**Experiment No. 8**

**NAME:** Omkar Sunil Khanvilkar **ROLL NO:** 07

**CLASS:** TY\_IT\_B **BATCH**: 2

**DATE OF PERFORMANCE:** 26/09/2024

**Write a program to compute the finish time, turnaround time and waiting time for the following algorithms:**

**a) First come First serve**

**b) Shortest Job First (Preemptive and Non-Preemptive)**

**c) Priority (Preemptive and Non-Preemptive)**

**d) Round robin**

## **CODE:**

import java.util.Arrays;  
import java.util.Scanner;  
  
public class SchedulingAlgorithms{  
  
 public static void findFCFS(int n, int[] bt, int[] at, int[] ft, int[] tat, int[] wt) {  
 int ct = 0;  
 for (int i = 0; i < n; i++) {  
 if (ct < at[i])  
 ct = at[i];  
 ft[i] = ct + bt[i];  
 ct = ft[i];  
 tat[i] = ft[i] - at[i];  
 wt[i] = tat[i] - bt[i];  
 }  
 }  
  
 public static void findSJFNonPreemptive(int n, int[] bt, int[] at, int[] ft, int[] tat, int[] wt) {  
 int[] done = new int[n];  
 int[] btRem = Arrays.*copyOf*(bt, n);  
 int time = 0;  
 for (int i = 0; i < n; i++) {  
 int minBt = Integer.*MAX\_VALUE*;  
 int shortest = -1;  
 for (int j = 0; j < n; j++) {  
 if (at[j] <= time && done[j] == 0 && btRem[j] < minBt) {  
 minBt = btRem[j];  
 shortest = j;  
 }  
 }  
 if (shortest != -1) {  
 time += btRem[shortest];  
 ft[shortest] = time;  
 tat[shortest] = ft[shortest] - at[shortest];  
 wt[shortest] = tat[shortest] - bt[shortest];  
 done[shortest] = 1;  
 } else {  
 time++;  
 i--;  
 }  
 }  
 }

public static void findSJFP(int n, int[] bt, int[] at, int[] ft, int[] tat, int[] wt) {  
 int[] btRem = Arrays.*copyOf*(bt, n);  
 int time = 0, count = 0;  
 while (count < n) {  
 int minBt = Integer.*MAX\_VALUE*;  
 int shortest = -1;  
 for (int i = 0; i < n; i++) {  
 if (at[i] <= time && btRem[i] < minBt && btRem[i] > 0) {  
 minBt = btRem[i];  
 shortest = i;  
 }  
 }  
 if (shortest != -1) {  
 time++;  
 btRem[shortest]--;  
 if (btRem[shortest] == 0) {  
 count++;  
 ft[shortest] = time;  
 tat[shortest] = ft[shortest] - at[shortest];  
 wt[shortest] = tat[shortest] - bt[shortest];  
 }  
 } else {  
 time++;  
 }  
 }  
 }

public static void findPriorityNonPreemptive(int n, int[] bt, int[] at, int[] pr, int[] ft, int[] tat, int[] wt) {  
 int[] done = new int[n];  
 int time = 0;  
 for (int i = 0; i < n; i++) {  
 int minPr = Integer.*MAX\_VALUE*;  
 int highest = -1;  
 for (int j = 0; j < n; j++) {  
 if (at[j] <= time && done[j] == 0 && pr[j] < minPr) {  
 minPr = pr[j];  
 highest = j;  
 }  
 }  
 if (highest != -1) {  
 time += bt[highest];  
 ft[highest] = time;  
 tat[highest] = ft[highest] - at[highest];  
 wt[highest] = tat[highest] - bt[highest];  
 done[highest] = 1;  
 } else {  
 time++;  
 i--;  
 }  
 }  
 }

