5) write a program to implement SJF scheduling

```
#include<stdio.h>
#define MAX_process 10

typedef struct {
    int pid;
    int arr;
    int burst;
    int comp;
    int turn;
    int wait;
    int processed;
} proc;

void sjf_np(proc procs[], int n) {
    int curr_time=0;
    int total_~~time~~=0; comp_time=0;
    int total_wait_time=0;
    int total_turnaround_time=0;

    while(1) {
        int shortest_job = -1;
        int shortest_burst= 9999;
        for(int i=0; i<n; i++) {
            if(procs[i].arr<=curr_time &&
                procs[i].processed == 0) {
                if(procs[i].burst < shortest_burst
                    shortest_burst = procs[i].burst
                    shortest_job=i;
                }
            }
        }
    }
```

PAGE NO :
DATE :

```
        if (shortest_job == -1){
            break;
        }

        procs[shortest_job].comp = curr_time + procs[shortest_job].
                                    burst;
        procs[shortest_job].turn = procs[shortest_job].comp -
                                    procs[shortest_job].burst;
        procs[shortest_job].wait = procs[shortest_job].turn
                                    - procs[shortest_job].burst;
        if(procs[shortest_job].wait < 0)
            procs[shortest_job].wait = 0;

        total_comp_time += procs[shortest_job].comp;
        total_wait_time += procs[shortest_job].wait;
        total_turnaround_time += procs[shortest_job].turn;

        procs[shortest_job].processed = 1;
        curr_time = procs[shortest_job].comp;
    }

    double avg_turnaround_time = (double) total_turn
                                    around_time / n;
    double avg_waiting_time = (double) total_wait_
                                    time / n;
    printf("process ID \t Arrival Time \t Burst Time \t
            Completion time \t waiting time \t turnarond");
    for(int i=0; i<n; i++){
        printf("%d \t\t %d \t\t %d \t\t %d \t\t %d \t\t
                %d \t\t %d",
            procs[i].pid, procs[i].arr, procs[i].burst,
            procs[i].comp, procs[i].wait, procs[i].turn);
    }
```

```
printf("\n Average turnaround Time: %.2f\n",
                avg_turnaround_time);
printf("Average waiting time: %.2f\n",
                avg_waiting_time);
}
int main(){
    int n;
    proc procs[Max_process];
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    for(int i=0; i<n; i++){
        procs[i].pid = i+1;
        printf("Enter arrival time for process %d:",
                                    i+1);
        scanf("%d", &procs[i].arr);
        printf("Enter burst time for process %d:
                                    i+1);
        scanf("%d", &procs[i].burst);
        procs[i].processed = 0;
    }
    SJF_NP(procs, n);
    return 0;
}
```

Output:

Enter the number of process: 5

Enter arrival time for process: 1 2 2 1 3 (arrival time)

Enter burst time for process: 4 0 5 2 (burst time)

Process	AT	BT	CT	WT	TAT
1	1	4	7	2	6
2	2	5	16	9	14
3	2	1	8	6	7
4	1	6	6	0	5
5	3	2	11	6	8