```
#include <stdio.h>


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


void sortByArrival(struct Process p[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (p[j].at > p[j + 1].at) {
                struct Process temp = p[j];
                p[j] = p[j + 1];
                p[j + 1] = temp;
            }
        }
    }
}


void sjfScheduling(struct Process p[], int n) {
    sortByArrival(p, n);
    int completed = 0, currentTime = 0;
    int totalWT = 0, totalTAT = 0, totalRT = 0;


    while (completed < n) {
        int minIndex = -1, minBT = 9999;
        for (int i = 0; i < n; i++) {
            if (p[i].at <= currentTime && p[i].bt < minBT && p[i].tat == 0) {
                minBT = p[i].bt;
                minIndex = i;
            }
        }

        if (minIndex == -1) {
            currentTime++;
            continue;
        }

        p[minIndex].wt = currentTime - p[minIndex].at;
        p[minIndex].tat = p[minIndex].wt + p[minIndex].bt;
        p[minIndex].rt = p[minIndex].wt;
        p[minIndex].ct = currentTime + p[minIndex].bt;
        currentTime += p[minIndex].bt;
        completed++;

        totalWT += p[minIndex].wt;
        totalTAT += p[minIndex].tat;
        totalRT += p[minIndex].rt;
    }

    printf("\nPID\tAT\tBT\tCT\tWT\tTAT\tRT\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\n", p[i].pid, p[i].at, p[i].bt, p[i].ct, p[i].wt, p[i].tat, p[i].rt);
    }
    printf("\nAverage WT: %.2f", (float)totalWT / n);
    printf("\nAverage TAT: %.2f", (float)totalTAT / n);
    printf("\nAverage RT: %.2f\n", (float)totalRT / n);
}

int main() {
    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    struct Process p[n];

    for (int i = 0; i < n; i++) {
        printf("Enter AT & BT for P%d: ", i + 1);
        scanf("%d %d", &p[i].at, &p[i].bt);
        p[i].pid = i + 1;
        p[i].wt = p[i].tat = p[i].rt = p[i].ct = 0;
    }

    sjfScheduling(p, n);
    return 0;
}

#include <stdio.h>
#include <limits.h>
```

```
struct Process {

    int pid, bt, at, wt, tat, rt, ct, remaining_bt;

};


void sortByArrival(struct Process p[], int n) {

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

        for (int j = 0; j < n - i - 1; j++) {

            if (p[j].at > p[j + 1].at) {

                struct Process temp = p[j];

                p[j] = p[j + 1];

                p[j + 1] = temp;

            }

        }

    }

}


void sjfPreemptive(struct Process p[], int n) {

    sortByArrival(p, n);

    int completed = 0, currentTime = 0, shortest = -1;

    int totalWT = 0, totalTAT = 0, totalRT = 0;

    int isFirstResponse[n];


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

        p[i].remaining_bt = p[i].bt;

        isFirstResponse[i] = 1;

    }
```

```
while (completed < n) {
    int minBT = INT_MAX;
    shortest = -1;

    for (int i = 0; i < n; i++) {
        if (p[i].at <= currentTime && p[i].remaining_bt > 0 && p[i].remaining_bt < minBT) {
            minBT = p[i].remaining_bt;
            shortest = i;
        }
    }

    if (shortest == -1) {
        currentTime++;
        continue;
    }

    if (isFirstResponse[shortest]) {
        p[shortest].rt = currentTime - p[shortest].at;
        isFirstResponse[shortest] = 0;
    }

    p[shortest].remaining_bt--;
    currentTime++;

    if (p[shortest].remaining_bt == 0) {
            p[shortest].ct = currentTime;

            p[shortest].tat = p[shortest].ct - p[shortest].at;

            p[shortest].wt = p[shortest].tat - p[shortest].bt;

            totalWT += p[shortest].wt;

            totalTAT += p[shortest].tat;

            totalRT += p[shortest].rt;

            completed++;

        }

    }


    printf("\nPID\tAT\tBT\tCT\tWT\tTAT\tRT\n");

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

        printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\n", p[i].pid, p[i].at, p[i].bt, p[i].ct, p[i].wt, p[i].tat, p[i].rt);

    }

    printf("\nAverage WT: %.2f", (float)totalWT / n);

    printf("\nAverage TAT: %.2f", (float)totalTAT / n);

    printf("\nAverage RT: %.2f\n", (float)totalRT / n);

}


int main() {

    int n;

    printf("Enter number of processes: ");

    scanf("%d", &n);

    struct Process p[n];


    for (int i = 0; i < n; i++) {
        printf("Enter AT & BT for P%d: ", i + 1);

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

        p[i].pid = i + 1;

        p[i].wt = p[i].tat = p[i].rt = p[i].ct = 0;

    }



    sjfPreemptive(p, n);

    return 0;

}


#include <stdio.h>
#include <stdlib.h>

struct Process {
    int id, AT, BT, CT, TAT, WT, RT, priority, completed;
};

void sortByPriority(struct Process p[], int n, int currentTime) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (!p[i].completed && !p[j].completed) {
                if ((p[i].AT <= currentTime && p[j].AT <= currentTime && p[i].priority > p[j].priority) ||
                    (p[i].AT > currentTime && p[j].AT <= currentTime)) {
                    struct Process temp = p[i];
                    p[i] = p[j];
                    p[j] = temp;
                }
            }
        }
    }
}

void calculatePriorityNonPreemptive(struct Process p[], int n) {
    int completed = 0, currentTime = 0;
    float totalWT = 0, totalTAT = 0;

    while (completed < n) {
        sortByPriority(p, n, currentTime);
        int index = -1;

        for (int i = 0; i < n; i++) {
            if (!p[i].completed && p[i].AT <= currentTime) {
                index = i;
                break;
            }
        }

        if (index == -1) {
            currentTime++;
        } else {
            p[index].CT = currentTime + p[index].BT;
            p[index].TAT = p[index].CT - p[index].AT;
            p[index].WT = p[index].TAT - p[index].BT;
            p[index].RT = currentTime - p[index].AT;
            p[index].completed = 1;
            totalWT += p[index].WT;
            totalTAT += p[index].TAT;
            currentTime = p[index].CT;
            completed++;
        }
    }

    printf("\nProcess\tAT\tBT\tPT\tCT\tTAT\tWT\tRT\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (p[j].id == i + 1) {
                printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",
                       p[j].id, p[j].AT, p[j].BT, p[j].priority, p[j].CT, p[j].TAT, p[j].WT, p[j].RT);
                break;
            }
        }
    }

    printf("\nAverage WT: %.2f", totalWT / n);
    printf("\nAverage TAT: %.2f\n", totalTAT / n);
}

int main() {
    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    struct Process p[n];
    for (int i = 0; i < n; i++) {
        p[i].id = i + 1;
        printf("Enter Arrival Time (AT), Burst Time (BT) & Priority for process %d: ", i + 1);
        scanf("%d %d %d", &p[i].AT, &p[i].BT, &p[i].priority);
        p[i].completed = 0;
    }

    calculatePriorityNonPreemptive(p, n);
    return 0;
}
```

