```
#include<iostream>
using namespace std;

 void findTurnAroundTime(int processes[], int n,int bt[], int wt[], int tat[]);
 void findWaitingTime(int processes[], int n, int bt[], int wt[], int quantum);

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

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

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

   cout << "Processes "<< " Burst time "<< " Waiting time " << " Turn around time\n";   // Display processes along with all details

   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;      // Calculate total waiting time and total turn around time
   }
   cout << "Average waiting time = " << (float)total_wt / (float)n;
   cout << "\nAverage turn around time = " << (float)total_tat / (float)n;
}
```

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

   int rem_bt[n];                    // Make a copy of burst times bt[] to store remaining burst times.
   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;

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

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


void findTurnAroundTime(int processes[], int n,int bt[], int wt[], int tat[])   // Function to calculate turn around time
{

   for (int i = 0; i < n ; i++)
           {
      tat[i] = bt[i] + wt[i];                  // calculating turnaround time by adding Burst Time and Waiting Time
   }
}



int main()
{
   int processes[] = { 1, 2, 3, 4};                      // process id's
   int n = sizeof processes / sizeof processes[0];          //Number of Processes
   int burst_time[] = {20, 36, 19, 42};                  // Burst time of all processes
   int quantum = 10;                                  // Time quantum
   findavgTime(processes, n, burst_time, quantum);
```

```
    return 0;
}
```