NAME – PINKY PAMECHA
ROLL NO-C114
SAPID-60004220056

EXPERIMENT NO 4
CODE AND OUTPUT

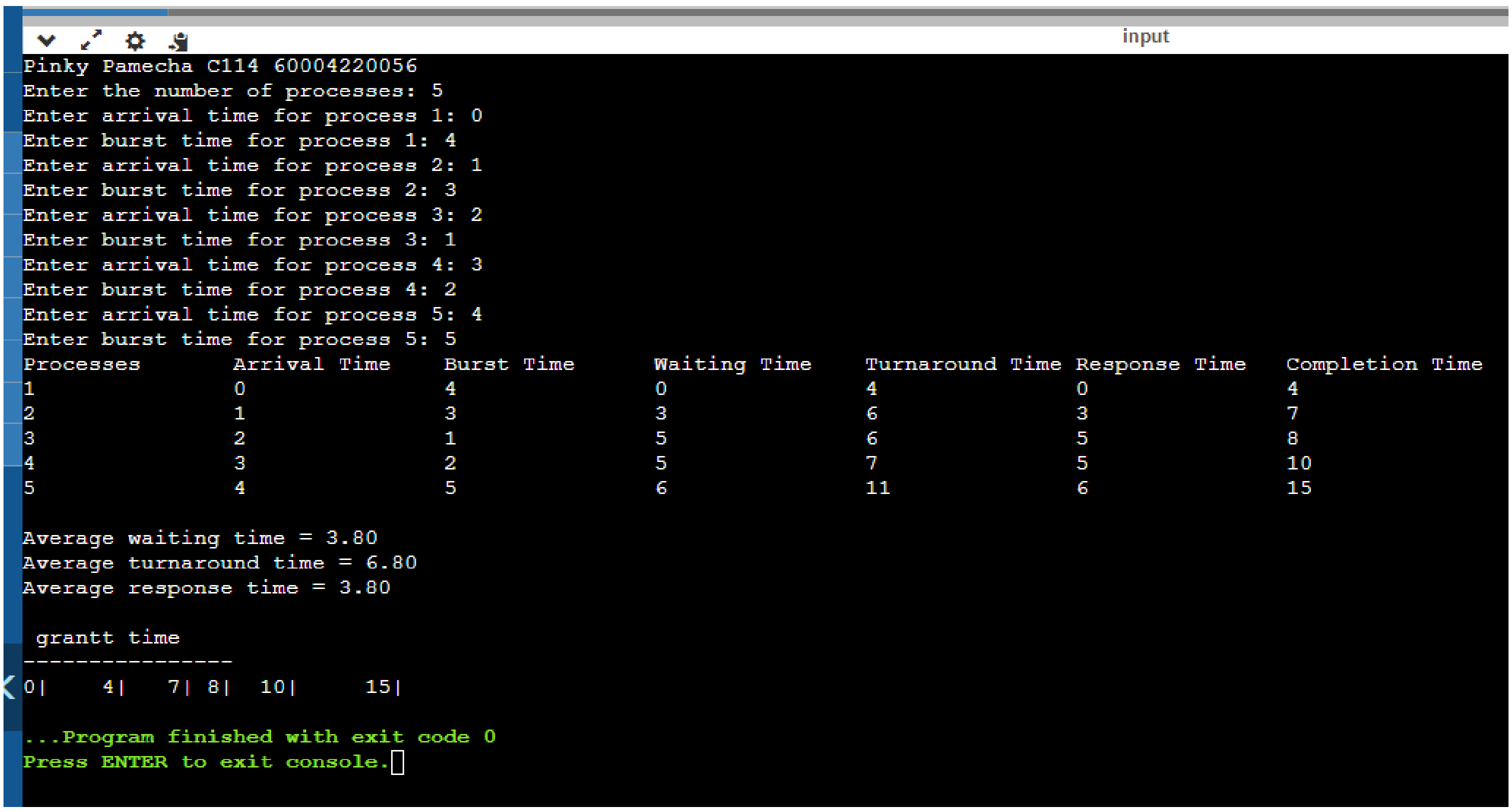
FCFS

CODE:

```
import java.util.Scanner;

class Main{
   // Function to find the waiting time for all
   static void findWaitingTime(int processes[], int n, int bt[], int wt[], int at[]) {
      int service_time[] = new int[n];
      service_time[0] = at[0];
      wt[0] = 0;

      // calculating waiting time
      for (int i = 1; i < n; i++) {
         service_time[i] = service_time[i - 1] + bt[i - 1];
         wt[i] = Math.max(0, service_time[i] - at[i]);
      }
   }

   // Function to calculate turn around time
   static 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 response time
   static void findResponseTime(int wt[], int n, int res[]) {
      // Response time is same as waiting time for FCFS
      for (int i = 0; i < n; i++) {
         res[i] = wt[i];
      }
   }

   // Function to calculate average time
   static void findavgTime(int processes[], int n, int bt[], int at[]) {
      int wt[] = new int[n], tat[] = new int[n], res[] = new int[n], ct[] = new int[n];
      int total_wt = 0, total_tat = 0, total_res = 0;
      // Function to find waiting time of all processes
      findWaitingTime(processes, n, bt, wt, at);
      // Function to find turn around time for all processes
      findTurnAroundTime(processes, n, bt, wt, tat);
      // Function to find response time for all processes
      findResponseTime(wt, n, res);
      // Calculate completion time for all processes
      for (int i = 0; i < n; i++) {
         ct[i] = at[i] + tat[i];
      }
      // Display processes along with all details
```

```
        System.out.printf("Processes\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\tResponse
Time\tCompletion Time\n");
        // Calculate total waiting time, total turn around time, and total response time
        for (int i = 0; i < n; i++) {
            total_wt += wt[i];
            total_tat += tat[i];
            total_res += res[i];
            System.out.printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", processes[i], at[i], bt[i], wt[i], tat[i], res[i], ct[i]);
        }
        // Calculate average waiting time, average turn around time, and average response time
        float avg_wt = (float) total_wt / n;
        float avg_tat = (float) total_tat / n;
        float avg_res = (float) total_res / n;
        System.out.printf("\nAverage waiting time = %.2f\n", avg_wt);
        System.out.printf("Average turnaround time = %.2f\n", avg_tat);
