Name: Vishal Shashikant Salvi.

UID: 2019230069

Class: SE Comps

Batch: C

Experiment No 2

Aim: Process Scheduling Policy.

Theory:
What is Round-Robin Scheduling?

The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turns. It is the oldest, simplest scheduling algorithm, which is mostly used for multitasking.

In Round-robin scheduling, each ready task runs turn by turn only in a cyclic queue for a limited time slice. This algorithm also offers starvation free execution of processes.

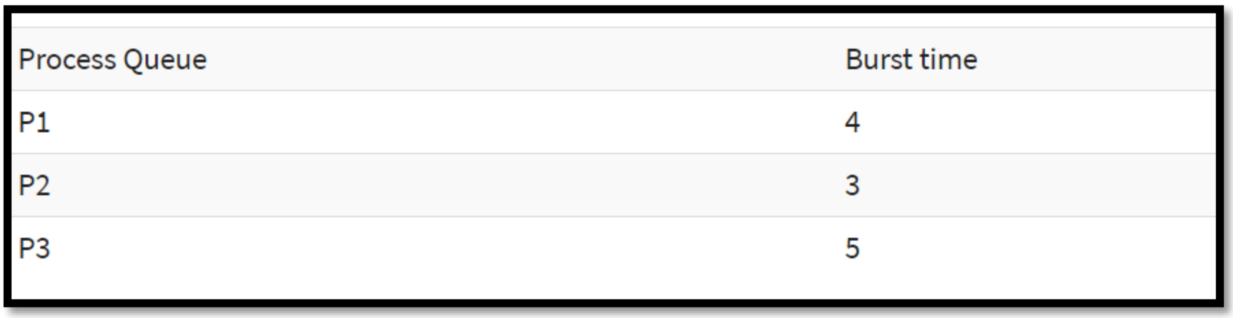
Characteristics of Round-Robin Scheduling

Here are the important characteristics of Round-Robin Scheduling:

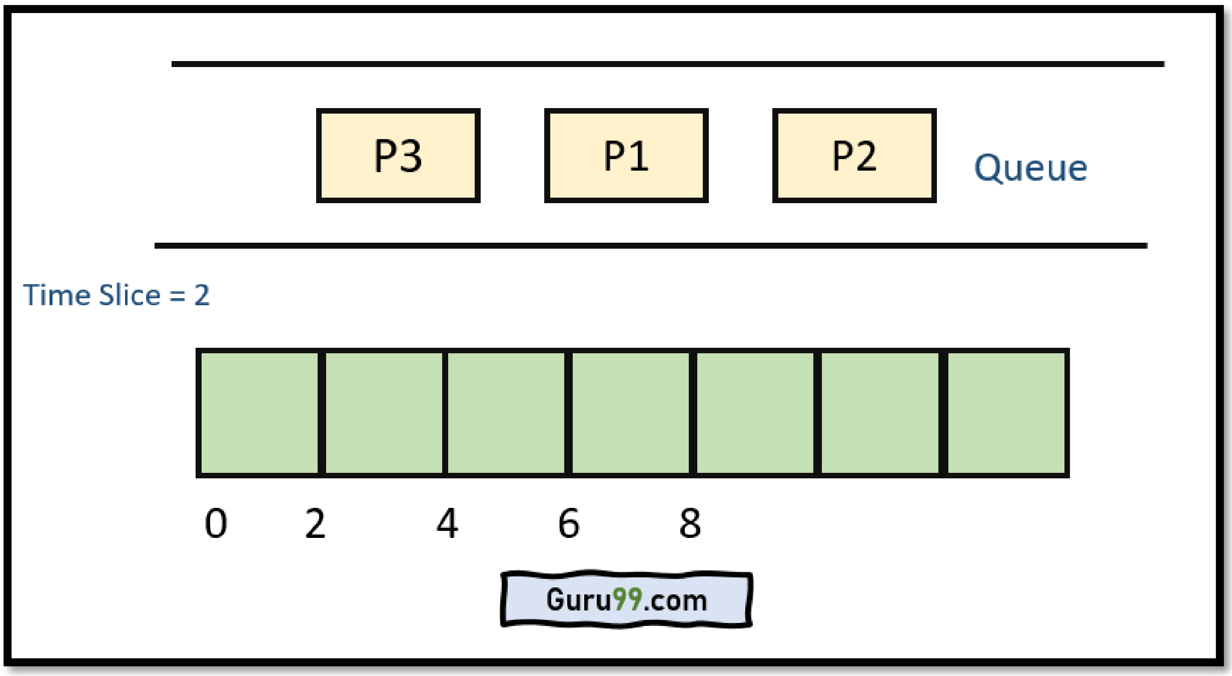
- Round robin is a pre-emptive algorithm
- The CPU is shifted to the next process after fixed interval time, which is called time quantum/time slice.
- The process that is preempted is added to the end of the queue.
- Round robin is a hybrid model which is clock-driven
- Time slice should be minimum, which is assigned for a specific task that needs to be processed. However, it may differ OS to OS.
- It is a real time algorithm which responds to the event within a specific time limit.
- Round robin is one of the oldest, fairest, and easiest algorithm.
- Widely used scheduling method in traditional OS.

