



**L** OVELY  
**P** ROFESSIONAL  
**U** NIVERSITY

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**SUMBITTED TO LOVELY PROFESSION UNIVERSITY**

In partial complete of the requirement of the award of  
**DEGREE OF BACHELOR OF TECHNOLOGY**  
**{CSE}**

**NAME: K.SIDDARTHA**

**ROLL NO: 39**

**REGISTRATION NO: 1 220027 1**

**SECTION : K22ZC**

## ❖ QUESTION

Design a scheduler following non-preemptive scheduling approach to schedule the processes that arrives at different units and having burst time double the arrival time. Scheduler selects the process with largest burst time from the queue for the execution. Process is not being preempted until it finishes its service time. Compute the average waiting time and average turnaround time. What should be the average waiting time if processes are executed according to Shortest Job First scheduling approach with the same attribute values.

# ANS:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
int i,n,p[10]={1,2,3,4,5,6,7,8,9,10},min,k=1,btime=0;
```

```
int bt[10],temp,j,at[10],wt[10],tt[10],sum=1,fsum=0,ct[10];
```

```
float wavg=0,tavg=0,tsum=0,wsum=0;
```

```
printf("Enter the number of processes: ");
```

```
scanf("%d",&n);
```

```
{
```

```
if(n==0)
```

```
printf("Enter at least 1 Process\n\n\n\n");
```

```
else
```

```
for(i=0;i<n;i++)
```

```
{
```

```
printf("\nEnter the Arrival Time of p%d process: ",i+1);
```

```
scanf("%d",&at[i]);
```

```
printf("\nBrust time of p%d process= ",i+1);
```

```
printf("%d\n",bt[i]=2*at[i]);
```

```
}
```

```
for(i=0;i<n;i++)
```

```
//Sorting According to Arrival Time
```

```
{
```

```
for(j=0;j<n;j++)
```

```
{
```

```
if(at[i]<at[j])
```

```
{  
temp=p[j];  
p[j]=p[i];  
p[i]=temp;  
temp=at[j];  
at[j]=at[i];  
at[i]=temp;  
temp=bt[j];  
bt[j]=bt[i];  
bt[i]=temp;  
}  
}  
}
```

```
for(j=0;j<n;j++)  
{  
btime=btime+bt[j];  
min=bt[k];  
for(i=k;i<n;i++)  
{  
if (btime>=at[i] && bt[i]<min)  
{  
temp=p[k];  
p[k]=p[i];  
p[i]=temp;  
temp=at[k];
```

```
at[k]=at[i];
at[i]=temp;
temp=bt[k];
bt[k]=bt[i];
bt[i]=temp;
}
}
k++;
}
wt[0]=0;
sum=1;
ct[0]=1;
for(i=1;i<=n;i++)
{
    if(at[i-1]==0)
        sum=sum+1;
    sum=sum+bt[i-1];
    ct[i-1]=sum;
    tt[i]=ct[i-1]-at[i-1];
    tsum=tsum+tt[i];
    wt[i]=tt[i]-bt[i-1];
    wsum=wsum+wt[i];
}

wavg=(wsum/n);
tavg=(tsum/n);
```

```

printf("*****");
printf("\n\t\t\t RESULT:-");
printf("\n\t Shortest Job First,Non-Preemptive\n\n");
printf("\n\t\t-----");
printf("\n\t\t| Process |\tBT\t|\tAT\t|\tWT\t|\tTAT\t|\tCT\t|" );
printf("\n\t\t\t-----");
for(i=0;i<n;i++)
{
printf("\n\t\t| p%d\t |\t%d\t|\t%d\t|\t%d\t|\t%d\t|",
p[i],bt[i],at[i],wt[i+1],tt[i+1],ct[i]);
}
printf("\n\t\t\t-----");

```

```

printf("\n\n*****
*****");

```

```

printf("\n\t\t\t\t Order of execution.\n");

```

```

printf("\n");

```

```

printf("\t\t\t\t\t-----\n\t\t\t\t\t");

```

```

for(i=0;i<n;i++)
{
printf("| Process[%d] |\n\t\t\t\t\t",p[i]);
}

```

```

printf("-----\n\t\t\t\t\t");