        System.out.printf("Average response time = %.2f\n", avg_res);

        System.out.printf("\n grantt time");
        System.out.printf("\n----------------\n");
        System.out.printf("0|");
        for(int i =0;i<n;i++){
            for(int j=0;j<bt[i];j++)
            {
                System.out.printf(" ");
            }
            System.out.printf("%d|",ct[i]);
        }

    }

    // Driver code
    public static void main(String[] args) {
        // Process id's
        System.out.println("Pinky Pamecha C114 60004220056");
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the number of processes: ");
        int n = s.nextInt();
        int processes[] = new int[n];
        int burst_time[] = new int[n];
        int arrival_time[] = new int[n];
        // Input arrival times and burst times for each process
        for (int i = 0; i < n; i++) {
            System.out.print("Enter arrival time for process " + (i + 1) + ": ");
            arrival_time[i] = s.nextInt();
            System.out.print("Enter burst time for process " + (i + 1) + ": ");
            burst_time[i] = s.nextInt();
            processes[i] = i + 1; // Assigning process IDs
        }
        findavgTime(processes, n, burst_time, arrival_time);
        s.close();
    }
}
```

OUTPUT:



```
Pinky Pamecha C114 60004220056
Enter the number of processes: 5
Enter arrival time for process 1: 0
Enter burst time for process 1: 4
Enter arrival time for process 2: 1
Enter burst time for process 2: 3
Enter arrival time for process 3: 2
Enter burst time for process 3: 1
Enter arrival time for process 4: 3
Enter burst time for process 4: 2
Enter arrival time for process 5: 4
Enter burst time for process 5: 5
Processes       Arrival Time    Burst Time      Waiting Time    Turnaround Time Response Time   Completion Time
1               0               4               0               4               0               4
2               1               3               3               6               3               7
3               2               1               5               6               5               8
4               3               2               5               7               5               10
5               4               5               6               11              6               15

Average waiting time = 3.80
Average turnaround time = 6.80
Average response time = 3.80

 grantt time
----------------
0|    4|   7| 8|  10|     15|

...Program finished with exit code 0
Press ENTER to exit console.
```

