**Scheduling Algorithms**

**Aim**: Write program to implement Process scheduling algorithms. First Come First Serve (FCFS) Scheduling, Shortest-Job-First (SJF) Scheduling, Priority Scheduling (Non-preemption), Round Robin (RR) Scheduling

**Definitions:**

* **FCFS (First-Come, First-Served):** Processes are executed in the order they arrive, with the first process in the ready queue being given CPU access first.
* **SJF (Shortest Job First):** The process with the shortest burst time is selected for execution next, minimizing the average waiting time.
* **RR (Round Robin):** Each process is given a fixed time slice (time quantum) to execute, and if it doesn't finish within that time, it's preempted and added to the end of the ready queue.
* **Priority Scheduling**: Processes are executed based on priority, with the highest priority process being selected for execution first, potentially preempting lower priority processes.
* **Turnaround Time (TAT):** The total time taken for a process to complete, including both waiting time and execution time.
* **Waiting Time (WT):** The total time a process spends waiting in the ready queue before it starts execution.
* **Completion** **Time (CT):** The time at which a process completes its execution.
* **Arrival Time (AT):** The time at which a process arrives in the ready queue and is ready for execution.
* **Burst Time (BT):** The time required by a process to complete its execution without interruption.
* **Response time :**is the time it takes for the CPU to respond to a request from a process. It's the amount of time between when a process arrives and when it first runs
* **Time Quantum:** The maximum amount of time a process is allowed to run in a single CPU burst in a Round Robin scheduling algorithm.
* **Context Switch:** The time required to switch between executing different processes on the CPU, typically involving saving and restoring process state.
* **Code for CPU Scheduling Algorithm**

import java.util.\*;

public class Scheduler{

    public static void main(String[] args)

    {

        Scanner sc = new Scanner(System.in);

        System.out.println("\n\*\*\*\*\*\*\*\*CPU Scheduling Algorithms\*\*\*\*\*\*\*\*\*");

        System.out.println("Name: Harmik Sarvaliya");

        System.out.println("Er.no: 230317 \n");

        System.out.println("Choose a scheduling algorithm to execute:");

**System.out.println("1. First Come First Serve (FCFS)");**

**System.out.println("2. Shortest Job First (SJF)");**

**System.out.println("3. Priority Scheduling");**

**System.out.println("4. Round Robin (RR)\n");**

        System.out.print("Enter your choice (1-4): ");

        int choice = sc.nextInt();

        System.out.println(" ");

        switch (choice){

            case 1:

                executeFCFS();

                break;

            case 2:

                executeSJF();

                break;

            case 3:

                executePriorityScheduling();

                break;

            case 4:

                executeRoundRobin();

                break;

            default:

                System.out.println("Invalid choice. Please choose a number between 1 and 4.");

        }

    }

**static void executeFCFS()** {

        Scanner sc = new Scanner(System.in);

        System.out.println("\*\*\*\*\*\*\*\*First Come First Serve(FCFS)\*\*\*\*\*\*\*\*\*");

        System.out.print("Enter the No.of the Process : ");

        int process = sc.nextInt();

        float avgTAT = 0, avgWT = 0, sumT = 0, sumW = 0;

        float[] at = new float[process];

        float[] bt = new float[process];

        float[] ct = new float[process];

        float[] tat = new float[process];

        float[] wt = new float[process];

        System.out.println("Enter the Arrival Time and Burst Time : ");

        for (int i = 0; i < process; i++)

        {

**at[i] = sc.nextFloat();**

**bt[i] = sc.nextFloat();**

        }

        System.out.println("ARRIVAL TIME | BRUST TIME | FINISH TIME | TURNAROUND TIME | WAITING TIME ");

        float temp = 0;

        for (int i = 0; i < process; i++)

        {

            ct[i] = temp + bt[i];  // completion time

            temp = ct[i];

            tat[i] = ct[i] - at[i]; **//** turnaround time

            wt[i] = tat[i] - bt[i]; //waiting time

            sumT += tat[i]; // sum of wt and tat

            sumW += wt[i];

        }

        for (int i = 0; i < process; i++)

        {

            System.out.println("       " + at[i] + "       " + bt[i] + "          " + ct[i] + "             " + tat[i] + "             " + wt[i]);

        }

        avgTAT = sumT / process;

        avgWT = sumW / process;

