Round Robin

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

struct process {
    char name[5];
    int AT, BT, ST[20], WT, FT, TAT, pos, CT,
 };

int quant;

int main() {
    int n, temp;
    printf("Enter the number of process: \
    ~~for~~ scanf("%d", &n);
    struct process p[n];
    printf(" Enter the arrival time & burst
    for (int i=0; i<n; i++) {
        printf("\nprocess %d:\n", i+1);
        sprintf(p[i].name, "p%d", i+1);
        printf( "Arrival time: ");
        scanf(" %d", &p[i].AT);
        printf(" Burst time: ");
        scanf("%d", &p[i].BT);
        p[i].pos = i+1;
    }

  printf("Enter the quantum: ");
  scanf("%d", &quant);
```

PAGE NO :
DATE :

```
int c=n, s[n][20];
 float time=0, mini=INT_MAX; b[n], a[n];

 int index=-1;
 for (int i=0; i<n; i++){
     b[i]= p[i].BT;
     a[i]= p[i].AT;
     for(int j=0 ; j<20; j++){
         s[i][j]= -1;
     }
 }
 int tot_wt=0, tot_tat=0, tot_rt=0;
 bool flag=false;
 while (c!=0){
     mini= INT_MAX;
     flag= false;
     for (int i=0 ; i<n; i++){
         float p = time+0.1;
         if (a[i]<=p && mini>a[i] && b[i]>0){
             index= i;
             mini= a[i];
             flag= true;
         }
     }
     if (!flag){
         time++;
         continue;
     }
     int j=0
     while (s[index][j] != -1){
         j++;
     }
if(s[index][i]==-1){
    s[index][i]= time;
    p[index].ST[i]= time;
  }
if (b[index] <= quant){
    time += b[index];
    b[index]=0;
  }
else
    time +=quant;
    b[index]-=quant;
  }
if (b[index]>0){
    a[index] = time+0.1;
}

if (b[index]==0){
    c--;
    p[index].FT= time;
    p[index].WT= p[index].FT- p[index].AT-
                                  p[index].BT;
    tot_wt += p[index].BT+ p[index].WT;
    p[index].TAT+= p[index].TAT;
    p[index].CT = time;
    p[index].RT= p[index].ST[0]-p[inde
    tot_rt += p[index].RT;
  }
}

printf("\n process \t Name \t arrival \t bur
        \t start \t final \t completion \t tot
      \t Response time \n");
```

PAGE NO :
DATE :

```
for(int i=0; i<n; i++){
    printf("%d \t %s\t %d \t %d \t \t",
        p[i].pos, p[i].name, p[i].AT, p[i].BT);
    int j=0;
    while (s[i][j] != -1){
        printf("%d", p[i].ST[j]);
        j++;
    }
    printf("\t %d \t %d \t %d \t %d",
        p[i].FT, p[i].CT, p[i].TAT, p[i].WT,
        p[i].RT);
}
double avg_wt= (double) tot_wt /n;
double avg_tat = (double) tot_tat /n;
double avg_rt = (double) tot_rt /n;

printf("\n the avg WT : %f \n", avg_wt);
printf("\n the avg TAT: %f \n", avg_tat);
printf("\n the avg Response: %f \n", avg_rt);
return 0;
}
```

<u>output</u>

```
Enter no.of process : 5
Enter time quantum: 2
Enter arival time & burst time : 0 5 1 3 2 1
Pid   at   bt   ct   tat   wt   rt
1     0    5    13   13    8    0
2     1    3    12   11    8    1
3     2    1    5    3     2    2
4     3    2    9    6     4    4
5     4    3    14   10    7    5

average turn around time: 8.6
average waiting time: 5.8
```