```
//FCFS
import java.util.*;

class Process {
    int pid, at, bt, ct, tat, wt;
}

public class FCFS {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of processes: ");
        int n = sc.nextInt();

        Process[] p = new Process[n];
        for (int i = 0; i < n; i++) {
            p[i] = new Process();
            System.out.print("Enter Arrival Time of Process " + (i + 1) + ": ");
            p[i].at = sc.nextInt();
            System.out.print("Enter Burst Time of Process " + (i + 1) + ": ");
            p[i].bt = sc.nextInt();
            p[i].pid = i + 1;
        }

        // Sort processes by Arrival Time
        Arrays.sort(p, Comparator.comparingInt(a -> a.at));

        int time = 0;
        for (int i = 0; i < n; i++) {
            if (time < p[i].at)
                time = p[i].at;
            p[i].ct = time + p[i].bt;
            time = p[i].ct;
            p[i].tat = p[i].ct - p[i].at;
            p[i].wt = p[i].tat - p[i].bt;
        }

        System.out.println("\nPID\tAT\tBT\tCT\tTAT\tWT");
        for (int i = 0; i < n; i++) {
            System.out.println(p[i].pid + "\t" + p[i].at + "\t" + p[i].bt + "\t" + p[i].ct + "\t" + p[i].tat + "\t" +
p[i].wt);
        }
```

```
        double avgTAT = 0, avgWT = 0;
        for (Process pr : p) {
            avgTAT += pr.tat;
            avgWT += pr.wt;
        }
        avgTAT /= n;
        avgWT /= n;

        System.out.println("\nAverage Turnaround Time: " + avgTAT);
        System.out.println("Average Waiting Time: " + avgWT);
    }
}
```

```
//SJF
import java.util.*;

class SJF {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter no. of processes: ");
        int n = sc.nextInt();
        int at[] = new int[n], bt[] = new int[n], ct[] = new int[n], tat[] = new int[n], wt[] = new int[n];
        boolean done[] = new boolean[n];

        for (int i = 0; i < n; i++) {
            System.out.print("AT of P" + (i + 1) + ": ");
            at[i] = sc.nextInt();
            System.out.print("BT of P" + (i + 1) + ": ");
            bt[i] = sc.nextInt();
        }

        int time = 0, completed = 0;
        while (completed < n) {
            int idx = -1, min = Integer.MAX_VALUE;
            for (int i = 0; i < n; i++)
                if (!done[i] && at[i] <= time && bt[i] < min) { min = bt[i]; idx = i; }
            if (idx == -1) { time++; continue; }

            ct[idx] = time + bt[idx];
```

```
            tat[idx] = ct[idx] - at[idx];
            wt[idx] = tat[idx] - bt[idx];
            done[idx] = true;
            completed++;
            time = ct[idx];
        }


        double avgTAT = 0, avgWT = 0;
        System.out.println("\nPID\tAT\tBT\tCT\tTAT\tWT");
        for (int i = 0; i < n; i++) {
            avgTAT += tat[i]; avgWT += wt[i];
            System.out.println("P" + (i + 1) + "\t" + at[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + tat[i] + "\t" + wt[i]);
        }
        System.out.println("\nAvg TAT: " + (avgTAT / n) + "\nAvg WT: " + (avgWT / n));
    }
}



//priority


import java.util.*;

class PriorityScheduling {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter no. of processes: ");
        int n = sc.nextInt();
        int at[] = new int[n], bt[] = new int[n], pr[] = new int[n];
        int ct[] = new int[n], tat[] = new int[n], wt[] = new int[n];
        boolean done[] = new boolean[n];

        for (int i = 0; i < n; i++) {
            System.out.print("AT of P" + (i + 1) + ": ");
            at[i] = sc.nextInt();
            System.out.print("BT of P" + (i + 1) + ": ");
            bt[i] = sc.nextInt();
            System.out.print("Priority of P" + (i + 1) + ": ");
            pr[i] = sc.nextInt();
        }
```

```
        int time = 0, completed = 0;
        while (completed < n) {
            int idx = -1, high = Integer.MAX_VALUE;
            for (int i = 0; i < n; i++)
                if (!done[i] && at[i] <= time && pr[i] < high) { high = pr[i]; idx = i; }
            if (idx == -1) { time++; continue; }

            ct[idx] = time + bt[idx];
            tat[idx] = ct[idx] - at[idx];
            wt[idx] = tat[idx] - bt[idx];
            done[idx] = true;
            completed++;
            time = ct[idx];
