Name → Shivani Negi
Student I'd → 20052026
Course → BSC (It)
Section → 'A'

Write a C Program to implement SRTN algorithom.

Q.1.)

Description :- Preemptive scheduling algorithm is an algorithm in which the processor is allocated to the job having minimum CPU burst time, but the job can preempted (replaced) by a newer job with shorter burst time.

Algorithm

```
STEP1 : START

STEP2 : Declare arrival-time, burst-time, i, smallest,
        count, end, time, limit, waiting-time, turnaround-
        time, average-waiting-time, average-turnaround-time

STEP3 : Read limit, arrival time, burst time

STEP4 : temp[i] = burst_time[i]

STEP5 : burst_time = 9999

STEP6 : Using a for loop for(time=0; count!=limit; time++)
                    smallest = 9
 Again using a loop for (i=0; i<limit; i++) check if(
     arrival_time[i] <= time && burst_time[i] < burst_time
     [smallest] && burst_time[i]>0)
   Then smallest = i
```

burst_time[smallest]--;
Check if (burst_time[smallest]==0) then count++
end = time+1

wait_time = wait_time + end - arrival_time[smallest]-
turnaround_time = turnaround_time + end -
arrival_time[smallest]

STEP7 : average-waiting_time = wait_time / limit
Average-turnaround_time = turnaround_time / limit

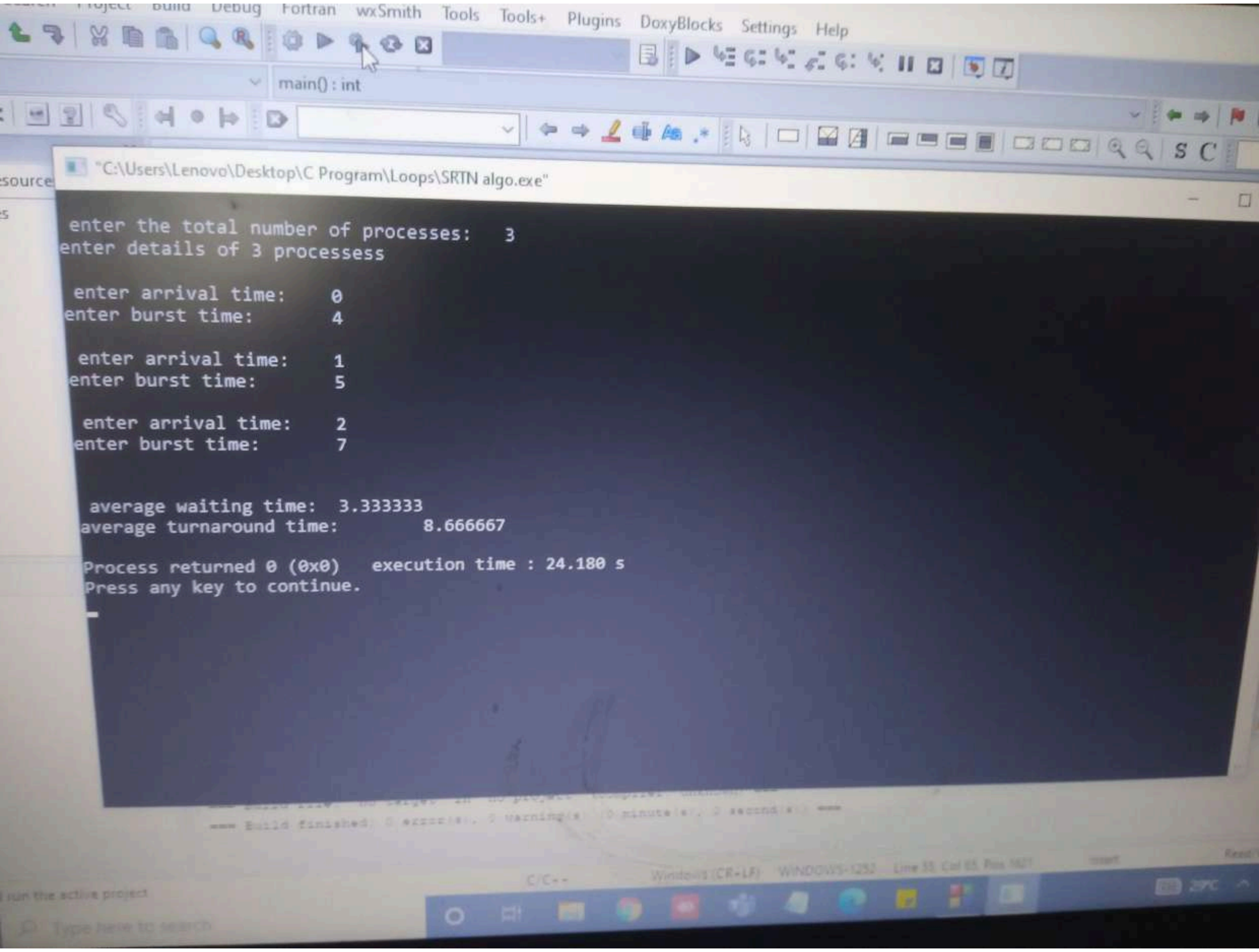
TEP8: Print average-waiting-time and average-turnaroun
_time.

TEP9 : STOP

SOURCE CODE

```
int main () {
 int arrival_time[10], burst_time[10], temp[10];
  int i, smallest, count =0, time, limit;
   double wait_time =0, turnaround_time=0, end;
   float average_waiting_time, average_turnaround_time;
 printf("\n Enter the total number of processes : \t");
 scanf("%d", &limit);
  printf("Enter detailes of %d processes\n", limit);
  for(i=0; i<limit; i++)
   {
     printf("\n Enter arrival time: \t");
    scanf("%d", &arrival_time[i]);
    printf("Enter burst time: \t");
    scanf("%d", &burst_time[i]);
    temp[i]= burst_time[i];
   }
   burst_time[9] = 9999;
  for (time=0 ; count != limit ; time++)
   {   smallest = 9;
    for (i=0 ; i<limit ; i++)
     {
```

```
if(arrival_time[i] <= time && burst_time[i] <
   burst_time[smallest] && burst_time > 0)
 {
   smallest = i;
 }
}
burst_time[smallest]--;
if(burst_time[smallest]==0)
 {
     Count++;
   end = time+1;
 wait_time = wait_time + end - arrival_time[smallest] -
  temp[smallest];
 turnaround_time = turnaround_time + end -
  arrival_time[smallest];
 }
}
average_waiting_time = wait_time / limit;
average_turnaround_time = turnaround_time/limit;
printf("\n\n Average waiting time : \t%lf\n",
        average_waiting_time);
printf("Average Turnaround Time : \t%lf\n",
           average_turnaround_time);
 return 0;
}
```



"C:\Users\Lenovo\Desktop\C Program\Loops\SRTN algo.exe"

```
 enter the total number of processes:   3
enter details of 3 processessss

 enter arrival time:     0
enter burst time:        4

 enter arrival time:     1
enter burst time:        5

 enter arrival time:     2
enter burst time:        7


 average waiting time:  3.333333
average turnaround time:        8.666667

Process returned 0 (0x0)   execution time : 24.180 s
Press any key to continue.
```