Program 3: Round Robin Scheduling

```
// C++ program for implementation of RR scheduling
#include<iostream>
using namespace std;

// Function to find the waiting time for all
// processes
void findWaitingTime(int processes[], int n,
int bt[], int wt[], int quantum)

{
// Make a copy of burst times bt[] to store remaining
// burst times.
int rem_bt[n];
for (int i = 0 ; i < n ; i++)
rem_bt[i] = bt[i];

int t = 0; // Current time

// Keep traversing processes in round robin manner
// until all of them are not done.
while (1)
{
bool done = true;

// Traverse all processes one by one repeatedly
for (int i = 0 ; i < n; i++)
```

```
{
// If burst time of a process is greater than 0
// then only need to process further
if (rem_bt[i] > 0)
{


done = false; // There is a pending process


if (rem_bt[i] > quantum)
{
// Increase the value of t i.e. shows
// how much time a process has been processed
t += quantum;

// Decrease the burst_time of current process
// by quantum
rem_bt[i] -= quantum;
}

// If burst time is smaller than or equal to
// quantum. Last cycle for this process
else
{
// Increase the value of t i.e. shows
// how much time a process has been processed
t = t + rem_bt[i];
```

```
// Waiting time is current time minus time

// used by this process

wt[i] = t - bt[i];


// As the process gets fully executed

// make its remaining burst time = 0

rem_bt[i] = 0;

}

}

}


// If all processes are done

if (done == true)

break;

}

}


// Function to calculate turn around time

void findTurnAroundTime(int processes[], int n,


int bt[], int wt[], int tat[])


{

// calculating turnaround time by adding

// bt[i] + wt[i]

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

```
tat[i] = bt[i] + wt[i];


}

// Function to calculate average time
void findavgTime(int processes[], int n, int bt[], int quantum)
{
int wt[n], tat[n], total_wt = 0, total_tat = 0;

// Function to find waiting time of all processes
findWaitingTime(processes, n, bt, wt, quantum);

// Function to find turn around time for all processes
findTurnAroundTime(processes, n, bt, wt, tat);

// Display processes along with all details
cout << "PN\t "<< " \tBT "
<< " WT " << " \tTAT\n";


// Calculate total waiting time and total turn
// around time
for (int i=0; i<n; i++)
{
total_wt = total_wt + wt[i];
total_tat = total_tat + tat[i];
cout << " " << i+1 << "\t\t" << bt[i] <<"\t "
```

```
<< wt[i] <<"\t\t " << tat[i] <<endl;

}

cout << "Average waiting time = "
<< (float)total_wt / (float)n;
cout << "\nAverage turn around time = "
<< (float)total_tat / (float)n;

}

// Driver code
int main()
{
// process id's
int processes[] = { 1, 2, 3};
int n = sizeof processes / sizeof processes[0];

// Burst time of all processes
int burst_time[] = {10, 5, 8};

// Time quantum
int quantum = 2;
findavgTime(processes, n, burst_time, quantum);
return 0;
}
```

OUTPUT:

PN BT WT TAT

1 10 13 23

2 5 10 15

3 8 13 21

Average waiting time = 12

Average turn around time = 19.6667