SJF NON-PREEMTIVE

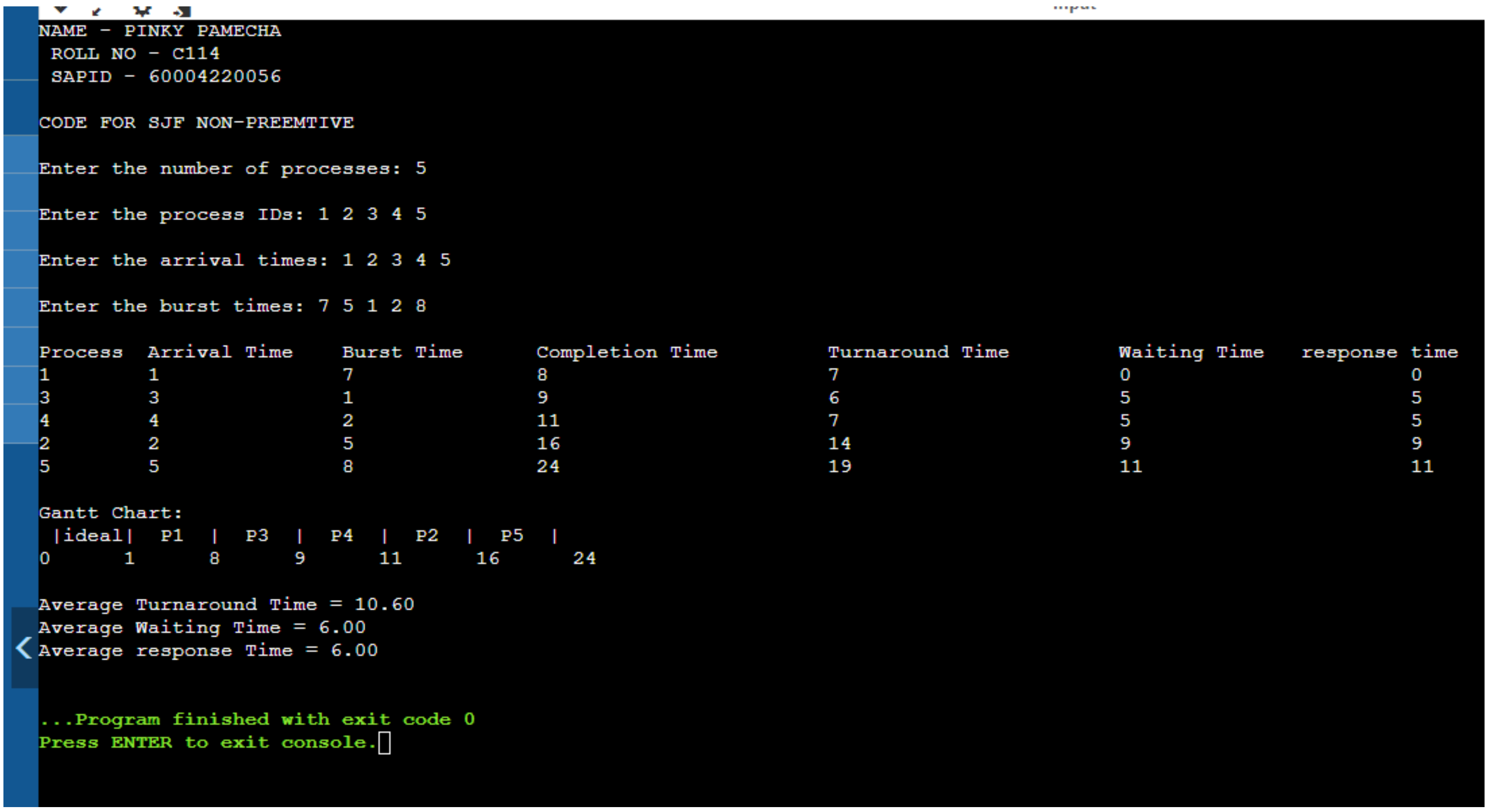
CODE:

```
import java.util.Scanner;

public class Main {

    public static void swap(int[] array, int i, int j) {
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }

    public static void sortat(int[] p, int[] at, int[] bt, int n) {
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                if (at[i] > at[j]) {
                    swap(p, i, j);
                    swap(at, i, j);
                    swap(bt, i, j);
                } else if (at[i] == at[j]) {
                    if (bt[i] > bt[j]) {
                        swap(p, i, j);
                        swap(at, i, j);
                        swap(bt, i, j);
                    }
                }
```

```java
            }
        }
    }

    public static void tatwtct(int[] ct, int[] at, int[] bt, int[] tat, int[] wt, int[]rt,int n) {
        for (int i = 0; i < n; i++) {
            tat[i] = ct[i] - at[i];
            wt[i] = tat[i] - bt[i];
            rt[i] = ct[i] - at[i] - bt[i];
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("NAME - PINKY PAMECHA \n ROLL NO - C114 \n SAPID - 60004220056");
        System.out.println("\nCODE FOR SJF NON-PREEMTIVE");

        int n;
        System.out.print("\nEnter the number of processes: ");
        n = scanner.nextInt();
        int[] p = new int[n];
        int[] at = new int[n];
        int[] bt = new int[n];
        int[] ct = new int[n];
        int[] wt = new int[n];
        int[] tat = new int[n];
        int[] rt= new int[n];
        System.out.print("\nEnter the process IDs: ");
        for (int i = 0; i < n; i++) {
            p[i] = scanner.nextInt();
        }
        System.out.print("\nEnter the arrival times: ");
        for (int i = 0; i < n; i++) {
            at[i] = scanner.nextInt();
        }
        System.out.print("\nEnter the burst times: ");
        for (int i = 0; i < n; i++) {
            bt[i] = scanner.nextInt();
        }
        sortat(p, at, bt, n);
        ct[0] = at[0] + bt[0];
        for (int i = 1; i < n; i++) {