        }

        double avgTAT = 0, avgWT = 0;
        System.out.println("\nPID\tAT\tBT\tPR\tCT\tTAT\tWT");
        for (int i = 0; i < n; i++) {
            avgTAT += tat[i]; avgWT += wt[i];
            System.out.println("P" + (i + 1) + "\t" + at[i] + "\t" + bt[i] + "\t" + pr[i] + "\t" + ct[i] + "\t" + tat[i]
+ "\t" + wt[i]);
        }
        System.out.println("\nAvg TAT: " + (avgTAT / n) + "\nAvg WT: " + (avgWT / n));
    }
}


//Round Robin


import java.util.*;

class RR {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter no. of processes: ");
        int n = sc.nextInt();
        int at[] = new int[n], bt[] = new int[n], rt[] = new int[n], ct[] = new int[n];
        int tat[] = new int[n], wt[] = new int[n];
        System.out.print("Enter Time Quantum: ");
        int tq = sc.nextInt();
```

```
for (int i = 0; i < n; i++) {
    System.out.print("AT of P" + (i + 1) + ": ");
    at[i] = sc.nextInt();
    System.out.print("BT of P" + (i + 1) + ": ");
    bt[i] = sc.nextInt();
    rt[i] = bt[i];
}

Queue<Integer> q = new LinkedList<>();
int time = 0, completed = 0;
boolean inQ[] = new boolean[n];

while (completed < n) {
    for (int i = 0; i < n; i++)
        if (at[i] <= time && !inQ[i] && rt[i] > 0) { q.add(i); inQ[i] = true; }

    if (q.isEmpty()) { time++; continue; }

    int i = q.poll();
    int exec = Math.min(rt[i], tq);
    rt[i] -= exec;
    time += exec;

    for (int j = 0; j < n; j++)
        if (at[j] <= time && !inQ[j] && rt[j] > 0) { q.add(j); inQ[j] = true; }

    if (rt[i] == 0) {
        ct[i] = time;
        tat[i] = ct[i] - at[i];
        wt[i] = tat[i] - bt[i];
        completed++;
    } else q.add(i);
}

double avgTAT = 0, avgWT = 0;
System.out.println("\nPID\tAT\tBT\tCT\tTAT\tWT");
for (int i = 0; i < n; i++) {
    avgTAT += tat[i]; avgWT += wt[i];
    System.out.println("P" + (i + 1) + "\t" + at[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + tat[i] + "\t" + wt[i]);
}
System.out.println("\nAvg TAT: " + (avgTAT / n));
```

```
        System.out.println("Avg WT: " + (avgWT / n));
    }
}



//page replacement

import java.util.*;

public class PageReplacement {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of frames: ");
        int f = sc.nextInt();
        System.out.print("Enter number of pages: ");
        int n = sc.nextInt();

        int pages[] = new int[n];
        System.out.println("Enter page reference string:");
        for (int i = 0; i < n; i++) pages[i] = sc.nextInt();

        System.out.println("\n1. LRU\n2. Optimal");
        System.out.print("Enter choice: ");
        int ch = sc.nextInt();

        if (ch == 1) lru(pages, f);
        else if (ch == 2) optimal(pages, f);
        else System.out.println("Invalid choice");
    }

    static void lru(int[] pages, int f) {
        ArrayList<Integer> frame = new ArrayList<>();
        int faults = 0;
        for (int p : pages) {
            if (!frame.contains(p)) {
                if (frame.size() == f) frame.remove(0);
                faults++;
            } else frame.remove((Integer)p);
            frame.add(p);
            System.out.println("Frames: " + frame);
        }
```

```
        System.out.println("Total Page Faults (LRU): " + faults);
    }

    static void optimal(int[] pages, int f) {
        ArrayList<Integer> frame = new ArrayList<>();
        int faults = 0;
        for (int i = 0; i < pages.length; i++) {
            int p = pages[i];
            if (!frame.contains(p)) {
                if (frame.size() == f) {
                    int far = -1, idx = -1;
                    for (int j = 0; j < f; j++) {
                        int val = frame.get(j);
                        int next = Integer.MAX_VALUE;
                        for (int k = i + 1; k < pages.length; k++)
                            if (pages[k] == val) { next = k; break; }
                        if (next > far) { far = next; idx = j; }
                    }
                    frame.set(idx, p);
                } else frame.add(p);
                faults++;
            }
            System.out.println("Frames: " + frame);
        }
        System.out.println("Total Page Faults (Optimal): " + faults);
    }
}
```