        System.out.println("Average Turnaround Time : " + avgTAT);

        System.out.println("Average Waiting Time : " + avgWT);

    }

**static void executeSJF()**     {

        Scanner sc = new Scanner(System.in);

        System.out.println("\*\*\*\*\*\*\*\*Shortest Job First (SJF)\*\*\*\*\*\*\*\*\*");

        System.out.print("Enter number of processes: ");

        int process = sc.nextInt();

        int[] id = new int[process];

        int[] at = new int[process];

        int[] bt = new int[process];

        int[] ct = new int[process];

        int[] tat = new int[process];

        int[] wt = new int[process];

        System.out.println("Enter the Arrival Time and Brust Time : ");

        for (int i = 0; i < process; i++)

        {

            id[i] = i + 1;

            at[i] = sc.nextInt();

            bt[i] = sc.nextInt();

        }

        int[] processActive = new int[process];

        int[] idct = new int[process];

        int executedCount = 0;

        int time = 0;

        while (executedCount < process)

        {

            int minBurstIndex = -1;

            int minbt = Integer.MAX\_VALUE;

            for (int i = 0; i < process; i++)

            {

                if (processActive[i] == 0 && at[i] <= time && bt[i] < minbt)

                {

                    minbt = bt[i];

                    minBurstIndex = i;

                }

            }

            if (minBurstIndex != -1)

            {

                processActive[minBurstIndex] = 1;

                time += bt[minBurstIndex];

                idct[id[minBurstIndex] - 1] = time;

                executedCount++;

            }else{

                time++;

            }

        }

        float avgWaitTime = 0, avgtat = 0;

        for (int i = 0; i < process; ++i)

        {

            ct[i] = idct[id[i] - 1];

            tat[i] = ct[i] - at[i];

            wt[i] = tat[i] - bt[i];

            avgWaitTime += wt[i];

            avgtat += tat[i];

        }

        avgWaitTime /= process;

        avgtat /= process;

        System.out.println("PROCESS | ARRIVAL TIME | BRUST TIME | FINISH TIME | TURNAROUND TIME | WAITING TIME ");

        for (int i = 0; i < process; ++i)

        {

            System.out.println(id[i] + " \t\t " + at[i] + " \t\t " + bt[i]+" \t " + ct[i] + "  \t\t  " + tat[i]+"  \t\t  "+ wt[i]);

        }

        System.out.println("Average Waiting Time: " + avgWaitTime);

        System.out.println("Average Turnaround Time: " + avgtat);

    }

**static void executePriorityScheduling()**

    {

        Scanner sc = new Scanner(System.in);

        System.out.println("\*\*\*\*\*Priority Scheduling (Non-preemption)\*\*\*\*\*\*\*");

        System.out.print("Enter total number of processes : ");

        int process = sc.nextInt();

        int[] id = new int[process];

        int[] at = new int[process];

        int[] bt = new int[process];

        int[] ct = new int[process];

        int[] tat = new int[process];

        int[] wt = new int[process];

        int[] priority = new int[process];

        System.out.println("Enter Process-id, Arrival-Time, Burst-Time, and Priority for each process (Lowest Value High Priority):");

        for (int i = 0; i < process; i++)

        {

            id[i] = sc.nextInt();

            at[i] = sc.nextInt();

            bt[i] = sc.nextInt();

            priority[i] = sc.nextInt();

        }

        for (int i = 0; i < process - 1; i++)

        {

            for (int j = i + 1; j < process; j++)

            {

                if (at[i] > at[j] || (at[i] == at[j] && priority[i] > priority[j])

                        || (at[i] == at[j] && priority[i] == priority[j] && id[i] > id[j]))

                {

                    int temp = id[i];

                    id[i] = id[j];

                    id[j] = temp;

                    temp = at[i];

                    at[i] = at[j];

                    at[j] = temp;

                    temp = bt[i];

                    bt[i] = bt[j];

                    bt[j] = temp;

                    temp = priority[i];

                    priority[i] = priority[j];

                    priority[j] = temp;

                }

            }

        }

        int temp = 0;

        long sumWT = 0, sumTAT = 0;

        for (int i = 0; i < process; i++)

        {

            ct[i] = Math.max(temp, at[i]) + bt[i];

            temp = ct[i];

            tat[i] = ct[i] - at[i];

            wt[i] = tat[i] - bt[i];

            sumWT += wt[i];

            sumTAT += tat[i];