            int min = 1000;
            int pos = i;
            for (int j = i; j < n; j++) {
                if (at[j] <= ct[i - 1] && bt[j] < min) {
                    min = bt[j];
                    pos = j;
                }
            }
            swap(p, i, pos);
            swap(at, i, pos);
            swap(bt, i, pos);
            ct[i] = (ct[i - 1] >= at[i]) ? ct[i - 1] + bt[i] : at[i] + bt[i];
        }
        tatwtct(ct, at, bt, tat, wt, rt,n);
        System.out.println("\nProcess\t Arrival Time\t Burst Time\t Completion Time\t Turnaround Time\t Waiting Time
\tresponse time");
```

```
        for (int i = 0; i < n; i++) {
            System.out.printf("%d\t %d\t\t %d\t\t %d\t\t\t %d\t\t\t %d\t\t\t %d\n", p[i], at[i], bt[i], ct[i], tat[i], wt[i] , rt[i]);
        }
        // Print Gantt Chart
        System.out.println("\nGantt Chart:");
        System.out.print(" ");
        int ct1 = 0;
        int i =0;
        while(i<n){
            if(at[i]>ct1){
                System.out.printf("|ideal");
                ct1++;
            }
            else{
            System.out.printf("|  P%d  ", p[i]);
                ct1 = ct1 + bt[i];
                i++;
            }
        }
        int ct2 =0;
        System.out.print("|\n0");
        int j =0;
        while(j<n) {
            if(at[j]>ct2){
                System.out.printf("      %d",at[j]);
                ct2++;
            }
            else{
            System.out.printf("      %d", ct[j]);
             ct2 = ct2 + bt[j];
             j++;
            }
        }
        System.out.println();
        float atat = 0, awt = 0 , art = 0;
        for (int k = 0; k < n; k++) {
            atat += tat[k];
            awt += wt[k];
            art +=rt[k];
        }
        atat /= n;
        awt /= n;
        art /=n;
        System.out.printf("\nAverage Turnaround Time = %.2f\n", atat);
        System.out.printf("Average Waiting Time = %.2f\n", awt);
        System.out.printf("Average response Time = %.2f\n", art);
        scanner.close();
    }
}
```