Example of Round-robin Scheduling

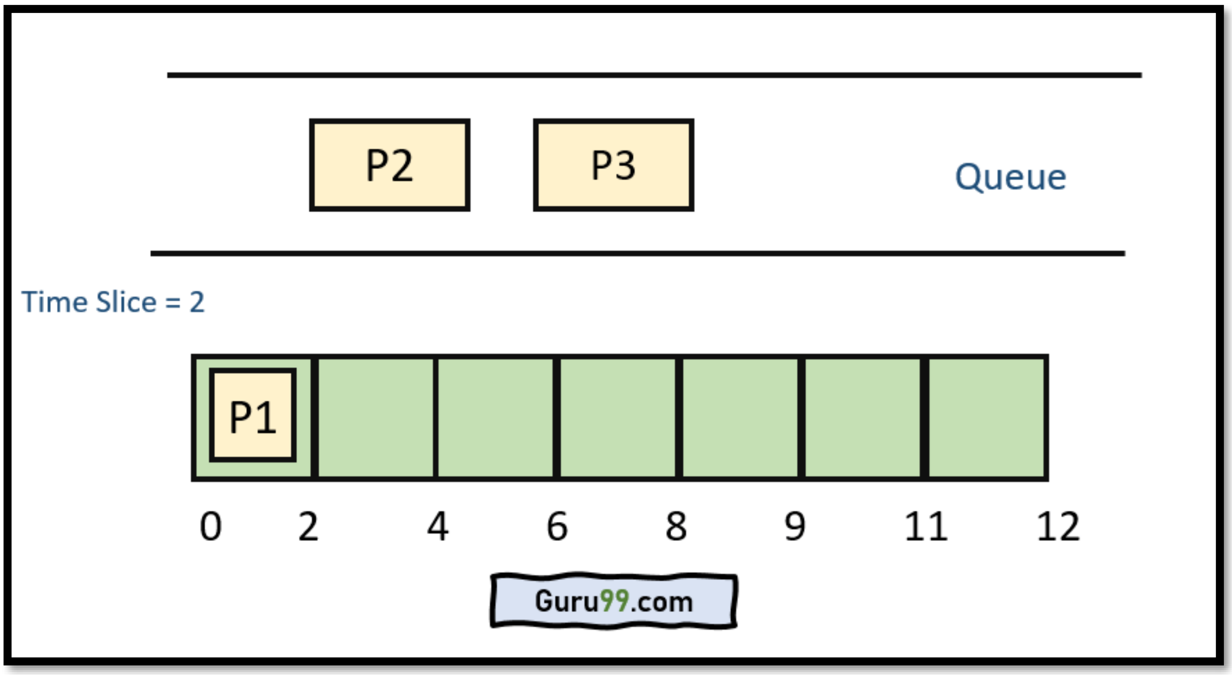
Consider this following three processes



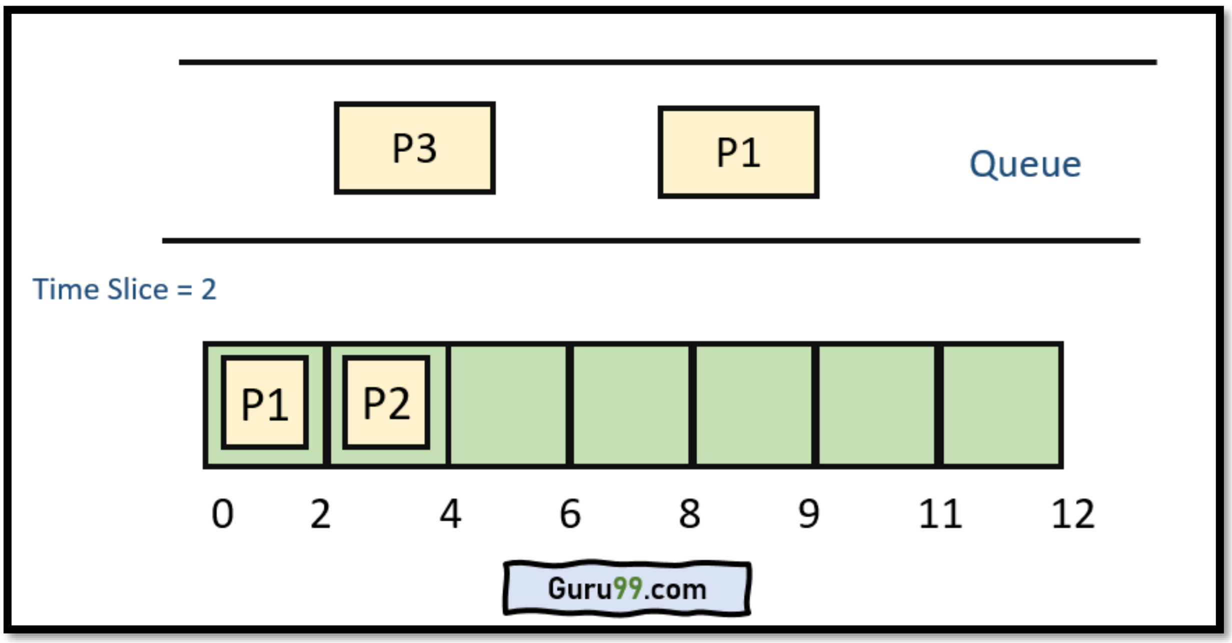
Process Queue	Burst time
P1	4
P2	3
P3	5



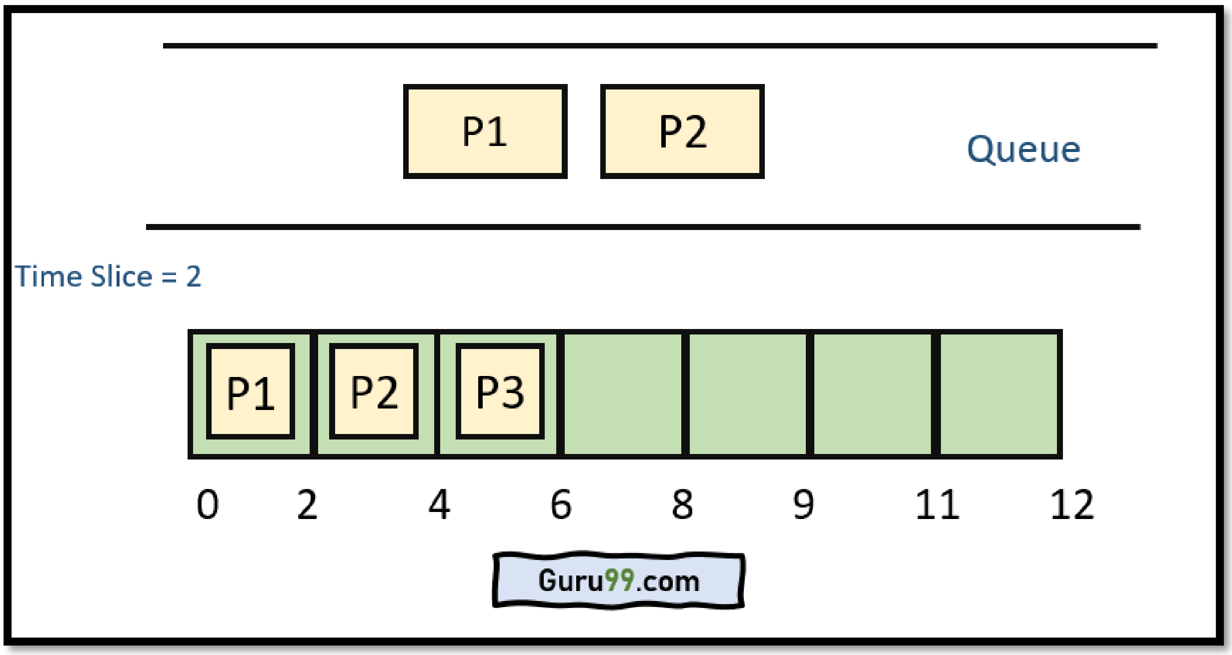
