Student Name: Saransh Dubey
Student ID: 11803436
Email Address : saransh251999@gmail.com
GitHub Link:

Problem:

Ques15. A uniprocessor system has n number of CPU intensive processes, each process has its own requirement of CPU burst. The process with lowest CPU burst is given the highest priority. A late-arriving higher priority process can preempt a currently running process with lower priority. Simulate a scheduler that is scheduling the processes in such a way that higher priority process is never starved due to the execution of lower priority process. What should be its average waiting time and average turnaround time if no two processes are arriving are arriving at same time.

Ans: In this problem , we have uniprocessor with n number of CPU intensive process and each one among them has its own cpu burst , highest property has to give to lowest burst for this purpose we have simulate a scheduler in the way given in the question, for doing this we have to follow the following steps written in the algorithm.

Algorithm:

Creating a function to sort_processes() the process according to the arrival time
1. Read the no of processes as array
2. Read arrival time.
3. Initialize flag = 0
4. for all processes in array

```
5. if (arrival time of process i > arrival time of process i+1)
6.       swap (process[i],process[i+1])
7.           end if then
8.       end for
```

Function for execution of process

```
1. Read processes as array
2. Read arrival time, burst time, completion time
3. sort the processes according to sort_processes() on behalf of arival time
4. for all element in process array
5. if (arrival time [i] <= arrival_time[0] && burst_time[i] <
burst_time[last_element] &7 flag != 1)
6.    calculate turn around time = completion time - arrival time
                        waiting time = turn around time - burst time
7.    repeat step 4,5,6 for all processes    present in process_array
8. calculate average turn around time = sum of turn around time of all process /
no. of processes
                   average waiting time = sum of waiting time of all processes / no. of
processes
9. display Process id, arrival time, completion time, turn around time, waiting
time, average turn around time, average waiting time.
10.end
```

Description:

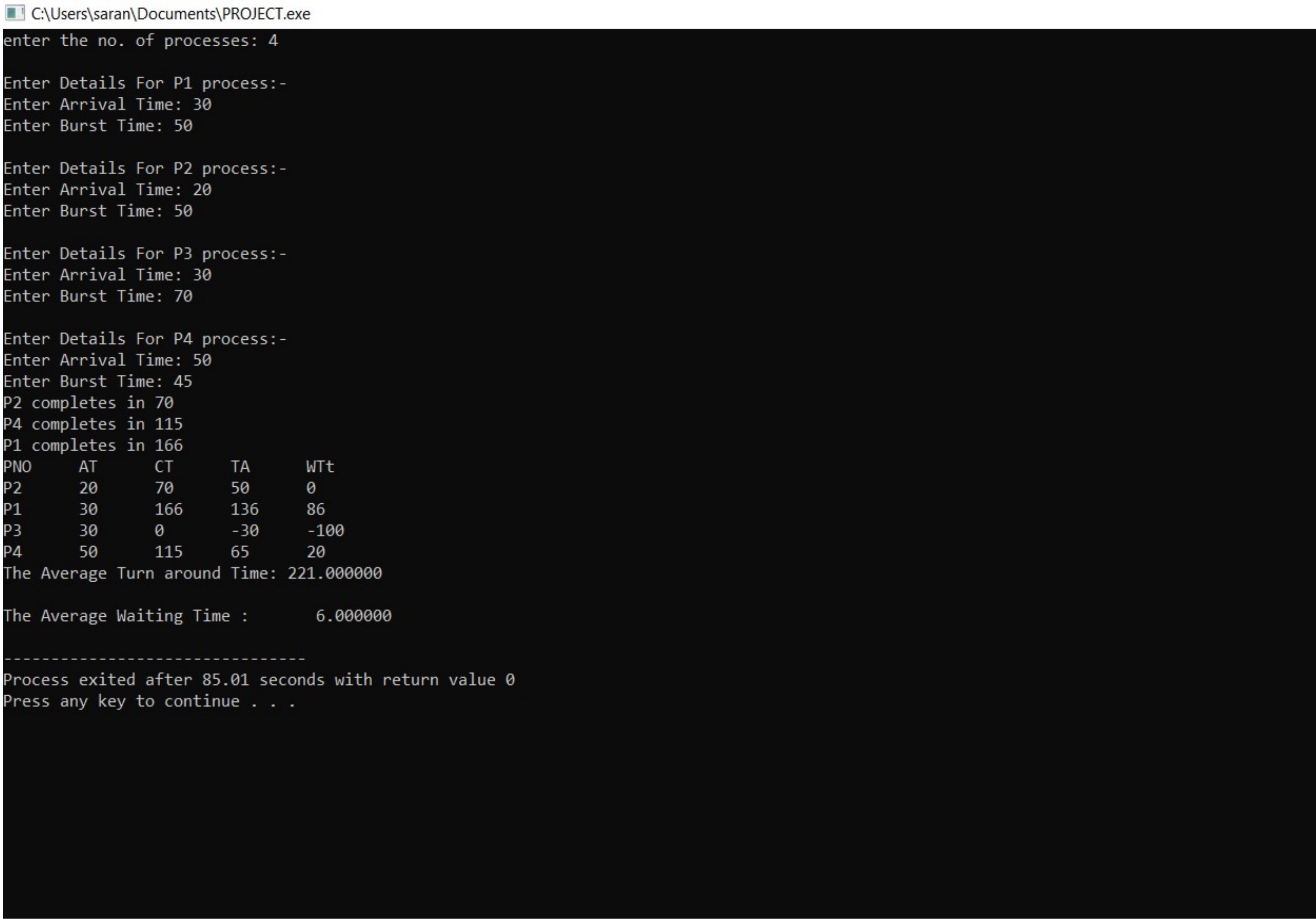
To implement the above problem we have to make functions for sort processess and exection process in order to fulfil the statements given in the question for that required code in C programming language is written below.