OUTPUT:



```
NAME - PINKY PAMECHA
 ROLL NO - C114
 SAPID - 60004220056

CODE FOR SJF NON-PREEMTIVE

Enter the number of processes: 5

Enter the process IDs: 1 2 3 4 5

Enter the arrival times: 1 2 3 4 5

Enter the burst times: 7 5 1 2 8

Process  Arrival Time    Burst Time      Completion Time         Turnaround Time         Waiting Time   response time
1        1               7               8                       7                       0                           0
3        3               1               9                       6                       5                           5
4        4               2               11                      7                       5                           5
2        2               5               16                      14                      9                           9
5        5               8               24                      19                      11                          11

Gantt Chart:
 |ideal|  P1   |  P3   |  P4   |  P2   |  P5   |
0      1      8      9      11      16      24

Average Turnaround Time = 10.60
Average Waiting Time = 6.00
Average response Time = 6.00


...Program finished with exit code 0
Press ENTER to exit console.
```

SJF PREEMTIVE

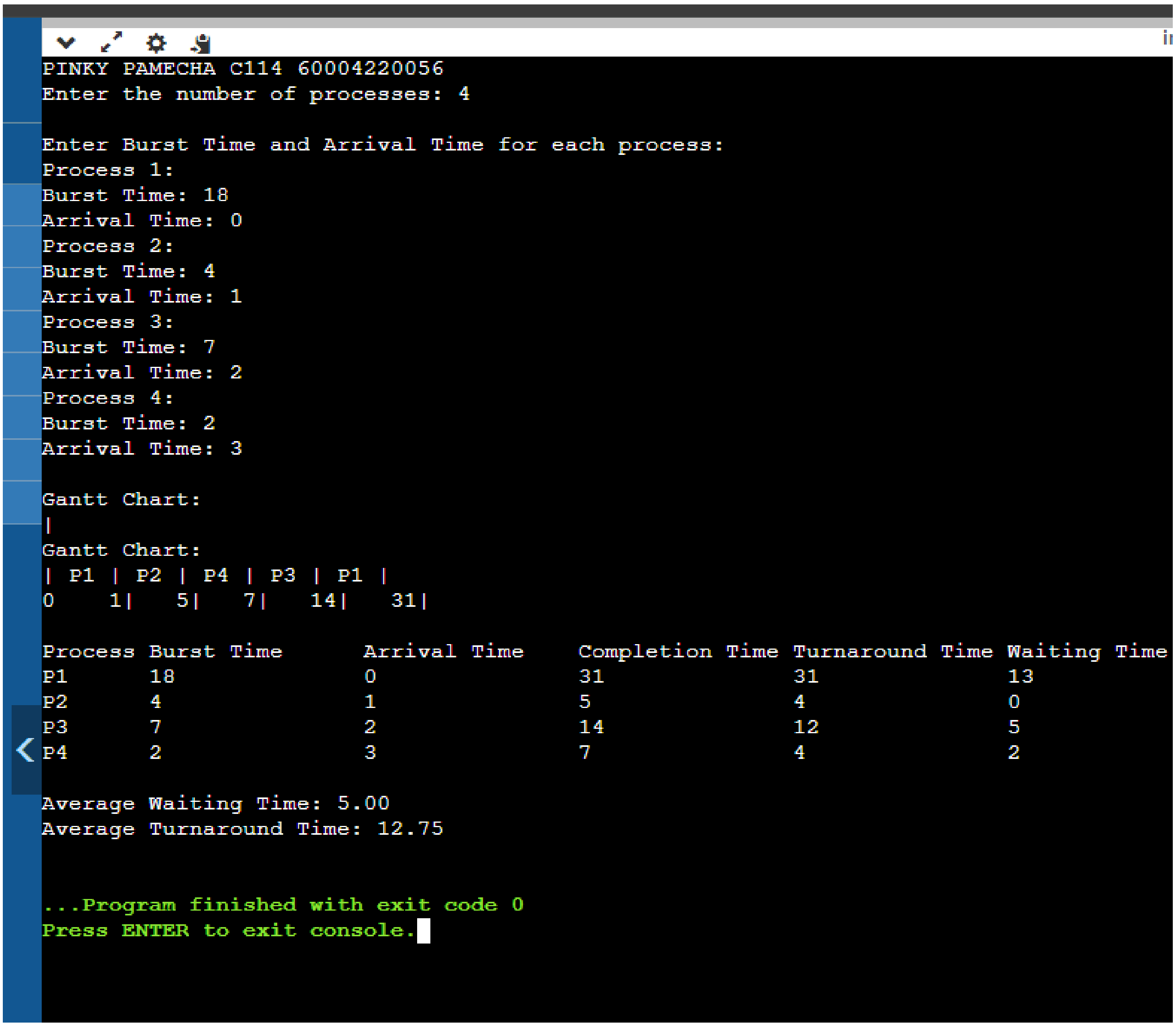
CODE:

```
// Class to represent a process
import java.util.Scanner;
class Process {
   int pid; // Process ID
   int bt; // Burst Time
   int at; // Arrival Time
   int rt; // Remaining Time
}

class Main {

   // Function to find the process with the minimum remaining time
   public static int findMinRtProcess(Process[] proc, int n, int currentTime) {
      int minIndex = -1, minRt = Integer.MAX_VALUE;
      for (int i = 0; i < n; i++) {
         if (proc[i].rt > 0 && proc[i].at <= currentTime && proc[i].rt < minRt) {
            minRt = proc[i].rt;
            minIndex = i;
         }
      }
      return minIndex;
   }

   // Function to perform SJF preemptive scheduling
   public static void SJF(Process[] proc, int n) {
      int currentTime = 0, completed = 0;
      int[] wt = new int[n], tat = new int[n], ct = new int[n];
```

```
        float avgWaitTime = 0, avgTurnaroundTime = 0;
        // Initialize remaining time for each process
        for (int i = 0; i < n; i++) {
            proc[i].rt = proc[i].bt;
        }
        // Array to track the process running at each time slot
        int[] gantt = new int[n * 100]; // Assuming each process runs for at most 100 units of time
        int idx = 0; // Index for the gantt chart array
        // If no process arrives at time 0, add an IDLE period to the Gantt chart
        if (proc[0].at > 0) {
            gantt[idx++] = -1; // -1 represents IDLE period
        }
        // Perform SJF scheduling
        while (completed < n) {
            int minIndex = findMinRtProcess(proc, n, currentTime);
            if (minIndex == -1) {
                currentTime++;