Step 1) The execution begins with process P1, which has burst time 4. Here, every process executes for 2 seconds. P2 and P3 are still in the waiting queue.



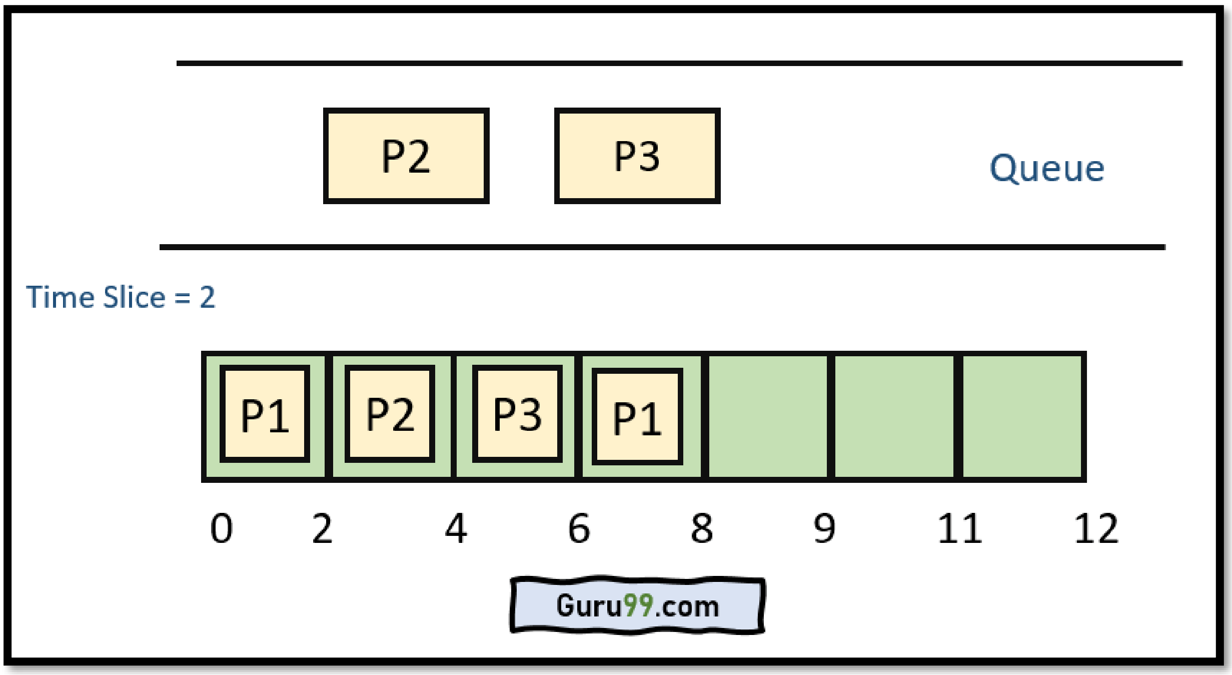
Step 2) At time =2, P1 is added to the end of the Queue and P2 starts executing