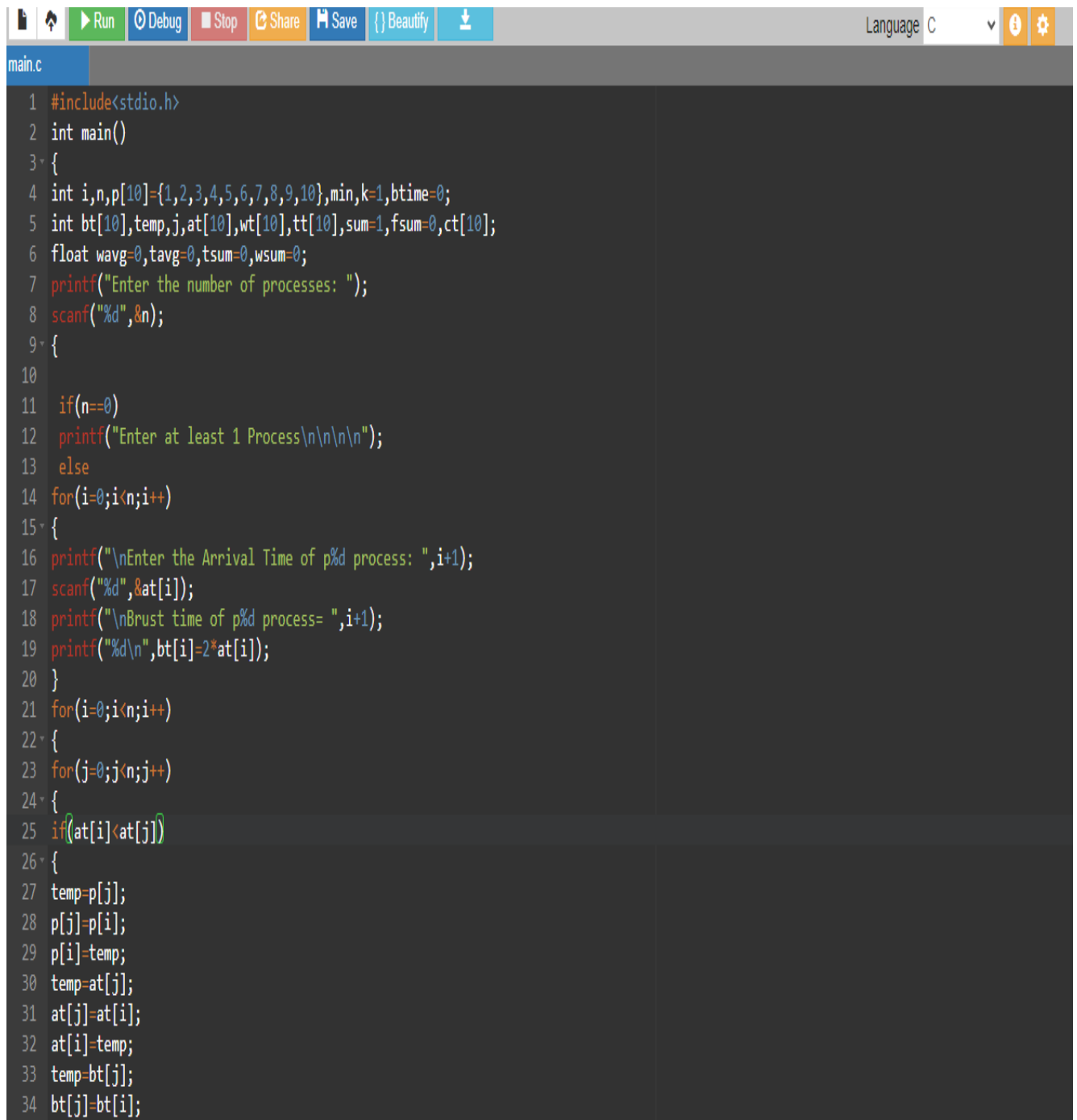
printf("\n\n\n\nAverage Waiting Time: %f",wavg);

printf("\nAverage Turn Around Time: %f",tavg);

return 0;

}}

```










The image shows a screenshot of a C code editor. The editor has a toolbar at the top with buttons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The language is set to C. The code is in a file named 'main.c' and implements a sorting algorithm (likely bubble sort) for an array of processes. The code includes headers, declares variables for process count, arrival times, burst times, waiting times, and turn-around times. It prompts the user to enter the number of processes and then the arrival and burst times for each process. The algorithm sorts the processes based on their arrival times and calculates the waiting and turn-around times for each process.

