n);

    return 0;
}
```

OUTPUT

```
➜ OSPracticals g++ Practical7.cpp -o Practical7
➜ OSPracticals ./Practical7

Please enter the Time Quantum : 3

Please enter the number of processes : 3

Please enter the Arrival Time (in ascending order) : 0
2
3

Please enter the CPU Burst Time of the processes : 10
20
30

Processes       Arrival Time    CPU Burst Time  Waiting Time    Turnaround Time
1               0               10              18              28
2               2               20              26              46
3               3               30              27              57

Average Waiting Time : 23.6667
Average Turn Around Time : 43.6667%
➜ OSPracticals
```