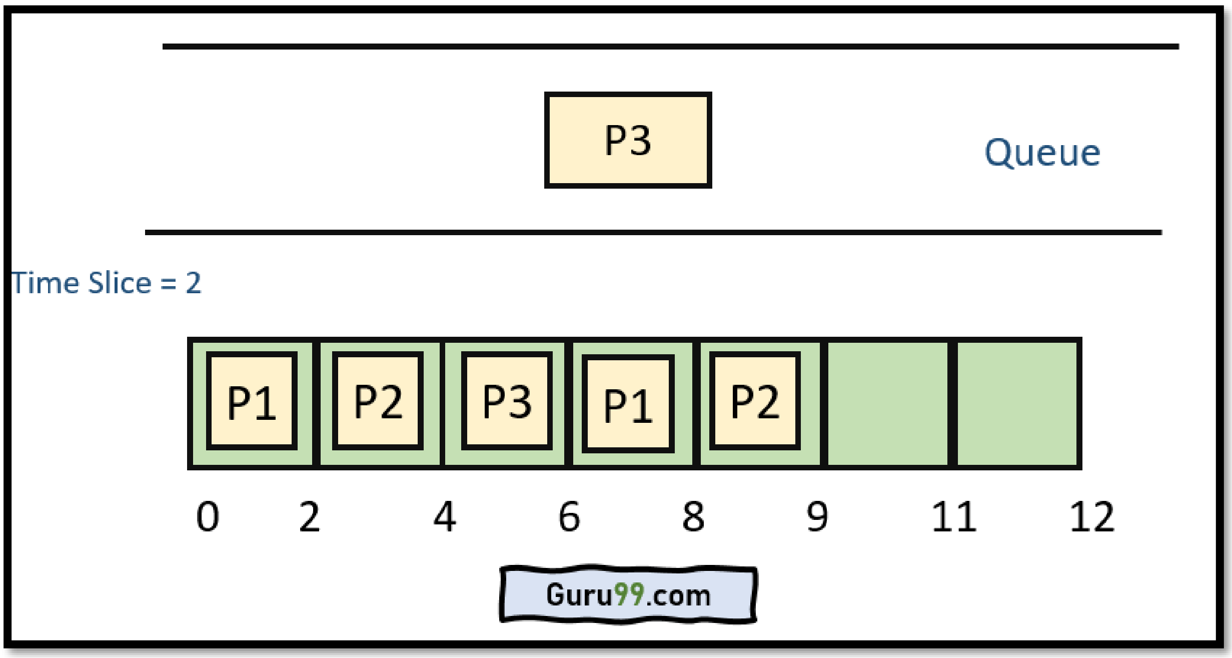
Step 3) At time=4 , P2 is preempted and add at the end of the queue. P3 starts executing.



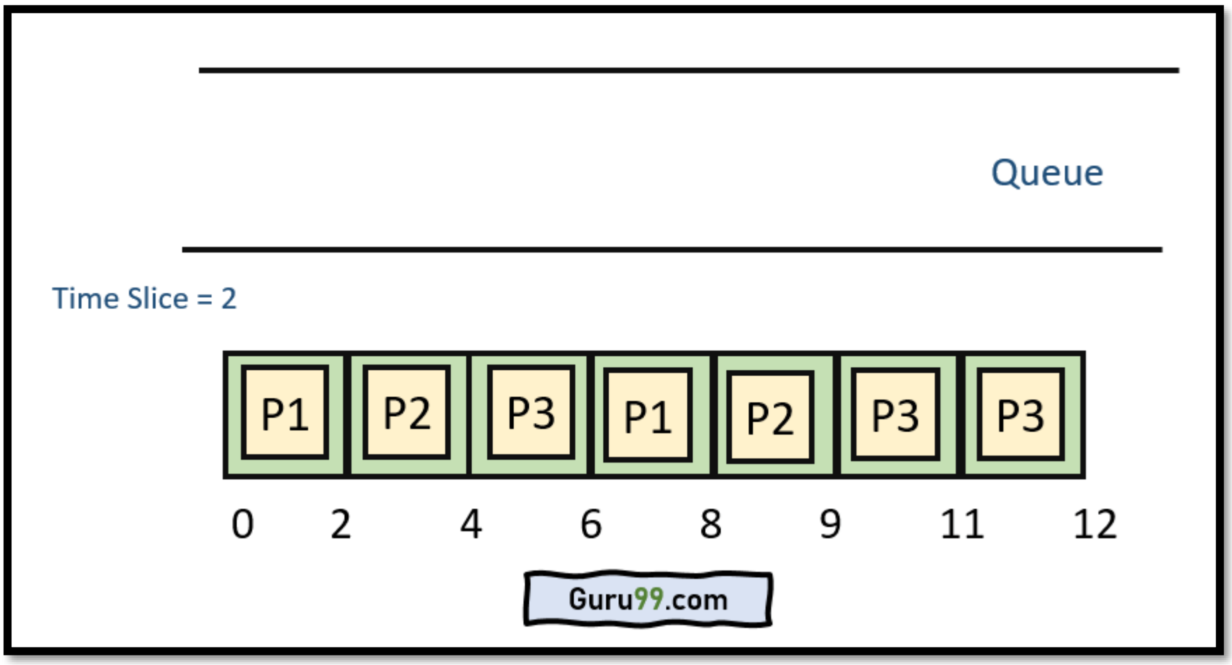
Step 4) At time=6 , P3 is preempted and add at the end of the queue. P1 starts executing.



Step 5) At time=8 , P1 has a burst time of 4. It has completed execution. P2 starts execution



Step 6) P2 has a burst time of 3. It has already executed for 2 interval. At time=9, P2 completes execution. Then, P3 starts execution till it completes.



Step 7) Let's calculate the average waiting time for above example.