```

1 #include<stdio.h>
2 int main()
3 {
4     int i,n,p[10]={1,2,3,4,5,6,7,8,9,10},min,k=1,btime=0;
5     int bt[10],temp,j,at[10],wt[10],tt[10],sum=1,fsum=0,ct[10];
6     float wavg=0,tavg=0,tsum=0,wsum=0;
7     printf("Enter the number of processes: ");
8     scanf("%d",&n);
9     {
10
11     if(n==0)
12         printf("Enter at least 1 Process\n\n\n\n");
13     else
14         for(i=0;i<n;i++)
15         {
16             printf("\nEnter the Arrival Time of p%d process: ",i+1);
17             scanf("%d",&at[i]);
18             printf("\nBurst time of p%d process= ",i+1);
19             printf("%d\n",bt[i]=2*at[i]);
20         }
21         for(i=0;i<n;i++)
22         {
23             for(j=0;j<n;j++)
24             {
25                 if(at[i]<at[j])
26                 {
27                     temp=p[j];
28                     p[j]=p[i];
29                     p[i]=temp;
30                     temp=at[j];
31                     at[j]=at[i];
32                     at[i]=temp;
33                     temp=bt[j];
34                     bt[j]=bt[i];

```

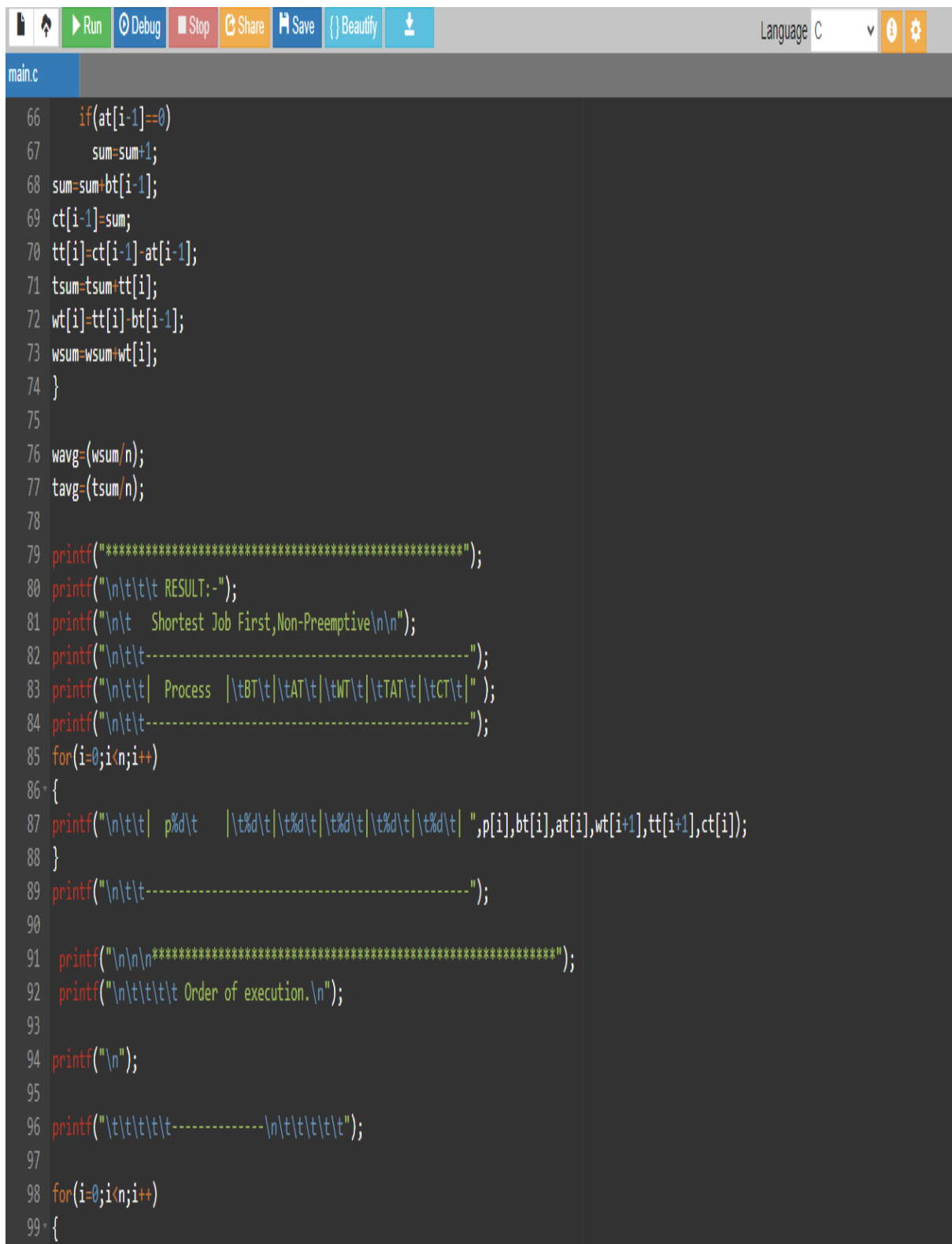


Language C

main.c