        }

        float avgWT = (float) sumWT / process;

        float avgTAT = (float) sumTAT / process;

        System.out.println("PROCESS ID | ARRIVAL TIME | BRUST TIME | PRIORITY | FINISH TIME | TURNAROUND TIME | WAITING TIME ");

        for (int i = 0; i < process; i++)

        {

            System.out.println(id[i] + " \t\t " + at[i] + " \t\t " + bt[i] + " \t " +

                    priority[i] + " \t\t " + ct[i] + " \t\t "+ tat[i] + " \t\t " + wt[i]);

        }

        System.out.println("Average Waiting Time = " + avgWT);

        System.out.println("Average Turnaround Time = " + avgTAT);

    }

**static void executeRoundRobin()**

    {

        Scanner sc = new Scanner(System.in);

        System.out.println("\*\*\*\*\*\*\*\*Round Robin (RR)\*\*\*\*\*\*\*\*\*");

        System.out.print("Enter total no. of processes: ");

        int process = sc.nextInt();

        int[] id = new int[process];

        int[] at = new int[process];

        int[] bt = new int[process];

        int[] ct = new int[process];

        int[] tat = new int[process];

        int[] wt = new int[process];

        int[] rbt = new int[process];

        int[] key = new int[process];

        System.out.println("\nEnter Arrival Time and Burst Time for each process: ");

        for (int i = 0; i < process; i++)

        {

            id[i] = i + 1;

            at[i] = sc.nextInt();

            bt[i] = sc.nextInt();

            rbt[i] = bt[i];

            key[i] = 0;

        }

        System.out.print("Enter Time quantum: ");

        int tq = sc.nextInt();

        enQ(0);

        key[0] = 1;

        gt=at[0];

        while (!isQEmpty())

        {

            int idx = deQ();

            if (rbt[idx] <= tq)

            {

                gt += rbt[idx];

                rbt[idx] = 0;

                ct[idx] = gt;

            }

            else{

                gt += tq;

                rbt[idx] -= tq;

            }

            for (int i = 0; i < process; i++)

            {

                if (rbt[i] > 0 && at[i] <= gt && key[i] == 0)

                {

                    enQ(i);

                    key[i] = 1;

                }

            }

            if (rbt[idx] > 0)

            {

                enQ(idx);

            }

        }

        float avgTAT = 0, avgWT = 0;

        for (int i = 0; i < process; i++)

        {

**tat[i] = ct[i] - at[i];**

**wt[i] = tat[i] - bt[i];**

**avgTAT += tat[i];**

**avgWT += wt[i];**

        }

        System.out.println("PROCESS | ARRIVAL TIME | BRUST TIME | FINISH TIME | TURNAROUND TIME | WAITING TIME ");

        for (int i = 0; i < process; i++)

        {

            System.out.println(id[i] + "\t\t" + at[i] + "\t\t" + bt[i] + "\t" + ct[i] + "\t\t" + tat[i] + "\t\t" + wt[i]);

        }

        System.out.printf("\nAverage Turn around time = %.2f", avgTAT / process);

        System.out.printf("\nAverage Waiting time = %.2f\n", avgWT / process);

    }

    static int gt = 0;

    static LinkedList<Integer> queue = new LinkedList<>();

**static void enQ(int ele)**

**{**

queue.offer(ele);

**}**

**static int deQ()**

**{**

return queue.poll();

**}**

**static boolean isQEmpty()**

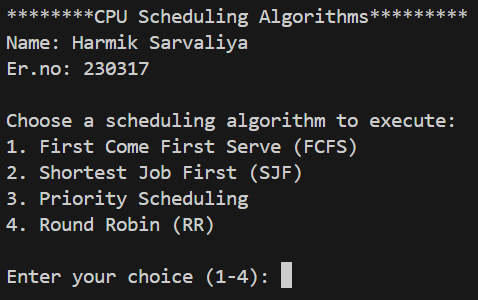
**{**

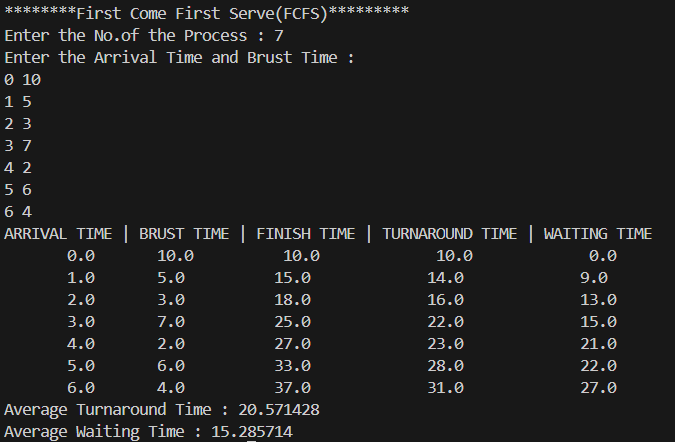
return queue.isEmpty();