Wait time
P1= 0+ 4= 4
P2= 2+4= 6
P3= 4+3= 7

Advantage of Round-robin Scheduling

Here, are pros/benefits of Round-robin scheduling method:

- It doesn't face the issues of starvation or convoy effect.
- All the jobs get a fair allocation of CPU.
- It deals with all process without any priority
- If you know the total number of processes on the run queue, then you can also assume the worst-case response time for the same process.
- This scheduling method does not depend upon burst time. That's why it is easily implementable on the system.
- Once a process is executed for a specific set of the period, the process is preempted, and another process executes for that given time period.
- Allows OS to use the Context switching method to save states of preempted processes.
- It gives the best performance in terms of average response time.

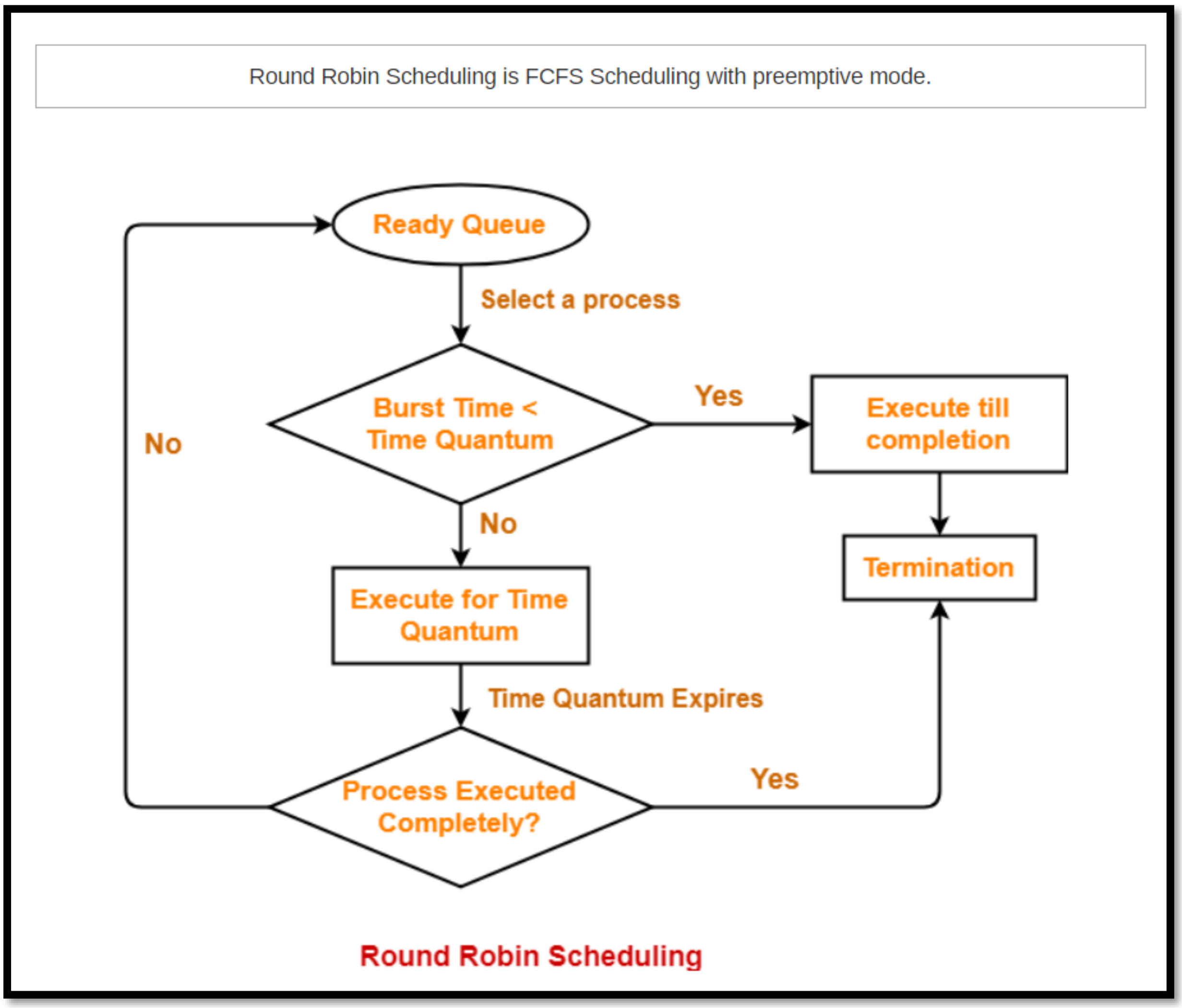
Disadvantages of Round-robin Scheduling

Here, are drawbacks/cons of using Round-robin scheduling:

- If slicing time of OS is low, the processor output will be reduced.
- This method spends more time on context switching
- Its performance heavily depends on time quantum.
- Priorities cannot be set for the processes.
- Round-robin scheduling doesn't give special priority to more important tasks.
- Decreases comprehension
- Lower time quantum results in higher the context switching overhead in the system.
- Finding a correct time quantum is a quite difficult task in this system.

Summary:

- The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turns.
- Round robin is one of the oldest, fairest, and easiest algorithms and widely used scheduling methods in traditional OS.
- Round robin is a pre-emptive algorithm
- The biggest advantage of the round-robin scheduling method is that If you know the total number of processes on the run queue, then you can also assume the worst-case response time for the same process.
- This method spends more time on context switching
- Worst-case latency is a term used for the maximum time taken for the execution of all the tasks.