```
34 bt[j]=bt[i];
35 bt[i]=temp;
36 }
37 }
38 }
39
40 for(j=0;j<n;j++)
41 {
42 btime=btime+bt[j];
43 min=bt[k];
44 for(i=k;i<n;i++)
45 {
46 if (btime>=at[i] && bt[i]<min)
47 {
48 temp=p[k];
49 p[k]=p[i];
50 p[i]=temp;
51 temp=at[k];
52 at[k]=at[i];
53 at[i]=temp;
54 temp=bt[k];
55 bt[k]=bt[i];
56 bt[i]=temp;
57 }
58 }
59 k++;
60 }
61 wt[0]=0;
62 sum=1;
63 ct[0]=1;
64 for(i=1;i<=n;i++)
65 {
66 if(at[i-1]==0)
67 sum=sum+1;
```






```

98 for(i=0;i<n;i++)
99 {
100 printf("| Process[%d] |\n\t\t\t\t\t",p[i]);
101 }
102 printf("-----\n\t\t\t\t\t");
103 printf("\n\n\nAverage Waiting Time: %f",wavg);
104 printf("\nAverage Turn Around Time: %f",tavg);
105 return 0;
106 }}

```


**OnlineGDB** beta  
online compiler and debugger for c/c++  
code. compile. run. debug. share.  
[IDE](#)  
[My Projects](#)  
[Classroom new](#)  
[Learn Programming](#)  
[Programming Questions](#)  
[Jobs new](#)  
[Sign Up](#)  
[Login](#)  
[Learn Python with KodeKloud](#)

input

```

Enter the number of processes: 5
Enter the Arrival Time of p1 process: 4
Burst time of p1 process= 8
Enter the Arrival Time of p2 process: 9
Burst time of p2 process= 18
Enter the Arrival Time of p3 process: 6
Burst time of p3 process= 12
Enter the Arrival Time of p4 process: 8
Burst time of p4 process= 16
Enter the Arrival Time of p5 process: 3
Burst time of p5 process= 6
*****
RESULT:-
Shortest Job First,Non-Preemptive

| Process | BT | AT | WT | TAT | CT |
|-----|-----|-----|-----|-----|-----|
| p5 | 6 | 3 | -2 | 4 | 7 |
| p1 | 8 | 4 | 3 | 11 | 15 |
| p3 | 12 | 6 | 9 | 21 | 27 |
| p4 | 16 | 8 | 19 | 35 | 43 |
| p2 | 18 | 9 | 34 | 52 | 61 |
|-----|-----|-----|-----|-----|
*****

Order of execution.
-----

```

[About](#) • [FAQ](#) • [Blog](#) • [Terms of Use](#) • [Contact Us](#) • [GDB Tutorial](#) • [Credits](#) • [Privacy](#)

⚡ OnlineGDB beta  
online compiler and debugger for c/c++

code compile run debug share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Jobs new

Sign Up

Login

Learn Python with  
KodeKloud

About • FAQ • Blog • Terms of Use • Contact Us • GDB  
Tutorial • Credits • Privacy  
© 2016 - 2023 GDB Online

input

Order of execution.

-----  
| Process[5] |  
| Process[1] |  
| Process[3] |  
| Process[4] |  
Process[2]

Average Waiting Time: 12.600000  
Average Turn Around Time: 24.600000  
..Program finished with exit code 0  
Press ENTER to exit console.

Enter the number of processes: 3  
Enter the Arrival Time of p1 process: 4  
Burst time of p1 process= 8  
Enter the Arrival Time of p2 process: 9  
Burst time of p2 process= 18  
Enter the Arrival Time of p3 process: 6  
Burst time of p3 process= 12  
\*\*\*\*\*  
RESULT:-  
Shortest Job First,Non-Preemptive

Process	BT	AT	WT	TAT	CT
p1	8	4	-3	5	9
p3	12	6	3	15	21
p2	18	9	12	30	39

\*\*\*\*\*  
Order of execution.

-----  
| Process[1] |  
| Process[3] |  
Process[2]

Average Waiting Time: 4.000000  
Average Turn Around Time: 16.666666