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

typedef struct process_info
{
        char pname[20];
        int at, bt, ct, bt1;
        struct process_info *next;
} NODE;

struct gantt_chart
{
        int start, end;
        char pname[30];
} s[100], s1[100];

int n, k, time;
NODE *first, *last;

void accept_info()
{
        NODE *p;

        printf("Enter no of process : ");
        scanf("%d", &n);
        printf("\nPNAME\tATIME\tCPUB\n");

        for (int i = 0; i < n; i++)
        {
                p = (NODE *)malloc(sizeof(NODE));

                scanf("%s%d%d", p->pname, &p->at, &p->bt);

                p->bt1 = p->bt;
                p->next = NULL;

                if (first == NULL)
                        first = p;
                else
                        last->next = p;

                last = p;
        }
}

void print_input() // extra
{
        NODE *p;
        p = first;

        printf("pname\tat\tbt\n");
        while (p != NULL)
        {
                printf("%s\t%d\t%d\n", p->pname, p->at, p->bt1);
                p = p->next;
        }
}

void print_output()
{
        NODE *p = first;
        float avg_tat = 0, avg_wt = 0;

        printf("\npname\tat\tbt\tct\ttat\twt\n");

        while (p != NULL)
        {
                int tat = p->ct - p->at;
                int wt = tat - p->bt;

                avg_tat += tat;
                avg_wt += wt;

                printf("%s\t%d\t%d\t%d\t%d\t%d\n", p->pname, p->at, p->bt, p->ct,
tat, wt);

                p = p->next;
        }

        printf("\nAvg TAT=%.2f\nAvg WT=%.2f\n\n", (avg_tat / n), (avg_wt / n));
}

NODE *get_sjf()
{
        NODE *p = first, *min_p = NULL;
        int min = 9999;

        while (p != NULL)
        {
                if (p->at <= time && p->bt1 != 0 && p->bt1 < min)
                {
                        min = p->bt1;
                        min_p = p;
                }
                p = p->next;
        }
        return min_p;
}
```

```
void sjfp()
{
        int prev = 0, n1 = 0;
        NODE *p;

        while (n1 != n)
        {
                p = get_sjf();

                if (p == NULL)
                {
                        time++;
                        s[k].start = prev;
                        strcpy(s[k].pname, "*");
                        s[k].end = time;

                        prev = time;
                        k++;
                }
                else
                {
                        time++; // diff
                        s[k].start = prev;
                        strcpy(s[k].pname, p->pname);
                        s[k].end = time;

                        prev = time;
                        k++;

                        p->ct = time;
                        p->bt1--; // diff

                        if (p->bt1 == 0) // extra
                                n1++;
                }

                print_input(); // extra
        }
}

void print_gantt_chart()
{
        int i, j, m;

        s1[0] = s[0];

        for (i = 1, j = 0; i < k; i++)
        {
                if (strcmp(s1[j].pname, s[i].pname) == 0)
```

```
                        s1[j].end = s[i].end;
                else
                        s1[++j] = s[i];
        }

        printf("%d", s1[0].start);
        for (i = 0; i <= j; i++)
        {
                m = (s1[i].end - s1[i].start);

                for (k = 0; k < m / 2; k++)
                        printf("-");

                printf("%s", s1[i].pname);

                for (k = 0; k < (m + 1) / 2; k++)
                        printf("-");

                printf("%d", s1[i].end);
        }
}

int main()
{
        accept_info();
        sjfp();
        print_output();
        print_gantt_chart();
        return 0;
}
```