Example:

1)Consider the set of 6 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	5	5
P2	4	6
P3	3	7

P4	1	9
P5	2	2
P6	6	3

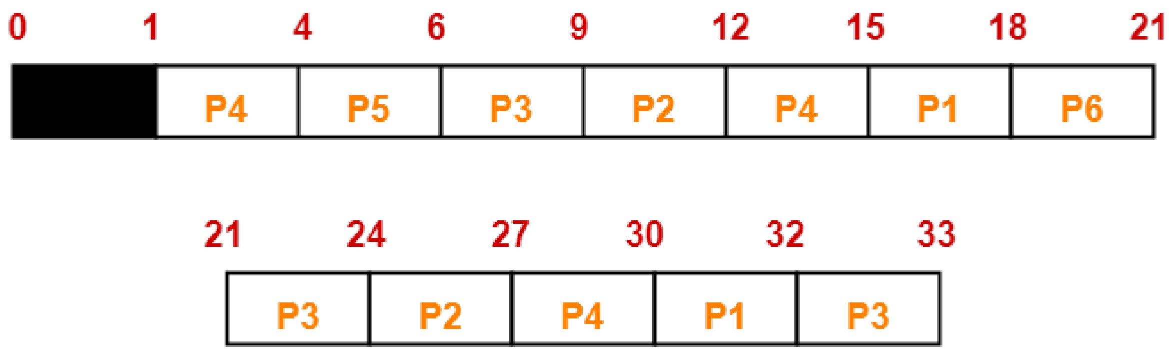
If the CPU scheduling policy is Round Robin with time quantum = 3, calculate the average waiting time and average turn around time.

<u>Solution-</u>

<u>Gantt chart-</u>

<u>Ready Queue-</u>

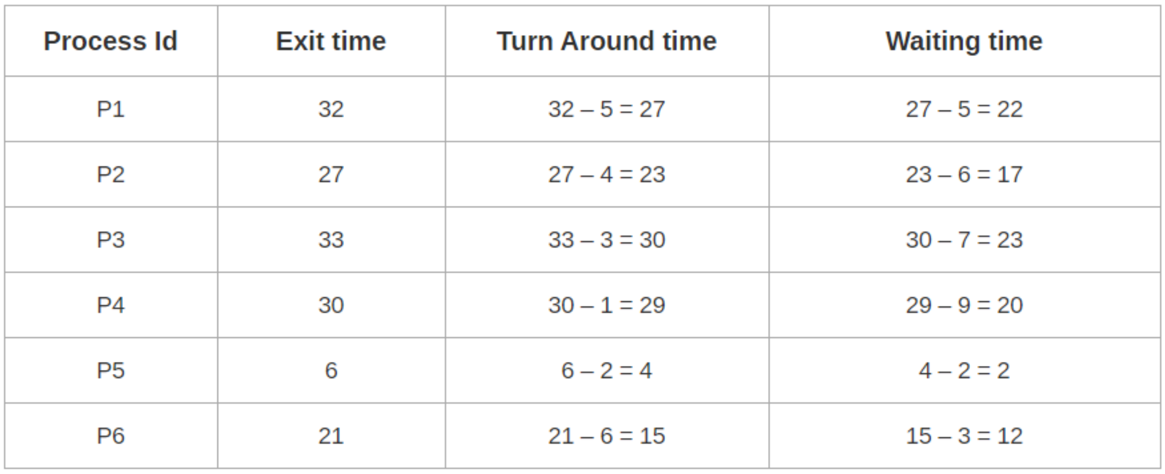
P3, P1, P4, P2, P3, P6, P1, P4, P2, P3, P5, P4



Gantt Chart

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time



Process Id	Exit time	Turn Around time	Waiting time
P1	32	32 – 5 = 27	27 – 5 = 22
P2	27	27 – 4 = 23	23 – 6 = 17
P3	33	33 – 3 = 30	30 – 7 = 23
P4	30	30 – 1 = 29	29 – 9 = 20
P5	6	6 – 2 = 4	4 – 2 = 2
P6	21	21 – 6 = 15	15 – 3 = 12

Now,

- Average Turn Around time = (27 + 23 + 30 + 29 + 4 + 15) / 6 = 128 / 6 = 21.33 unit
- Average waiting time = (22 + 17 + 23 + 20 + 2 + 12) / 6 = 96 / 6 = 16 unit

2)Consider the set of 6 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	4
P2	1	5
P3	2	2
P4	3	1
P5	4	6
P6	6	3

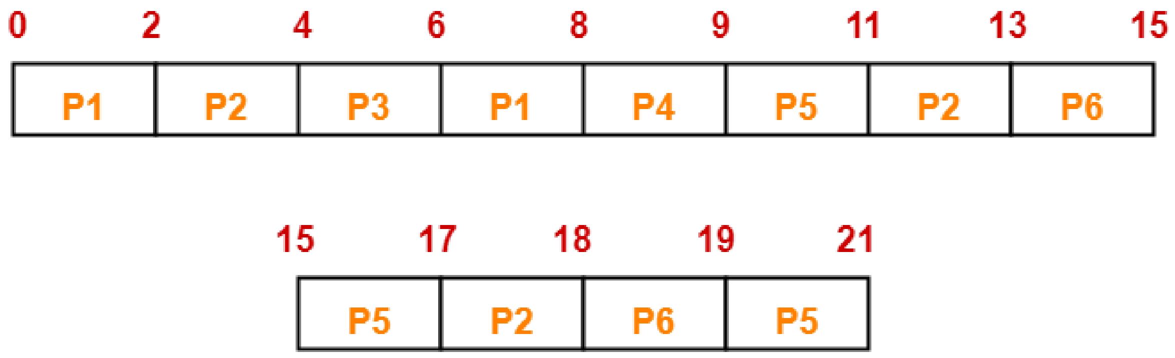
If the CPU scheduling policy is Round Robin with time quantum = 2, calculate the average waiting time and average turn around time.

