Non-Preemptive Shortest Job First

```c
#include<stdio.h>
struct process
{
    int id,WT,AT,BT,TAT;
};
struct process a[10];

// function for swapping
void swap(int *b,int *c)
{
    int tem;
    tem=*c;
    *c=*b;
    *b=tem;
}

//Driver function
int main()
{
    int n,check_ar=0;
    int Cmp_time=0;
    float Total_WT=0,Total_TAT=0,Avg_WT,Avg_TAT;
    printf("Enter the number of process \n");
    scanf("%d",&n);
    printf("Enter the Arrival time and Burst time of the process\n");
    printf("AT BT\n");
    for(int i=0;i<n;i++)
    {
        scanf("%d%d",&a[i].AT,&a[i].BT);
        a[i].id=i+1;
        // here we are checking that arrival time
        // of the process are same or different
        if(i==0)
         check_ar=a[i].AT;

        if(check_ar!=a[i].AT )
         check_ar=1;

    }
    // if process are arrived at the different time
    // then sort the process on the basis of AT
    if(check_ar!=0)
    {
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n-i-1;j++)
            {
                if(a[j].AT>a[j+1].AT)
                {
                      swap(&a[j].id,&a[j+1].id);
                      swap(&a[j].AT,&a[j+1].AT);
                      swap(&a[j].BT,&a[j+1].BT);
                }
            }
```

```
    }
}

// logic of SJF non preemptive algo
// if all the process are arrived at different time
if(check_ar!=0)
{
    a[0].WT=a[0].AT;
    a[0].TAT=a[0].BT-a[0].AT;
    Cmp_time=a[0].TAT;
    Total_WT=Total_WT+a[0].WT;
    Total_TAT=Total_TAT+a[0].TAT;
    for(int i=1;i<n;i++)
    {
        int min=a[i].BT;
        for(int j=i+1;j<n;j++)
        {
            if(min>a[j].BT && a[j].AT<=Cmp_time)
            {
                  min=a[j].BT;
                  swap(&a[i].id,&a[j].id);
                  swap(&a[i].AT,&a[j].AT);
                  swap(&a[i].BT,&a[j].BT);
            }

        }
        a[i].WT=Cmp_time-a[i].AT;
        Total_WT=Total_WT+a[i].WT;
        // completion time of the process
        Cmp_time=Cmp_time+a[i].BT;

        // Turn Around Time of the process
        // compl-Arival
        a[i].TAT=Cmp_time-a[i].AT;
        Total_TAT=Total_TAT+a[i].TAT;

    }
}

// if all the process are arrived at same time
else
{
    for(int i=0;i<n;i++)
    {
        int min=a[i].BT;
        for(int j=i+1;j<n;j++)
        {
            if(min>a[j].BT && a[j].AT<=Cmp_time)
            {
                  min=a[j].BT;
                  swap(&a[i].id,&a[j].id);
                  swap(&a[i].AT,&a[j].AT);
                  swap(&a[i].BT,&a[j].BT);
            }

        }
```

```
            a[i].WT=Cmp_time-a[i].AT;

             // completion time of the process
            Cmp_time=Cmp_time+a[i].BT;

            // Turn Around Time of the process
            // compl-Arrival
            a[i].TAT=Cmp_time-a[i].AT;
            Total_WT=Total_WT+a[i].WT;
            Total_TAT=Total_TAT+a[i].TAT;

        }

    }

    Avg_WT=Total_WT/n;
    Avg_TAT=Total_TAT/n;

    // Printing of the results
    printf("The process are\n");
    printf("ID WT TAT\n");
    for(int i=0;i<n;i++)
    {
        printf("%d\t%d\t%d\n",a[i].id,a[i].WT,a[i].TAT);
    }

    printf("Avg waiting time is:- %f\n",Avg_WT);
    printf("Avg turn around time is:- %f",Avg_TAT);
    return 0;
}
```

OutPut

```
Enter the number of process
4
Enter the Arrival time and Burst time of the process
AT BT
0 2
2 3
2 4
3 3
The process are
ID WT TAT
1   0   2
2   0   3
4   2   5
3   6   10
Avg waiting time is:- 2.000000
Avg turn around time is:- 5.000000
```