1. Explain the problem in terms of operating system concept?

The problem can be considered to be a simple case of pre-emptive shortest remaining process time first (SRTF) which  is the preemptive version of Shortest Job First (SJF) , SRTF algorithm is faster in execution than SJN algorithm, As the problem states we have to take inputs for processors arrival time and burst time to calculate average turn around and average wait time so specific logic is implemented to solve this which is mentioned in code snippet in this file

1. Write the algorithm for proposed solution of the assigned problem.

This Algorithm is the preemptive version of SJF scheduling. In SRTF, the execution of the process can be stopped after certain amount of time. At the arrival of every process, the short term scheduler schedules the process with the least remaining burst time among the list of available processes and the running process. Once all the processes are available in the ready queue, No preemption will be done and the algorithm will work as SJF scheduling. The context of the process is saved in the Process Control Block when the process is removed from the execution and the next process is scheduled. This PCB is accessed on the next execution of this process.

1. Calculate complexity of implemented algorithm.

The complexity of the program in worst case can be O(n+n2) which will eventually lead to O(n2) and in the best case and average case also it will remain the same, this algorithm can be used to find average turn around and average waiting time by processors where user can enter the information about total number of processors their arrival time and burst time to find out the average turnaround and average waiting time with accuracy as well as precision

1. Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.

#include<iostream>

#include <iomanip>

using namespace std;

int main()

{

int i, smallest, count = 0, time, limit;

double wait\_time = 0, turnaround\_time = 0, end;

int at[20], bt[20], temp[20];

double average\_waiting\_time, average\_turnaround\_time;

cout<<"\nPlease Enter The Total Number Of Processes:\t";

cin>>limit;

cout<<"\nPlease Enter Details Of "<<limit<<" Processes\n";

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

{

cout<<"\nPlease Enter Arrival Time:\t";

cin>>at[i];

cout<<"Please Enter Burst Time:\t";

cin>>bt[i];

temp[i] = bt[i];

}

bt[9] = 100000;

for(time = 0; count != limit; time++)

{

smallest = 9;

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

{

if(at[i] <= time && bt[i] < bt[smallest] && bt[i] > 0)

{

smallest = i;

}

}

bt[smallest]--;

if(bt[smallest] == 0)

{

count++;

end = time + 1;

wait\_time = wait\_time + end - at[smallest] - temp[smallest];

turnaround\_time = turnaround\_time + end - at[smallest];

}

}

average\_waiting\_time = wait\_time / limit;

average\_turnaround\_time = turnaround\_time / limit;

cout<<"\nAs Per Your Entered Values Average Waiting Time:\t "<<fixed<<average\_waiting\_time;

cout<<"\nAs Per Your Entered Values Average Turnaround Time:\t"<<fixed<<average\_turnaround\_time;

}

The code is written in C++ so very first constraint is that user should must have a gcc/gnu or any C++ compiler of his own choice with whichever user is comfortable

5 .If you have implemented any additional algorithm to support the solution, explain the need and usage of the same.

