✅ **Program Name: Round Robin CPU Scheduling Algorithm**

✅ **Type:** Preemptive Scheduling
✅ **Key Feature:** Each process gets equal CPU time in cyclic order
✅ **Uses:** Time-sharing operating systems

🔍 **Concept of Round Robin (RR)**

- RR is a **preemptive scheduling algorithm**.
- Every process gets a fixed **Time Quantum (TQ)**.
- If a process doesn't finish in its time slice, it goes back to the **ready queue**.
- CPU moves to next process.
- This continues until all processes complete.

🧠 **Important Terms**

Term	Meaning
AT	Arrival Time (When process enters CPU queue)
BT	Burst Time (Total execution time needed)
RT	Remaining Time (BT that is still left)
WT	Waiting Time (Time spent in queue, not executing)
TAT	Turn Around Time (Completion Time – Arrival Time)
TQ	Time Quantum (Fixed time slot per process)

Formula:

WT = Completion Time – Arrival Time – Burst Time

TAT = Burst Time + Waiting Time

📌 **Step-by-step Code Explanation**

1️⃣ **Declaring Variables**

```
int n, i, tq, time = 0, completed = 0;
int at[20], bt[20], rt[20], wt[20], tat[20];
```

- n = number of processes
- at[] = arrival time

- bt[] = burst time
- rt[] = remaining time (copy of burst time)
- wt[] = waiting time
- tat[] = turn around time
- time = system clock
- completed = number of finished processes

2️⃣ Input Section

```
for(i = 0; i < n; i++) {
   scanf("%d", &at[i]);
   scanf("%d", &bt[i]);
   rt[i] = bt[i];  // remaining time initialized
}
```

✓ Stores arrival & burst time for each process
✓ Copies burst time into remaining time

3️⃣ Round Robin Execution Loop

```
while (completed != n) {
   for(i = 0; i < n; i++) {
      if(at[i] <= time && rt[i] > 0) {
```

✓ Loop runs until all processes finish
✓ Each process executes if:

- It has arrived (at[i] <= time)
- It still has remaining time (rt[i] > 0)

4️⃣ If Remaining Time > Time Quantum

```
if(rt[i] > tq) {
   time += tq;
   rt[i] -= tq;
}
```

➞ Process executes only for time quantum
➞ Remaining time reduces

5️⃣ If Remaining Time ≤ Time Quantum

```
else {
   time += rt[i];
   wt[i] = time - at[i] - bt[i];
   rt[i] = 0;
   completed++;
}
```

✓ Process finishes completely
✓ Waiting time is calculated
✓ completed++ increases finished count

6️⃣ Gantt Chart Printing

```
printf(" -- P%d -- %d", i+1, time);
```

✓ Prints execution order

Example output:

0 -- P1 -- 4 -- P2 -- 8 -- P1 -- 10 ...

7️⃣ Calculating & Displaying Table

```
tat[i] = bt[i] + wt[i];
printf("P%d\t%d\t%d\t%d\t%d\n", ...);
```

✓ Turnaround Time = Burst + Waiting

8️⃣ Average WT & TAT

```
printf("\nAverage Waiting Time: %.2f", avgWT / n);
printf("\nAverage Turnaround Time: %.2f\n", avgTAT / n);
```

📘 Theory Answer for Practical Viva

✅ **Definition:**
Round Robin (RR) Scheduling is a **preemptive CPU scheduling algorithm** where each process gets a fixed time slice called **Time Quantum** in a cyclic order.

✅ **Why Used:**
Used in **time-sharing operating systems** to give fair CPU time to all processes.

✅ **Advantages:**

- No starvation
- All processes treated equally
- Good for interactive systems

✅ **Disadvantages:**

- If time quantum is too large → behaves like FCFS
- If time quantum is too small → too many context switches