Solution-

Gantt chart-

Ready Queue-

P5, P6, P2, P5, P6, P2, P5, P4, P1, P3, P2, P1



Gantt Chart

Now, we know-

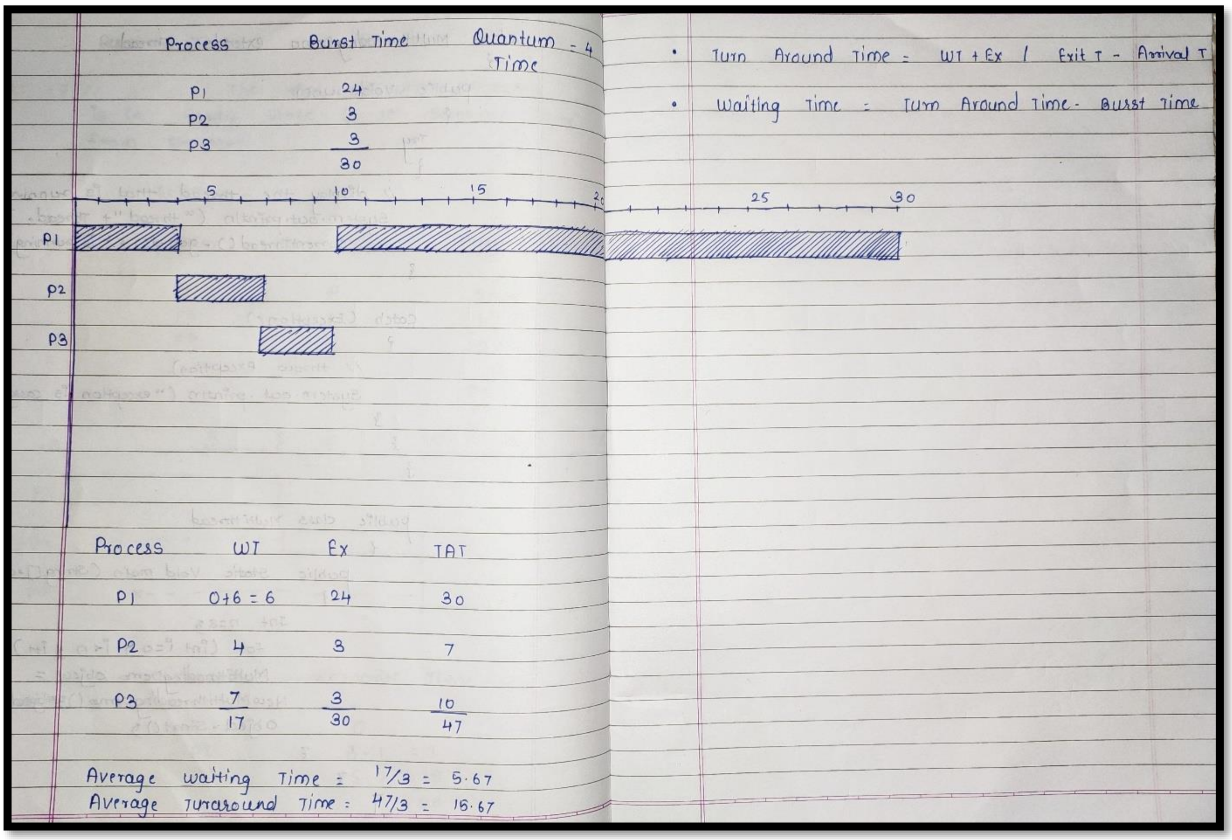
- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	8	8 – 0 = 8	8 – 4 = 4
P2	18	18 – 1 = 17	17 – 5 = 12
P3	6	6 – 2 = 4	4 – 2 = 2
P4	9	9 – 3 = 6	6 – 1 = 5
P5	21	21 – 4 = 17	17 – 6 = 11
P6	19	19 – 6 = 13	13 – 3 = 10

Now,

- Average Turn Around time = (8 + 17 + 4 + 6 + 17 + 13) / 6 = 65 / 6 = 10.84 unit
- Average waiting time = (4 + 12 + 2 + 5 + 11 + 10) / 6 = 44 / 6 = 7.33 unit

- **Snapshot:**



Process	Burst Time	Quantum Time = 4
P1	24
P2	3
P3	3
	30

- Turn Around Time = WT + Ex / Exit T - Arrival T
- Waiting Time = Turn Around Time - Burst Time

