Program 4: Write a program to implement the Shortest Remaining Time First (Shortest job first preemptive) scheduling algorithm and find the average turnaround time, waiting time, completion time and response time for overall process. Also Print Gantt chart for it.

Solution:

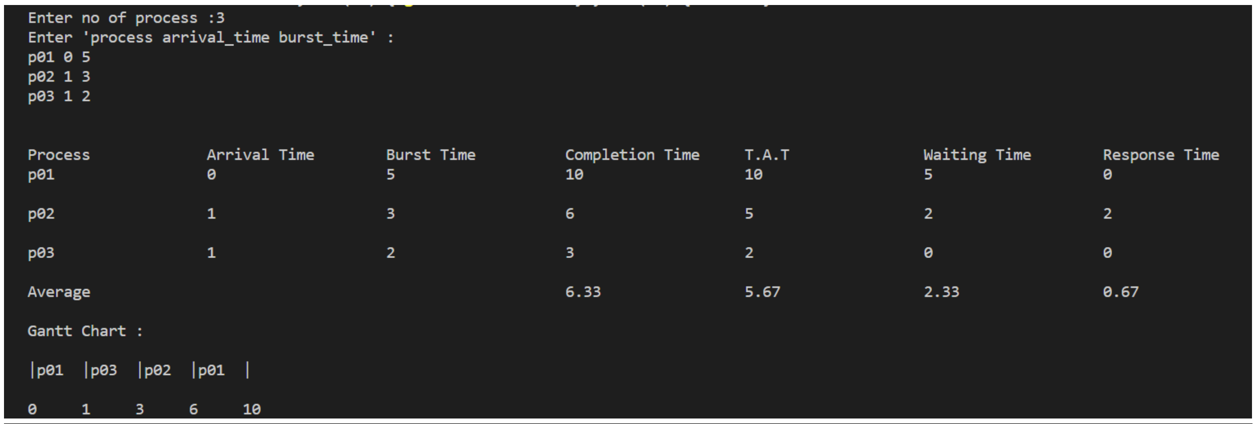
Source Code:

```
#include<stdio.h>

#include<string.h>

void sort(int arr[][6],char str[][10], int at, int bt, char p[], int n, int m);

void print(int n, char str[][10], int arr[][6] );

void ganttChart(int time[],char gantt[][10], int m, int l);


int main(){

   char process[10], gantt[100][10];

   int time[100];

   int arrival_time,burst_time,n;

   printf("Enter no of process :");

   scanf("%d",&n);


   int arr[n][6];

   int temp[n+1];

   char str[n][10];
```

```
printf("Enter 'process arrival_time burst_time' :\n");
scanf("%s",str[0]);
scanf("%d",&arr[0][0]);
scanf("%d",&arr[0][1]);
for (int i=1; i<n; i++){
    scanf("%s",process);
    scanf("%d",&arrival_time);
    scanf("%d",&burst_time);
    int j=0;
    while (j<i && arr[j][0]<=arrival_time){
        j++;
    }
    sort(arr,str,arrival_time,burst_time,process,i,j);
}

for (int i=0; i<n; i++){
    arr[i][5]=-1;
    temp[i]=arr[i][1];
}

temp[n]=1000;
time[0]=arr[0][0];
int l=1, m=0, count=0, prev=0, last;
for (int t=0; count<n; t++){
    int min=n;
    for (int i=0; i<n; i++){
        if (arr[i][0]<=t && temp[i]<temp[min] && temp[i]>0){
            min=i;
        }
    }

    if (arr[min][5]==-1){
        arr[min][5]=t-arr[min][0];
    }

    if (prev!=min){
        strcpy(gantt[m],str[prev]);
        time[l]=t;
        l++;
        m++;
        prev=min;
    }

    temp[min]--;
    if (temp[min]==0){
        count++;
        arr[min][2]=t+1;
        last=min;
    }
}

strcpy(gantt[m],str[last]);
time[l]=arr[last][2];
l++;
m++;

for (int i=0; i<n; i++){
    arr[i][3]=arr[i][2]-arr[i][0];
    arr[i][4]=arr[i][3]-arr[i][1];
}
print(n,str,arr);
ganttChart(time,gantt,m,l);
return 0;
}

void sort(int arr[][6],char str[][10], int at, int bt, char p[], int n, int m){
for (int i=n-1; i>=m; i--){
    arr[i+1][0]=arr[i][0];
    arr[i+1][1]=arr[i][1];
    strcpy(str[i+1],str[i]);
}
arr[m][0]=at;
arr[m][1]=bt;
strcpy(str[m],p);
}

void print(int n, char str[][10], int arr[][6] ){
float avg;
float sum;
char title[7][20]={"Process","Arrival Time","Burst Time","Completion Time","T.A.T",
                "Waiting Time","Response Time"};

printf("\n\n");
for (int i=0; i<7; i++){
    printf("%-20s",title[i]);
}
printf("\n");
for (int i=0; i<n; i++){
    printf("%-20s",str[i]);
    for (int j=0; j<6; j++){
        printf("%-20d",arr[i][j]);
    }
    printf("\n\n");
}
printf("%-60s","Average");
for (int j=2; j<6; j++){
    sum=0;
    for (int i=0; i<n; i++){
        sum+=arr[i][j];
    }
    avg=sum/n;
    printf("%-20.2f",avg);
}
printf("\n\n");
}

void ganttChart(int time[],char gantt[][10], int m, int l){
printf("Gantt Chart :\n\n");
printf("|");
for (int i=0; i<m; i++){
    printf("%-5s|",gantt[i]);
}
printf("\n\n");
for (int i=0; i<l; i++){
    printf("%-6d",time[i]);
}
```

```
}
```

Output:



```
Enter no of process :3
Enter 'process arrival_time burst_time' :
p01 0 5
p02 1 3
p03 1 2


Process             Arrival Time        Burst Time          Completion Time     T.A.T               Waiting Time        Response Time
p01                 0                   5                   10                  10                  5                   0

p02                 1                   3                   6                   5                   2                   2

p03                 1                   2                   3                   2                   0                   0

Average                                                     6.33                5.67                2.33                0.67

Gantt Chart :

|p01  |p03  |p02  |p01  |

0     1     3     6     10
```