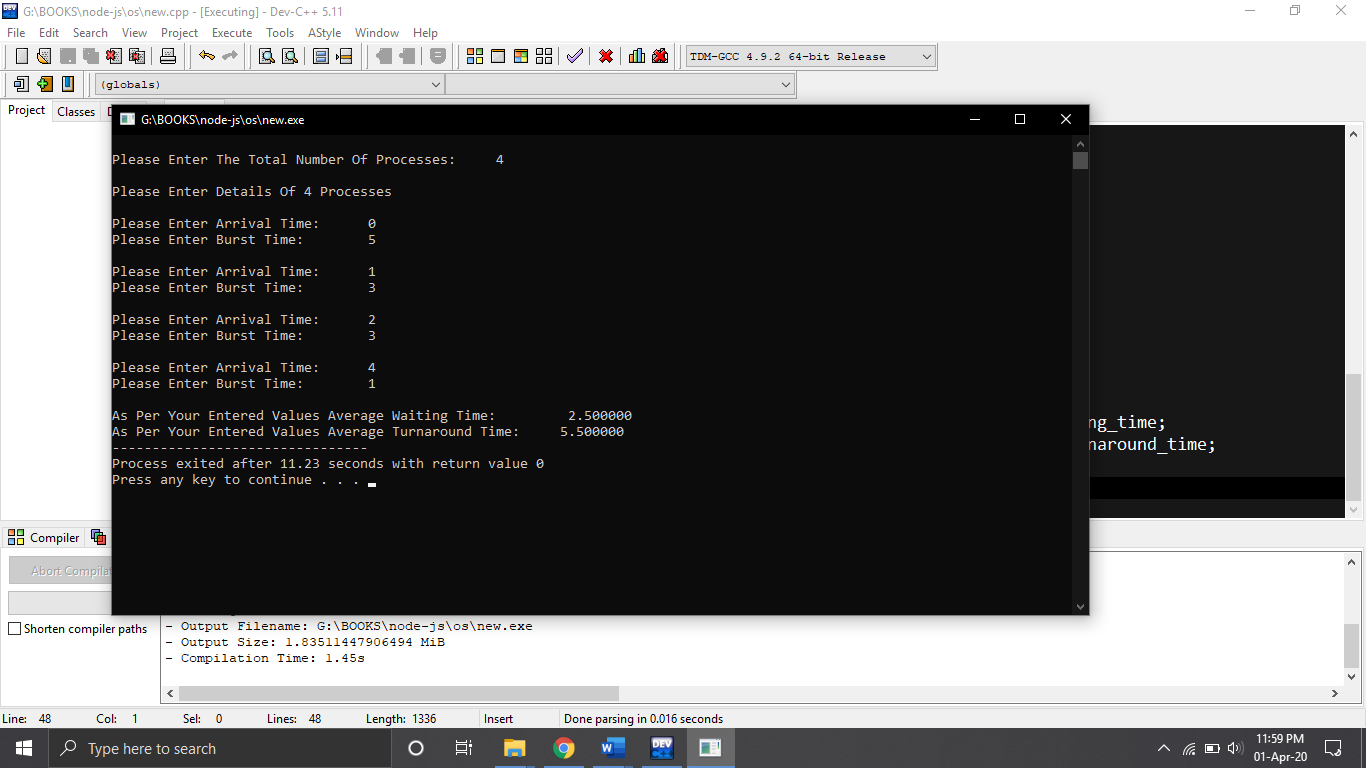
As this question is a simple case of Shortest response time first so no additional algorithm is used to implement its solution

6. Explain the boundary conditions of the implemented code.

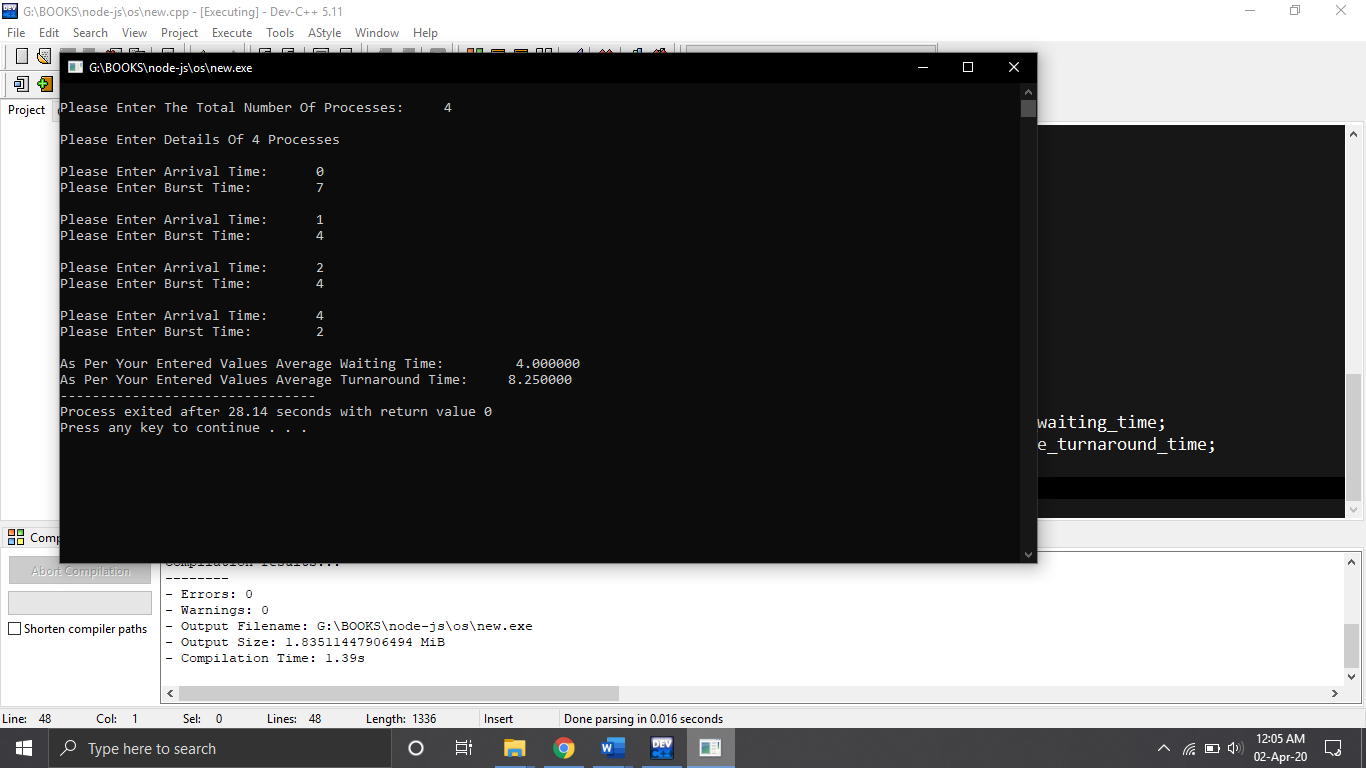
As per the current solution code, one boundary condition is that it will work up to 20 processors only but in case user want to enter the values of more than processor a simple edit it in array size from 20 to n in three arrays which are at, bt and temp. So this is not a really big issue. And initial array size is set to twenty because usually user will not even exceed more than five processors so to reduce runtime, the initial value is set to twenty

