Round Robin

#include <stdio.h>

#define MAX 100

void roundRobin(int n, int at[], int bt[], int quant) {

    int ct[n], tat[n], wt[n], rem\_bt[n];

    int queue[MAX], front = 0, rear = 0;

    int time = 0, completed = 0, visited[n];

    for (int i = 0; i < n; i++) {

        rem\_bt[i] = bt[i];

        visited[i] = 0;

    }

    queue[rear++] = 0;

    visited[0] = 1;

    while (completed < n) {

        int index = queue[front++];

        if (rem\_bt[index] > quant) {

            time += quant;

            rem\_bt[index] -= quant;

        } else {

            time += rem\_bt[index];

            rem\_bt[index] = 0;

            ct[index] = time;

            completed++;

        }

        for (int i = 0; i < n; i++) {

            if (at[i] <= time && rem\_bt[i] > 0 && !visited[i]) {

                queue[rear++] = i;

                visited[i] = 1;

            }

        }

        if (rem\_bt[index] > 0) {

            queue[rear++] = index;

        }

        if (front == rear) {

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

                if (rem\_bt[i] > 0) {

                    queue[rear++] = i;

                    visited[i] = 1;

                    break;

                }

            }

        }

    }

    float total\_tat = 0, total\_wt = 0;

    printf("P#\tAT\tBT\tCT\tTAT\tWT\n");

    for (int i = 0; i < n; i++) {

        tat[i] = ct[i] - at[i];

        wt[i] = tat[i] - bt[i];

        total\_tat += tat[i];

        total\_wt += wt[i];

        printf("%d\t%d\t%d\t%d\t%d\t%d\n", i + 1, at[i], bt[i], ct[i], tat[i], wt[i]);

    }

    printf("Average TAT: %.2f\n", total\_tat / n);

    printf("Average WT: %.2f\n", total\_wt / n);

}

int main() {

    int n, quant;

    printf("Enter number of processes: ");

    scanf("%d", &n);

    int at[n], bt[n];

    for (int i = 0; i < n; i++) {

        printf("Enter AT and BT for process %d: ", i + 1);

        scanf("%d %d", &at[i], &bt[i]);

    }

    printf("Enter time quantum: ");

    scanf("%d", &quant);

    roundRobin(n, at, bt, quant);

    return 0;

}