Code:

```
#include<stdio.h>
int n;
struct process
{

int p_no;

int arrival_t,burst_t,ct,wait_t,taround_time,p;

int flag;
  }p_list[100];
  void Sorting()
  {
struct process p;

int i, j;

for(i=0;i<n-1;i++)

{

for(j=i+1;j<n;j++)

{

if(p_list[i].arrival_t > p_list[j].arrival_t)

{

p = p_list[i];

p_list[i] = p_list[j];
p_list[j] = p;

}

}

}
  }
  int main()
  {

int i,t=0,b_t=0,peak;

int a[10];

float wait_time = 0, taround_time = 0, avg_w_t=0, avg_taround_time=0;

printf("enter the no. of processes: ");

scanf("%d",&n);

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

{

p_list[i].p_no = i+1;

printf("\nEnter Details For P%d process:-\n", p_list[i].p_no);
printf("Enter Arrival Time: ");
scanf("%d", &p_list[i].arrival_t );
printf("Enter Burst Time: ");
scanf("%d", &p_list[i].burst_t);
p_list[i].flag = 0;
b_t = b_t + p_list[i].burst_t;
}
Sorting();
for(int i=0;i<n;i++)
{
a[i]=p_list[i].burst_t;
}
p_list[9].burst_t = 9999;
```

```
for(t = p_list[0].arrival_t; t <= b_t+1;)
{
peak = 9;
for(i=0;i<n;i++)
{
if(p_list[i].arrival_t <= t && p_list[i].burst_t < p_list[peak].burst_t && p_list[i].flag != 1)
{
peak = i;
}
if(p_list[peak].burst_t==0 && p_list[i].flag != 1)
{
p_list[i].flag = 1;
p_list[peak].ct=t;p_list[peak].burst_t=9999;
printf("P%d completes in %d\n",p_list[i].p_no,p_list[peak].ct);
}
}
t++;
(p_list[peak].burst_t)--;
}
for(i=0;i<n;i++)
{
p_list[i].taround_time=(p_list[i].ct)-(p_list[i].arrival_t);
avg_taround_time=avg_taround_time+p_list[i].taround_time;
p_list[i].wait_t=((p_list[i].taround_time)-a[i]);
avg_w_t=avg_w_t+p_list[i].wait_t;
}
printf("PNO\tAT\tCT\tTA\tWTt\n");
for(i=0;i<n;i++)
{
printf("P%d\t%d\t%d\t%d\t%d\n",p_list[i].p_no,p_list[i].arrival_t,p_list[i].ct,p_list[i].ta
round_time
,p_list[i].wait_t);
  }
printf("The Average Turn around Time: %f\t\n\n",avg_taround_time);
printf("The Average Waiting Time :\t %f\t\n",avg_w_t);
}
```

OUTPUT:



```
enter the no. of processes: 4

Enter Details For P1 process:-
Enter Arrival Time: 30
Enter Burst Time: 50

Enter Details For P2 process:-
Enter Arrival Time: 20
Enter Burst Time: 50

Enter Details For P3 process:-
Enter Arrival Time: 30
Enter Burst Time: 70

Enter Details For P4 process:-
Enter Arrival Time: 50
Enter Burst Time: 45
P2 completes in 70
P4 completes in 115
P1 completes in 166
PNO     AT      CT      TA      WTt
P2      20      70      50      0
P1      30      166     136     86
P3      30      0       -30     -100
P4      50      115     65      20
The Average Turn around Time: 221.000000

The Average Waiting Time :       6.000000

--------------------------------
Process exited after 85.01 seconds with return value 0
Press any key to continue . . .
```