Process	WT	Ex	TAT
P1	0+6 = 6	24	30
P2	4	3	7
P3	7	3	10
	17	30	47

Average waiting Time = 17/3 = 5.67
Average Turaround Time = 47/3 = 15.67

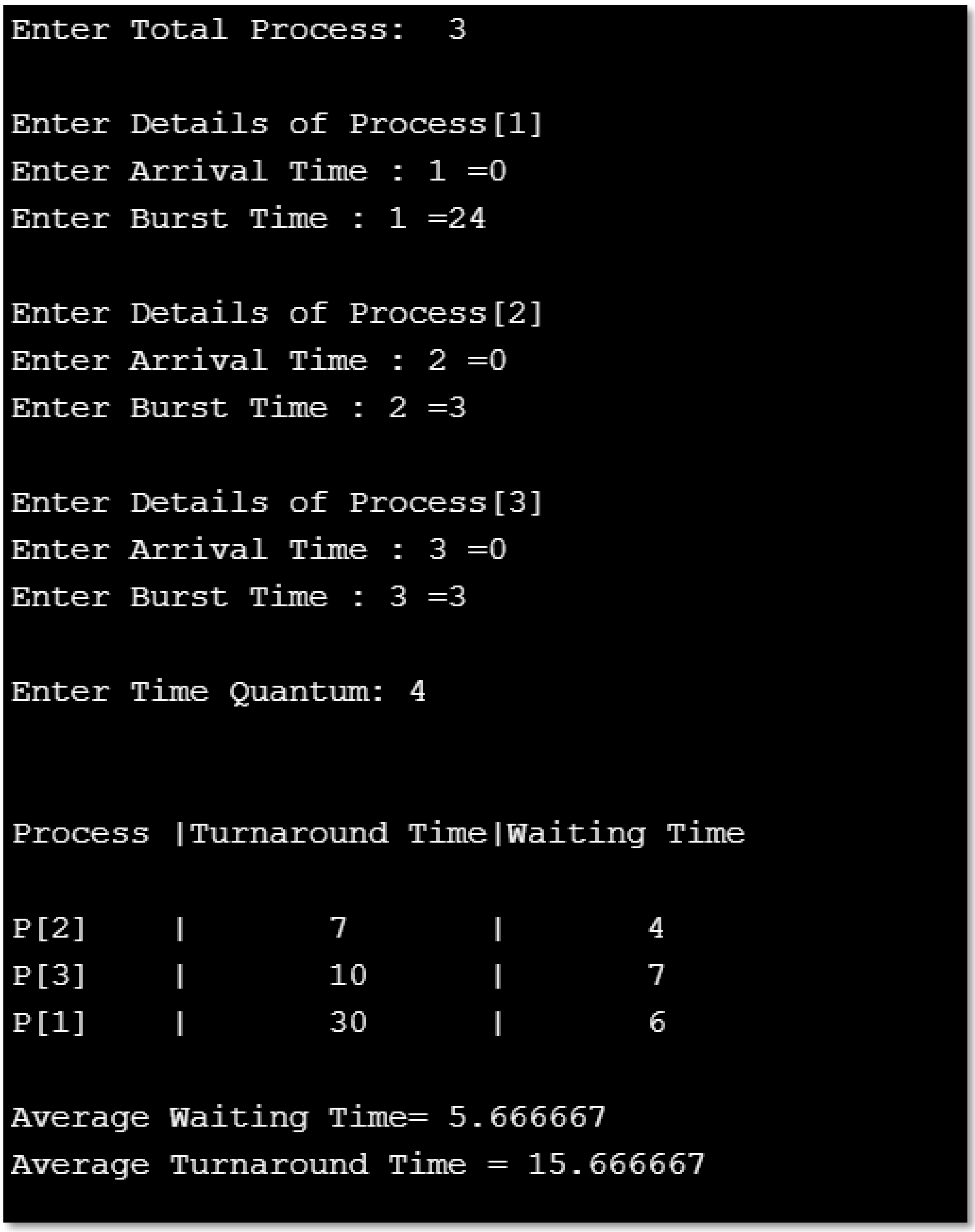
Code:

```
#include<stdio.h>

#include<conio.h>

void main()

{

  int count,j,n,time,remain,flag=0,time_quantum;

  int wt=0,tt=0,at[100],bt[100],rt[100];



  printf("\n********** Round Robin Scheduling**********");

  printf("\n\nEnter Total Process:  ");

  scanf("%d",&n);

  remain=n;
```

```
for(count=0;count<n;count++)
{
  printf("\nEnter Details of Process[%d]\n", count+1);
  printf("Enter Arrival Time : %d =",count+1);
  scanf("%d",&at[count]);
  printf("Enter Burst Time : %d =",count+1);
  scanf("%d",&bt[count]);
  rt[count]=bt[count];
}
printf("\nEnter Time Quantum: ");
scanf("%d",&time_quantum);
printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
for(time=0,count=0;remain!=0;)
{
  if(rt[count]<=time_quantum && rt[count]>0)
  {
    time+=rt[count];
    rt[count]=0;
    flag=1;
  }
  else if(rt[count]>0)
  {
    rt[count]-=time_quantum;
    time+=time_quantum;
  }
```

```
    if(rt[count]==0 && flag==1)
    {
      remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);
      wt+=time-at[count]-bt[count];
      tt+=time-at[count];
      flag=0;
    }
    if(count==n-1)
      count=0;
    else if(at[count+1]<=time)
      count++;
    else
      count=0;
  }
  printf("\nAverage Waiting Time= %f\n",wt*1.0/n);
  printf("Average Turnaround Time = %f",tt*1.0/n);


}
```

Output:



```
Enter Total Process:   3

Enter Details of Process[1]
Enter Arrival Time : 1 =0
Enter Burst Time : 1 =24

Enter Details of Process[2]
Enter Arrival Time : 2 =0
Enter Burst Time : 2 =3

Enter Details of Process[3]
Enter Arrival Time : 3 =0
Enter Burst Time : 3 =3

Enter Time Quantum: 4


Process |Turnaround Time|Waiting Time

P[2]    |       7       |       4
P[3]    |       10      |       7
P[1]    |       30      |       6

Average Waiting Time= 5.666667
Average Turnaround Time = 15.666667
```

Conclusion:

Scheduling is the process of allocating processes to the CPU in order to optimize some objective function. There are many algorithms used to schedule processes. The Round Robin (RR) CPU scheduling algorithm is one of these algorithms which is effective in time sharing and real time operating systems. It gives reasonable response time. But it suffers from several disadvantages such as high turnaround time, high waiting time and many context switches. There are large numbers of algorithms proposed to enhance the standard Round Robin algorithm.