**}**

}

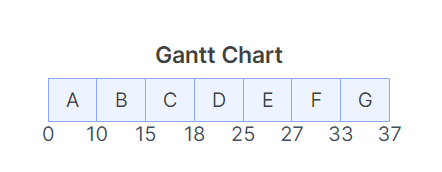
* **Example and Outputs**

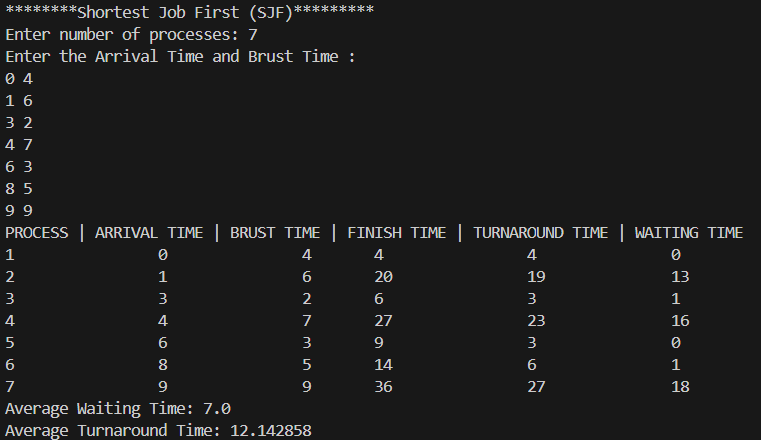


* O**utput :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process** | **Arrival Time(AT)** | **Burst Time(BT)** | **Completion Time(CT)** | **Turnaround Time(TAT)** | **Waiting Time(WT)** |
| **P1** | 0 | 10 | 0 + 10 = 10 | 10 – 0 = 10 | 10 – 1 = 0 |
| **P2** | 1 | 5 | 1 + 14 = 15 | 15 – 1 = 14 | 14 – 5 = 9 |
| **P3** | 2 | 3 | 2 + 16 = 18 | 18 – 2 = 16 | 16 – 3 = 13 |
| **P4** | 3 | 7 | 3 + 22 = 25 | 25 – 3 = 22 | 22 – 7 = 15 |
| **P5** | 4 | 2 | 4 + 23 = 27 | 27 – 4 = 23 | 23 – 2 = 21 |
| **P6** | 5 | 6 | 5 + 28 = 33 | 33 – 5 = 28 | 28 – 6 = 22 |
| **P7** | 6 | 4 | 6 + 31 = 37 | 37 – 6 = 31 | 31 – 4 = 27 |
| **Average TAT and WT** | | | | **144 / 7 = 20.571** | **107 / 7 = 15.286** |

