```c
//Round Robin Scheduling
#include <stdio.h>
#define MAX 100

#define MAXq 100

struct caq
{
    int array[MAXq];
    int front;
    int rear;
};

void push(caq *cir, int x)
{
    if((cir->rear + 1)%MAXq == cir->front)
        return;
    else
    {
        if(cir->rear == MAXq-1)
        {
            cir->rear = (cir->rear+1)%MAXq;
            cir->array[cir->rear] = x;
            return;
        }
        cir->array[cir->rear] = x;
        cir->rear = (cir->rear+1)%MAXq;
    }
}

int pop(caq *cir)
{
    if(cir->rear == cir->front)
        return -1;
    int rr = cir->array[cir->front];
    cir->front = (cir->front+1)%MAXq;
    return rr;
}

struct process
{
    int AT;
    int BT;
    int id;
    int CT;
    int WT;
    int TAT;
    int rem;
};

void swap(process &a, process &b)
{
    process t = a;
    a = b;
    b = t;
}

int get_partion(process A[], int start , int end)
{
    int pivot = A[end].AT;
    int i=start-1;
    for(int j=start ; j<end ; j++)
    {
        if(A[j].AT <= pivot)
        {
            i++;
            swap(A[i], A[j]);
        }
    }
    swap(A[end], A[i+1]);
    return i+1;
}

void quicksort(process A[], int start, int end)
{
    if(start < end)
    {
        int partion = get_partion(A, start, end);
        quicksort(A, start, partion-1);
        quicksort(A, partion+1, end);
    }
}

void simulate(process P[], int n, int quanta)
{
    int count=0;
    caq cir;
    cir.front = cir.rear = 0;
    push(&cir, 0);
    for(int time=P[0].AT; count!=n;)
    {
        int to_execute = pop(&cir),flag=quanta;
        if(to_execute != -1)
        {
            int start = time;
            if(P[to_execute].rem < quanta)
            {
                flag=P[to_execute].rem;
                time+=P[to_execute].rem;
                P[to_execute].rem = 0;
            }
            else
            {
                P[to_execute].rem -= quanta;
                time+=quanta;
            }
            if (P[to_execute].rem <= 0)
            {
                count++;
                P[to_execute].CT = time;
                P[to_execute].TAT = P[to_execute].CT - P[to_execute].AT;
            }
            for(int i=0 ; i<n ; i++)
            {
                if(P[i].AT >= start && P[i].rem>0 && P[i].AT<=time && i!
=to_execute)
                {
                    push(&cir, i);
                }
            }
            if (P[to_execute].CT == 0)
                push(&cir, to_execute);
        }
        else
        {
            time++;
        }
        for(int i=0 ; i<n ; i++)
        {
            if(i!=to_execute && P[i].AT < time && P[i].rem>0)
            {
                if(P[i].WT == 0 && i!=0)
                    P[i].WT += (time-P[i].AT);
                else
                    P[i].WT+=flag;
            }
        }
    }
```

```c
}

int main()
{
    process P[101];
    P[MAX].rem = 99999;
    int n,temp;
    int quanta = 5;
    printf("Number of processes : ");
    scanf("%d", &n);
    for(int i=0 ; i<n ; i++)
    {
        printf("Process %d:\n", i+1);
        printf("AT : ");
        scanf("%d", &temp);
        P[i].AT = temp;
        printf("BT : ");
        scanf("%d", &temp);
        P[i].BT = temp;
        P[i].id = i+1;
        P[i].CT = 0;
        P[i].WT = 0;
        P[i].TAT = 0;
        P[i].rem = P[i].BT;
    }
    quicksort(P, 0, n);
    simulate(P, n, quanta);
    printf("\n\n");
    printf("P\tAT\tBT\tCT\tTAT\tWT\n");
    float avgWT=0,avgTAT=0;
    for (int i = 0; i < n; ++i)
    {
        printf("%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);
        avgTAT+=P[i].TAT;
        avgWT+=P[i].WT;
    }
    printf("Average Turn Around Time : %f\n", (avgTAT/n)*1.0);
    printf("Average Wating Time : %f\n", (avgWT/n)*1.0);
}

/*
OUTPUT

Number of processes : 4
Process 1:
AT : 1
BT : 9
Process 2:
AT : 2
BT : 5
Process 3:
AT : 3
BT : 3
Process 4:
AT : 4
BT : 4


P   AT  BT  CT  TAT WT
1   1   9   22  21  12
2   2   5   11  9   4
3   3   3   14  11  8
4   4   4   18  14  10
Average Turn Around Time : 13.750000
Average Wating Time : 8.500000
*/
```