7.Explain all the test cases applied on the solution of assigned problem.

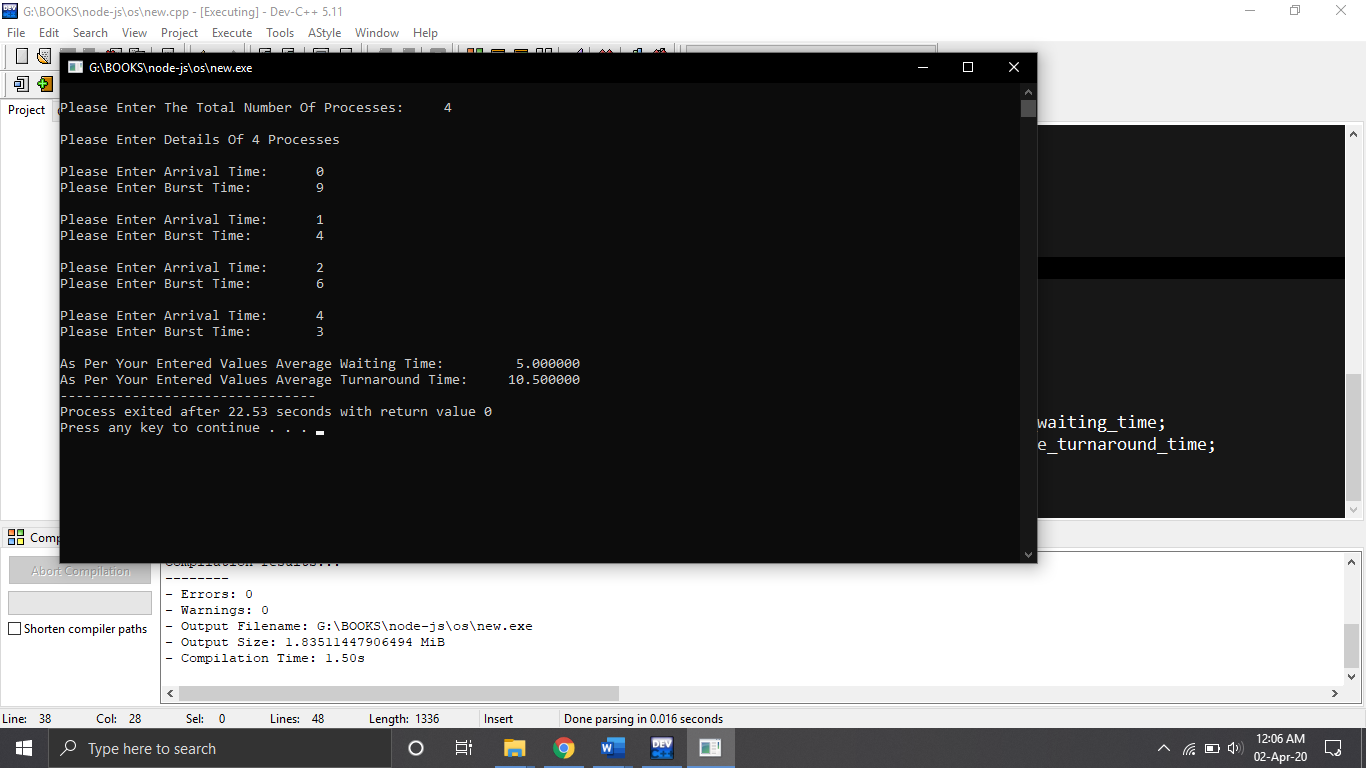
First test case is same as given in the question and below is its screenshot



Second test case



Third test case



**8.Have you made minimum 5 revisions of solution on GitHub?**

**GitHub Link:** <https://github.com/harjotscs/os>

Yes