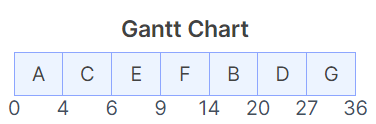
* **Example:**

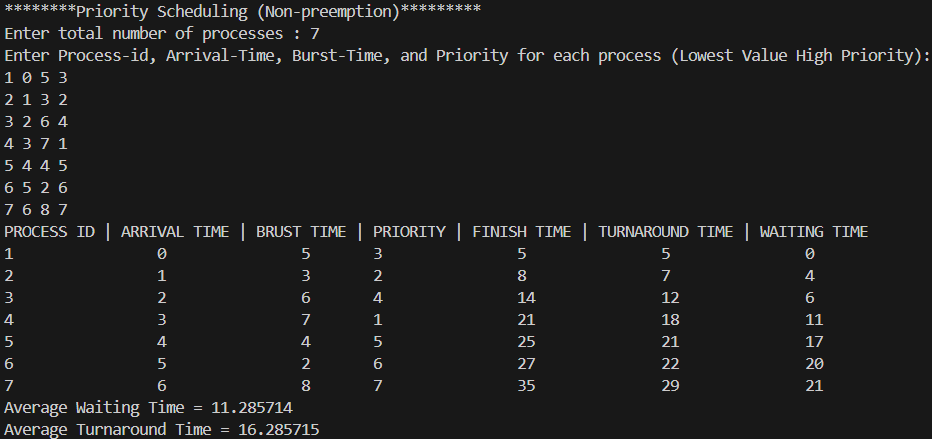


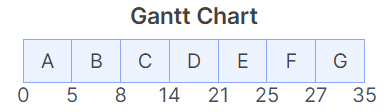
* O**utput :**

* **Example :**

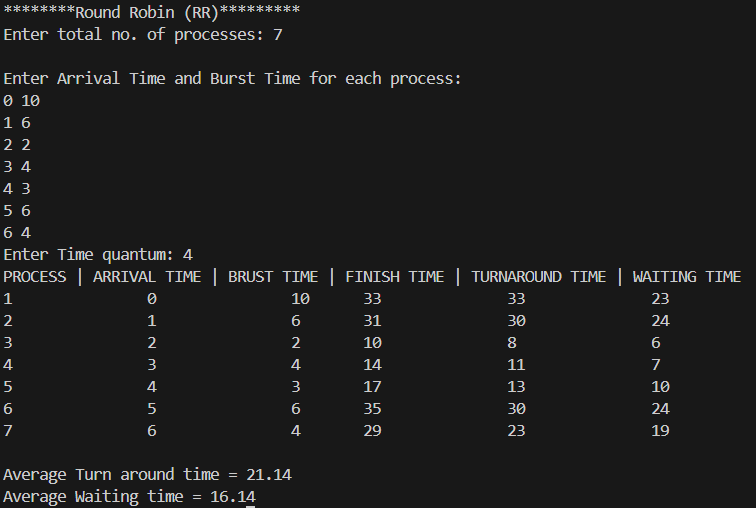
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process** | **Arrival Time(AT)** | **Burst Time(BT)** | **Completion Time(CT)** | **Turnaround Time(TAT)** | **Waiting Time(WT)** |
| **P1** | 0 | 4 | 0 + 4 = 4 | 4 – 0 = 4 | 4 – 4 = 0 |
| **P2** | 1 | 6 | 1 + 19 = 20 | 20 – 1 = 19 | 19 – 6 = 13 |
| **P3** | 3 | 2 | 3 + 3 = 6 | 6 – 3 = 3 | 3 – 2 = 1 |
| **P4** | 4 | 7 | 4 + 23 = 27 | 27 – 4 = 23 | 23 – 7 = 16 |
| **P5** | 6 | 3 | 6 + 3 = 9 | 9 – 6 = 3 | 3 – 3 = 0 |
| **P6** | 8 | 5 | 8 + 6 = 14 | 14 – 8 = 6 | 6 – 5 = 1 |
| **P7** | 9 | 9 | 9 + 27 = 36 | 36 – 9 = 27 | 27 – 9 = 18 |
| **Average TAT and WT** | | | | **85 / 7 = 12.143** | **49 / 7 = 7** |



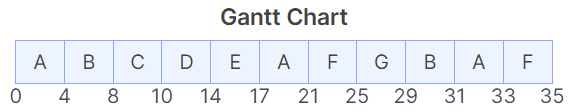
* O**utput :**
* **Example:**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Arrival Time(AT)** | **Burst Time(BT)** | **Priority** | **Completion Time(CT)** | **Turnaround Time(TAT)** | **Waiting Time(WT)** |
| **P1** | 0 | 5 | 3 | 0 + 5 = 5 | 5 – 0 = 5 | 5 – 5 = 0 |
| **P2** | 1 | 3 | 4 | 1 + 7 = 8 | 8 – 1 = 7 | 7 – 3 = 4 |
| **P3** | 2 | 6 | 4 | 2 + 12 = 14 | 14 – 2 = 12 | 12 – 6 = 6 |
| **P4** | 3 | 7 | 1 | 3 + 18 = 21 | 21 – 3 = 18 | 18 – 7 = 11 |
| **P5** | 4 | 4 | 5 | 4 + 21 = 25 | 25 – 4 = 21 | 21 – 4 = 17 |
| **P6** | 5 | 2 | 6 | 5 + 22 = 27 | 27 – 5 = 22 | 22 – 2 = 20 |
| **P7** | 6 | 8 | 7 | 6 + 29 = 35 | 35 – 6 = 29 | 29 – 8 = 21 |
| **Average TAT and WT** | | | | | **114 / 7 = 16.286** | **79 / 7 = 11.286** |