On Thu, Mar 13, 2025 at 12:24 AM Dhruthi Vijay <dhruthivijay.ad23@bmsce.ac.in> wrote:

SJF-P

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define MAX 10

struct process {
    int id, AT, BT, CT, TAT, WT, RT, remaining_BT;
    int completed;
};
void calculate_SJF_Preemptive(struct process p[], int n) {
    int completed = 0, currentTime = 0;
    for (int i = 0; i < n; i++) {
        p[i].remaining_BT = p[i].BT;
    }
    while (completed < n) {
        int shortest = -1, minBT = INT_MAX;
        for (int i = 0; i < n; i++) {
            if (!p[i].completed && p[i].AT <= currentTime && p[i].remaining_BT < minBT) {
                minBT = p[i].remaining_BT;
                shortest = i;
            }
        }

        if (shortest == -1) {
            currentTime++;
        } else {
            if (p[shortest].remaining_BT == p[shortest].BT)
                p[shortest].RT = currentTime - p[shortest].AT;
            p[shortest].remaining_BT--;
            currentTime++;

            if (p[shortest].remaining_BT == 0) {
                p[shortest].CT = currentTime;
                p[shortest].TAT = p[shortest].CT - p[shortest].AT;
                p[shortest].WT = p[shortest].TAT - p[shortest].BT;
                p[shortest].completed = 1;
                completed++;
            }
        }
    }
}

void display(struct process p[], int n) {
    printf("\nProcess\tAT\tBT\tCT\tTAT\tWT\tRT\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\n", p[i].id, p[i].AT, p[i].BT, p[i].CT, p[i].TAT, p[i].WT,
p[i].RT);
    }
}
int main() {
    int n, choice;
    struct process p[MAX];

    printf("Enter number of processes: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        p[i].id = i + 1;
        printf("Enter Arrival Time (AT) for process %d: ", i + 1);
        scanf("%d", &p[i].AT);
        printf("Enter Burst Time (BT) for process %d: ", i + 1);
        scanf("%d", &p[i].BT);
        p[i].completed = 0;
    }

    calculate_SJF_Preemptive(p, n);
    display(p, n);

    return 0;
}
```