                continue;
            }
            // Update remaining time
            proc[minIndex].rt--;
            // Add process ID to the gantt chart array
            gantt[idx++] = proc[minIndex].pid;
            // Update current time, completion time, and check if the process is completed
            currentTime++;
            if (proc[minIndex].rt == 0) {
                ct[minIndex] = currentTime;
                tat[minIndex] = ct[minIndex] - proc[minIndex].at;
                wt[minIndex] = tat[minIndex] - proc[minIndex].bt;
                avgWaitTime += wt[minIndex];
                avgTurnaroundTime += tat[minIndex];
                completed++;
            }
        }
        // Calculate averages
        avgWaitTime /= n;
        avgTurnaroundTime /= n;
        // Print Gantt Chart
        System.out.println("\nGantt Chart:");
        System.out.print("|");
        System.out.println("\nGantt Chart:");
System.out.print("|");

// Print the Gantt Chart
for (int i = 0; i < idx; i++) {
    if (i > 0 && gantt[i] == gantt[i - 1]) {
        System.out.print(""); // Print space if the current process is the same as the previous one
    } else {
        if (gantt[i] == -1) {
            System.out.print(" IDLE |"); // Print IDLE period
        } else {
            System.out.printf(" P%d |", gantt[i]); // Print each process with its ID
        }
    }
}
System.out.print("\n0 ");

// Print the timeline
```

```
for (int i = 1; i <= idx; i++) {
   if (i > 0 && gantt[i] == gantt[i - 1]) {
      System.out.print("");
   } else {
   System.out.printf("   %d|", i);
}}
System.out.println();

      // Print process details
      System.out.println("\nProcess\tBurst Time\tArrival Time\tCompletion Time\tTurnaround Time\tWaiting Time");
      for (int i = 0; i < n; i++) {
         System.out.printf("P%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].bt, proc[i].at, ct[i], tat[i], wt[i]);
      }
      // Print average times
      System.out.printf("\nAverage Waiting Time: %.2f", avgWaitTime);
      System.out.printf("\nAverage Turnaround Time: %.2f\n", avgTurnaroundTime);
   }

   public static void main(String[] args) {
   Scanner scanner = new Scanner(System.in);
   System.out.println("PINKY PAMECHA C114 60004220056");
      System.out.print("Enter the number of processes: ");
      int n = scanner.nextInt();
      Process[] proc = new Process[n];
      System.out.println("\nEnter Burst Time and Arrival Time for each process:");
      for (int i = 0; i < n; i++) {
         proc[i] = new Process();
         proc[i].pid = i + 1;
         System.out.printf("Process %d:\n", proc[i].pid);
         System.out.print("Burst Time: ");
         proc[i].bt = scanner.nextInt();
         System.out.print("Arrival Time: ");
         proc[i].at = scanner.nextInt();
      }
      // Perform SJF preemptive scheduling
      SJF(proc, n);
      scanner.close();
   }
}
```