* O**utput:**
* **Example:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process** | **Arrival Time(AT)** | **Burst Time(BT)** | **Completion Time(CT)** | **Turnaround Time(TAT)** | **Waiting Time(WT)** |
| **P1** | **0** | **10** | **0 + 33 = 33** | **33 – 0 = 33** | **33 – 10 = 23** |
| **P2** | **1** | **6** | **1 + 30 = 31** | **31 – 1 = 30** | **30 – 6 = 24** |
| **P3** | **2** | **2** | **2 + 8 = 10** | **10 – 2 = 8** | **8 – 2 = 6** |
| **P4** | **3** | **4** | **3 + 11 = 14** | **14 – 3 = 11** | **11 – 4 = 7** |
| **P5** | **4** | **3** | **4 + 13 = 17** | **17 – 4 = 13** | **13 – 3 = 10** |
| **P6** | **5** | **6** | **5 + 30 = 35** | **35 – 5 = 30** | **30 – 6 = 24** |
| **P7** | **6** | **4** | **6 + 23 = 29** | **29 – 6 = 23** | **23 – 4 = 19** |
| **Average TAT and WT** | | | | **148 / 7 = 21.143** | **113 / 7 = 16.143** |



Date : Signature:

**Practical 15**

**(Round Robin with Context switch)**

**Aim**: Write program to implement Process scheduling algorithms. Round Robin (RR) Scheduling with context switch.

* **Context switching :** It is a technique used by an operating system (OS) to switch processes from one state to another. It's an essential feature of a multiprogramming or multitasking operating system. It allows multiple processes to share a single CPU, while making it appear as though the CPU is executing multiple processes simultaneously
* **Example: Round Robin with Context switch**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process** | **Arrival Time(AT)** | **Burst Time(BT)** | **Completion Time(CT)** | **Turnaround Time(TAT)** | **Waiting Time(WT)** |
| **P1** | 0 | 3 | 0+12 = 12 | 12 – 0 = 12 | 12 – 3 = 9 |
| **P2** | 1 | 6 | 1+32 = 33 | 33 – 1 = 32 | 32 – 6 = 26 |
| **P3** | 1 | 10 | 1+51 = 52 | 52 – 1 = 51 | 51 – 10= 41 |
| **P4** | 2 | 1 | 2+8 = 10 | 10 – 2 = 8 | 8 – 1 = 7 |
| **P5** | 4 | 5 | 3+37 = 41 | 41 – 4 = 37 | 37 – 5 = 32 |
| **P6** | 5 | 2 | 5+13 = 18 | 18 – 5 = 13 | 13 – 2 = 11 |
| **P7** | 6 | 7 | 6+42 = 49 | 49 – 6 = 42 | 42 – 7 = 36 |
| **Average TAT and WT** | | | | **196/7=28** | **162/7=23.14** |

* **Gantt chart:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **P1** |  | **P2** |  | **P3** |  | **P4** |  | **P1** |  | **P5** |  | **P6** |  | **P2** |  | **P7** |  | **P3** |  | **P5** |  | **P2** |  | **P7** |  | **P3** |  | **P5** |  | **P7** |  | **P3** |  | **P7** |  | **P3** |

0 2 3 5 6 8 9 10 11 12 13 15 16 18 19 21 22 24 25 27 28 30 31 33 34 36 37 39 40 41 42 44 45 47 48 49 50 52

* **Code : Round Robin with Context switch**

import java.util.\*;

class RoundRobinCS

{

static int gt = 0; // gt-executed time of cpu

static LinkedList<Integer> queue = new LinkedList<>();

**static void enQ(int ele){**

**queue.offer(ele);** // insert a new element

**}**

**static int deQ(){**

**return queue.poll();** // removes an element from queue

**}**

**static boolean isQEmpty() {**

**return queue.isEmpty();**

**}**

public static void main(String[] args) {

System.out.println("\*\*\*\*\*\*\*\*Round Robin(RR)\*\*\*\*\*\*\*\*\*");