public static void findPriorityP(int n, int[] bt, int[] at, int[] pr, int[] ft, int[] tat, int[] wt) {  
 int[] btRem = Arrays.*copyOf*(bt, n);  
 int time = 0, count = 0;  
 while (count < n) {  
 int minPr = Integer.*MAX\_VALUE*;  
 int highest = -1;  
 for (int i = 0; i < n; i++) {  
 if (at[i] <= time && pr[i] < minPr && btRem[i] > 0) {  
 minPr = pr[i];  
 highest = i;  
 }  
 }  
 if (highest != -1) {  
 time++;  
 btRem[highest]--;  
 if (btRem[highest] == 0) {  
 count++;  
 ft[highest] = time;  
 tat[highest] = ft[highest] - at[highest];  
 wt[highest] = tat[highest] - bt[highest];  
 }  
 } else {  
 time++;  
 }  
 }  
 }  
  
 public static void findRR(int n, int[] bt, int[] at, int[] ft, int[] tat, int[] wt, int quantum) {  
 int[] btRem = Arrays.*copyOf*(bt, n);  
 int time = 0, count = 0;  
 while (count < n) {  
 for (int i = 0; i < n; i++) {  
 if (at[i] <= time && btRem[i] > 0) {  
 if (btRem[i] > quantum) {  
 time += quantum;  
 btRem[i] -= quantum;  
 } else {  
 time += btRem[i];  
 btRem[i] = 0;  
 ft[i] = time;  
 tat[i] = ft[i] - at[i];  
 wt[i] = tat[i] - bt[i];  
 count++;  
 }  
 }  
 }  
 }  
 }

public static void printTimes(int n, int[] pr, int[] at, int[] bt, int[] ft, int[] tat, int[] wt) {  
 System.*out*.println("Process\t\tPriority\tArrival Time\tBurst Time\tFinish Time\tTurnaround Time\tWaiting Time");  
 for (int i = 0; i < n; i++) {  
 System.*out*.printf(" %d\t\t\t%d\t\t\t%d\t\t\t\t%d\t\t\t%d\t\t\t%d\t\t\t\t%d%n",  
 i + 1, pr[i], at[i], bt[i], ft[i], tat[i], wt[i]);  
 }  
 }

public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int n;  
 int[] bt = new int[20], at = new int[20], pr = new int[20], ft = new int[20], tat = new int[20], wt = new int[20];  
 int quantum;  
  
 System.*out*.print("Enter number of processes: ");  
 n = scanner.nextInt();  
 System.*out*.print("Enter burst times: ");  
 for (int i = 0; i < n; i++) {  
 bt[i] = scanner.nextInt();  
 }  
 System.*out*.print("Enter arrival times: ");  
 for (int i = 0; i < n; i++) {  
 at[i] = scanner.nextInt();  
 }  
 System.*out*.print("Enter priorities: ");  
 for (int i = 0; i < n; i++) {  
 pr[i] = scanner.nextInt();  
 }  
 System.*out*.print("Enter time quantum for Round Robin: ");  
 quantum = scanner.nextInt();  
  
 System.*out*.println("\nFirst Come First Serve:");  
 *findFCFS*(n, bt, at, ft, tat, wt);  
 *printTimes*(n, pr, at, bt, ft, tat, wt);  
  
 System.*out*.println("\nShortest Job First (Non-Preemptive):");  
 *findSJFNonPreemptive*(n, bt, at, ft, tat, wt);  
 *printTimes*(n, pr, at, bt, ft, tat, wt);  
  
 System.*out*.println("\nShortest Job First (Preemptive):");  
 *findSJFP*(n, bt, at, ft, tat, wt);  
 *printTimes*(n, pr, at, bt, ft, tat, wt);  
  
 System.*out*.println("\nPriority (Non-Preemptive):");  
 *findPriorityNonPreemptive*(n, bt, at, pr, ft, tat, wt);  
 *printTimes*(n, pr, at, bt, ft, tat, wt);  
  
 System.*out*.println("\nPriority (Preemptive):");  
 *findPriorityP*(n, bt, at, pr, ft, tat, wt);  
 *printTimes*(n, pr, at, bt, ft, tat, wt);  
  
 System.*out*.println("\nRound Robin:");  
 *findRR*(n, bt, at, ft, tat, wt, quantum);  
 *printTimes*(n, pr, at, bt, ft, tat, wt);  
  
 scanner.close();  
 }  
}

## **OUTPUT:**