OUTPUT:



```
PINKY PAMECHA C114 60004220056
Enter the number of processes: 4

Enter Burst Time and Arrival Time for each process:
Process 1:
Burst Time: 18
Arrival Time: 0
Process 2:
Burst Time: 4
Arrival Time: 1
Process 3:
Burst Time: 7
Arrival Time: 2
Process 4:
Burst Time: 2
Arrival Time: 3

Gantt Chart:
|
Gantt Chart:
| P1 | P2 | P4 | P3 | P1 |
0    1|   5|   7|   14|   31|

Process Burst Time      Arrival Time    Completion Time Turnaround Time Waiting Time
P1      18              0               31              31              13
P2      4               1               5               4               0
P3      7               2               14              12              5
P4      2               3               7               4               2

Average Waiting Time: 5.00
Average Turnaround Time: 12.75


...Program finished with exit code 0
Press ENTER to exit console.
```

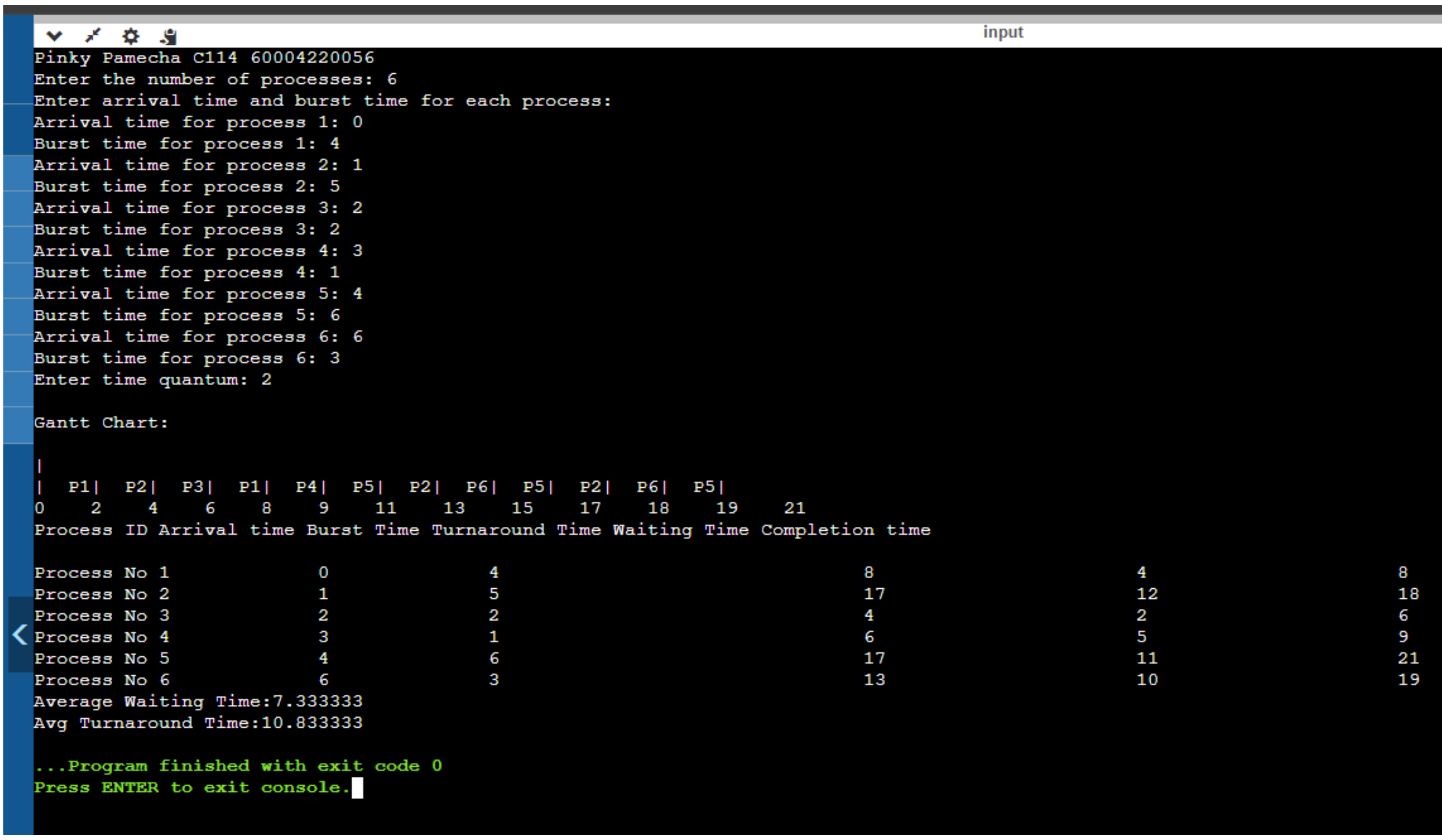
ROUND ROBIN:

CODE:

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class Main {
   private static Queue<Integer> queue = new LinkedList<>();
   private static int[] stamp = new int[50];

   private static void enqueue(int num) {
      queue.add(num);
   }

   private static Integer dequeue() {
      return queue.poll();
```

```
}

private static void sort(int[] arr, int n) {
    for (int i = 0; i < n - 1; i++) {
        boolean swapped = false;
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = true;
            }
        }
        if (!swapped) break;
    }
}

private static int getIndex(int[] arr, int num, int n) {
    for (int i = 0; i < n; i++) {
        if (arr[i] == num) {
            return i;
        }
    }
    return -1;
}

private static void gantt(int[] bt, int[] at, int n, int q, int[] ct) {
    int[] at_new = new int[n];
    int[] p = new int[n];
    int[] bt_new = new int[n];
    System.arraycopy(at, 0, at_new, 0, n);
    sort(at_new, n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (at[j] == at_new[i]) {
                p[i] = j + 1;
                bt_new[i] = bt[j];
            }
        }
    }
    System.out.print("|\t");
    System.out.println();
    int time = 0, id, c = 0;
    int j = 0;
    enqueue(p[j++]);
    while (!queue.isEmpty()) {
        id = dequeue();
        System.out.printf("|  P%d", id);
        time += q;
        while (j < n && at_new[j] <= time) {
            enqueue(p[j++]);
        }
        int index = getIndex(p, id, n);
        if (bt_new[index] <= q) {
            stamp[c] = c == 0 ? bt_new[index] : stamp[c - 1] + bt_new[index];
            bt_new[index] = 0;
            ct[id - 1] = stamp[c]; // Update completion time
        } else {
            stamp[c] = c == 0 ? q : stamp[c - 1] + q;
            bt_new[index] -= q;
            enqueue(id);
         }
         c++;
      }
      System.out.println("|");
      System.out.print("0    ");
      for (int i = 0; i < c; i++) {
         System.out.printf("%d    ", stamp[i]);
      }
      System.out.println();
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Pinky Pamecha C114 60004220056");
      System.out.print("Enter the number of processes: ");
      int n = scanner.nextInt();
      int[] arr_time = new int[n];
      int[] burst_time = new int[n];
      int[] ct = new int[n];

      System.out.println("Enter arrival time and burst time for each process:");
      for (int i = 0; i < n; i++) {
         System.out.printf("Arrival time for process %d: ", i + 1);
         arr_time[i] = scanner.nextInt();
         System.out.printf("Burst time for process %d: ", i + 1);
         burst_time[i] = scanner.nextInt();
      }

      System.out.print("Enter time quantum: ");
      int time_slot = scanner.nextInt();

      System.out.println("\nGantt Chart:\n");
      gantt(burst_time, arr_time, n, time_slot, ct);

      int wait_time = 0, ta_time = 0;
      System.out.println("Process ID Arrival time Burst Time Turnaround Time Waiting Time Completion time");
      for (int i = 0; i < n; i++) {
         int turnaroundTime = ct[i] - arr_time[i];
         int waitingTime = turnaroundTime - burst_time[i];
         wait_time += waitingTime;
         ta_time += turnaroundTime;
         System.out.printf("\nProcess No %d \t\t %d\t\t%d\t\t\t\t %d\t\t\t %d\t\t\t%d", i + 1, arr_time[i], burst_time[i],
turnaroundTime, waitingTime, ct[i]);
      }
      float average_wait_time = wait_time * 1.0f / n;
      float average_turnaround_time = ta_time * 1.0f / n;
      System.out.printf("\nAverage Waiting Time:%f", average_wait_time);
      System.out.printf("\nAvg Turnaround Time:%f", average_turnaround_time);
   }
}
```

OUTPUT:



```
Pinky Pamecha C114 60004220056
Enter the number of processes: 6
Enter arrival time and burst time for each process:
Arrival time for process 1: 0
Burst time for process 1: 4
Arrival time for process 2: 1
Burst time for process 2: 5
Arrival time for process 3: 2
Burst time for process 3: 2
Arrival time for process 4: 3
Burst time for process 4: 1
Arrival time for process 5: 4
Burst time for process 5: 6
Arrival time for process 6: 6
Burst time for process 6: 3
Enter time quantum: 2

Gantt Chart:

|
|  P1|  P2|  P3|  P1|  P4|  P5|  P2|  P6|  P5|  P2|  P6|  P5|
0    2    4    6    8    9    11    13    15    17    18    19    21
Process ID Arrival time Burst Time Turnaround Time Waiting Time Completion time

Process No 1              0              4                                8                       4                      8
Process No 2              1              5                                17                      12                     18
Process No 3              2              2                                4                       2                      6
Process No 4              3              1                                6                       5                      9
Process No 5              4              6                                17                      11                     21
Process No 6              6              3                                13                      10                     19
Average Waiting Time:7.333333
Avg Turnaround Time:10.833333

...Program finished with exit code 0
Press ENTER to exit console.
```