System.out.println("Name : Harmik Sarvaliya");

System.out.println("Er.no : 230317");

Scanner sc = new Scanner(System.in);

System.out.print("Enter total no. of processes: ");

int process = sc.nextInt();

int[] id = new int[process];

int[] at = new int[process];

int[] bt = new int[process];

int[] ct = new int[process];

int[] tat = new int[process];

int[] wt = new int[process];

int[] rbt = new int[process]; // remaining burst time

int[] key = new int[process]; // determines whether process is present in

queue or not

System.out.println("\nEnter Arrival Time and Burst Time for each process: ");

for (int i = 0; i < process; i++)

{

id[i] = i + 1;

at[i] = sc.nextInt();

bt[i] = sc.nextInt();

rbt[i] = bt[i];

key[i] = 0;

}

System.out.print("Enter Time quantum: ");

int tq = sc.nextInt();

System.out.print("Enter Context Switch Time: ");

int contextSwitchTime = sc.nextInt();

enQ(0); // Enqueue first process ID(pID starts from 1 but LinkedList index starts from 0)

key[0] = 1; // Mark the first process as present in the queue

gt = at[0];

int lastExecuted = -1; // Index of the last executed process

while (!isQEmpty()){

int idx = deQ(); // Dequeue a process ID

if (lastExecuted != -1 && lastExecuted != idx) { **// Context Switch**

gt += contextSwitchTime;

}

int remainingTime = rbt[idx]; //Stores remaining burst time before execution

if (remainingTime <= tq) {

gt += remainingTime;

rbt[idx] = 0;

ct[idx] = gt;

}

else {

gt += tq;

rbt[idx] -= tq;

}

lastExecuted = idx; // Update the last executed process

// Enqueue processes that have arrived and have remaining burst time

for (int i = 0; i < process; i++) {

if (rbt[i] > 0 && at[i] <= gt && key[i] == 0){

enQ(i); // Enqueue the process ID

key[i] = 1; // Mark the process as present in the queue

}

}

// Enqueue the current process again if it still has remaining burst time

if (rbt[idx] > 0) {

enQ(idx);

}

}

float avgTAT = 0, avgWT = 0;

for (int i = 0; i < process; i++) {

tat[i] = ct[i] - at[i];

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

avgTAT += tat[i];

avgWT += wt[i];

}

System.out.println("PROCESS | ARRIVAL TIME | BRUST TIME | FINISH TIME | TURNAROUND TIME | WAITING TIME ");

for (int i = 0; i < process; i++) {

System.out.println(id[i] + "\t\t" + at[i] + "\t\t" + bt[i] + "\t" + ct[i] + "\t\t" + tat[i] + "\t\t" + wt[i]);

}

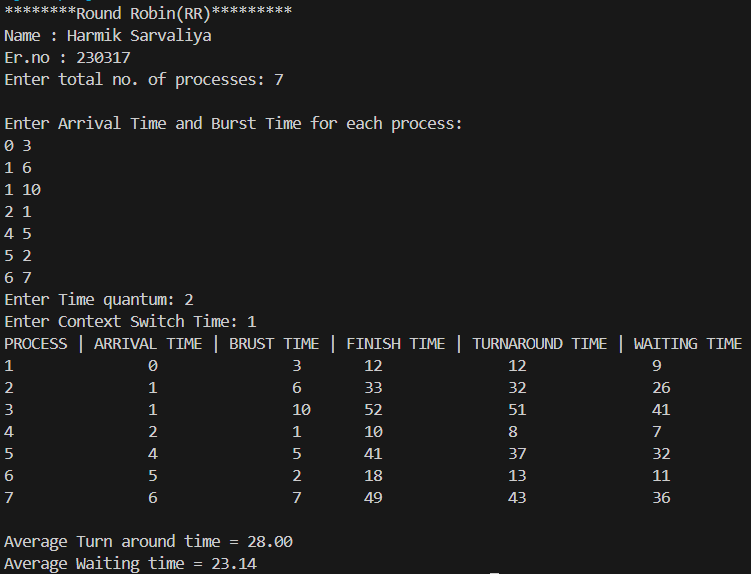
System.out.printf("\nAverage Turn around time = %.2f", avgTAT / process);

System.out.printf("\nAverage Waiting time = %.2f\n", avgWT / process);

}

}

* **Output:**

****

Date : Signature: