SHEET 3

Name :Gehad Mohammad Sabry Ragab

G5

Choose at least three algorithms and write the Pseudo Code of them and also implement them with any programming language.

Question 1:

1- FCFS

the Pseudo Code of FCFS scheduling

Step1: Create number of processes, get the number of process from user.

Step2: Get the ID and Brust time of each process from user.

Step3: Initially, set the waiting time of first process as 0 and set Total time for the first process equal to its brust time.

Step4: Calculate the Total time and waiting time for the remaining processes.

Step5: Waiting time of one process is the total brust time of all the previous processes.

Step6: Total time of process is calculated by adding its Waiting time and its brust time.

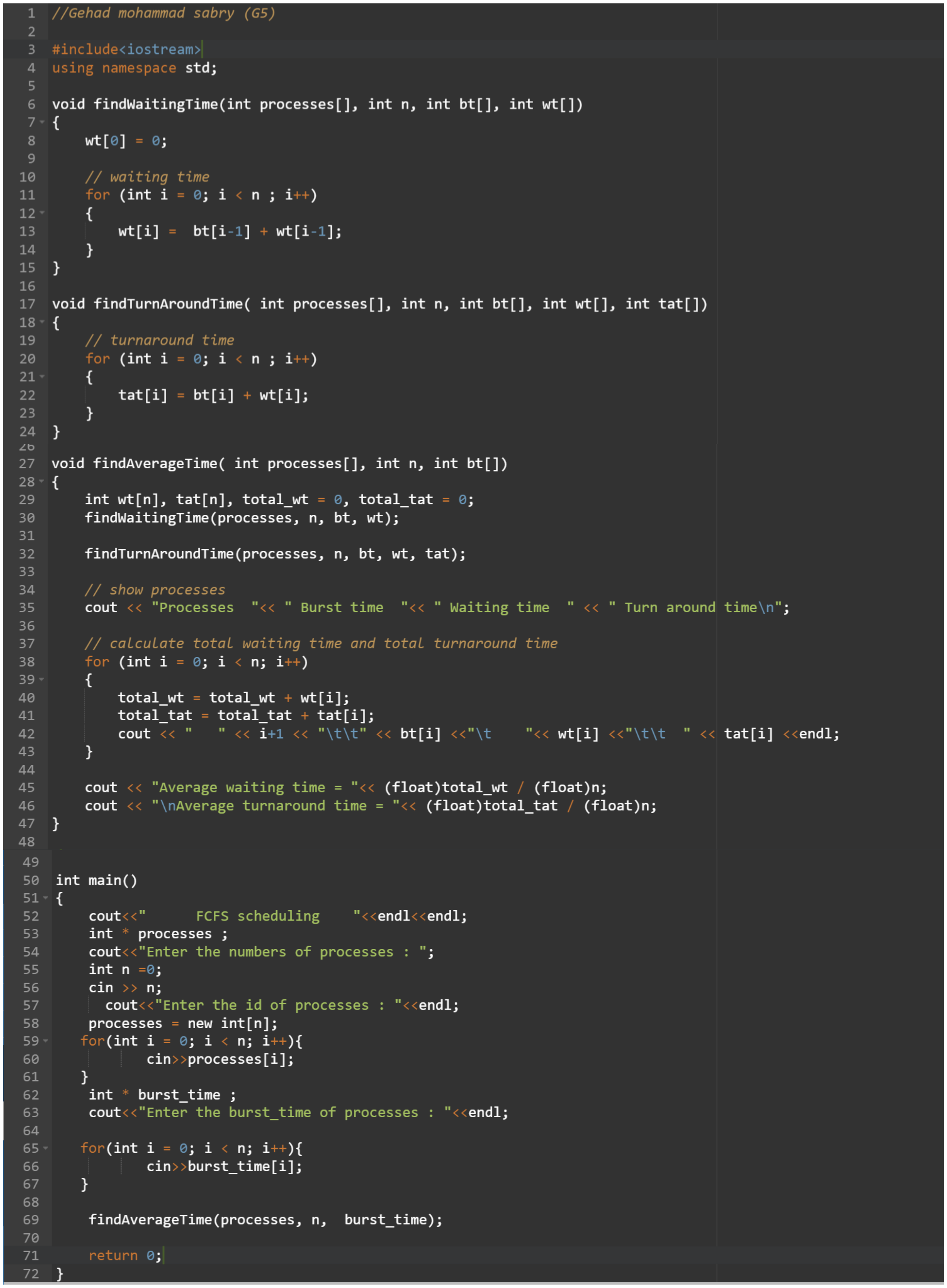
Step7: TotalWaiting Time of all processes is calculated by adding the waiting time for all process.

Step8: Total TurnAround time is calculated by adding total time of all processes.

Step9: Now Calculate AverageWaiting Time by dividing the total waiting time by total number of process and display.

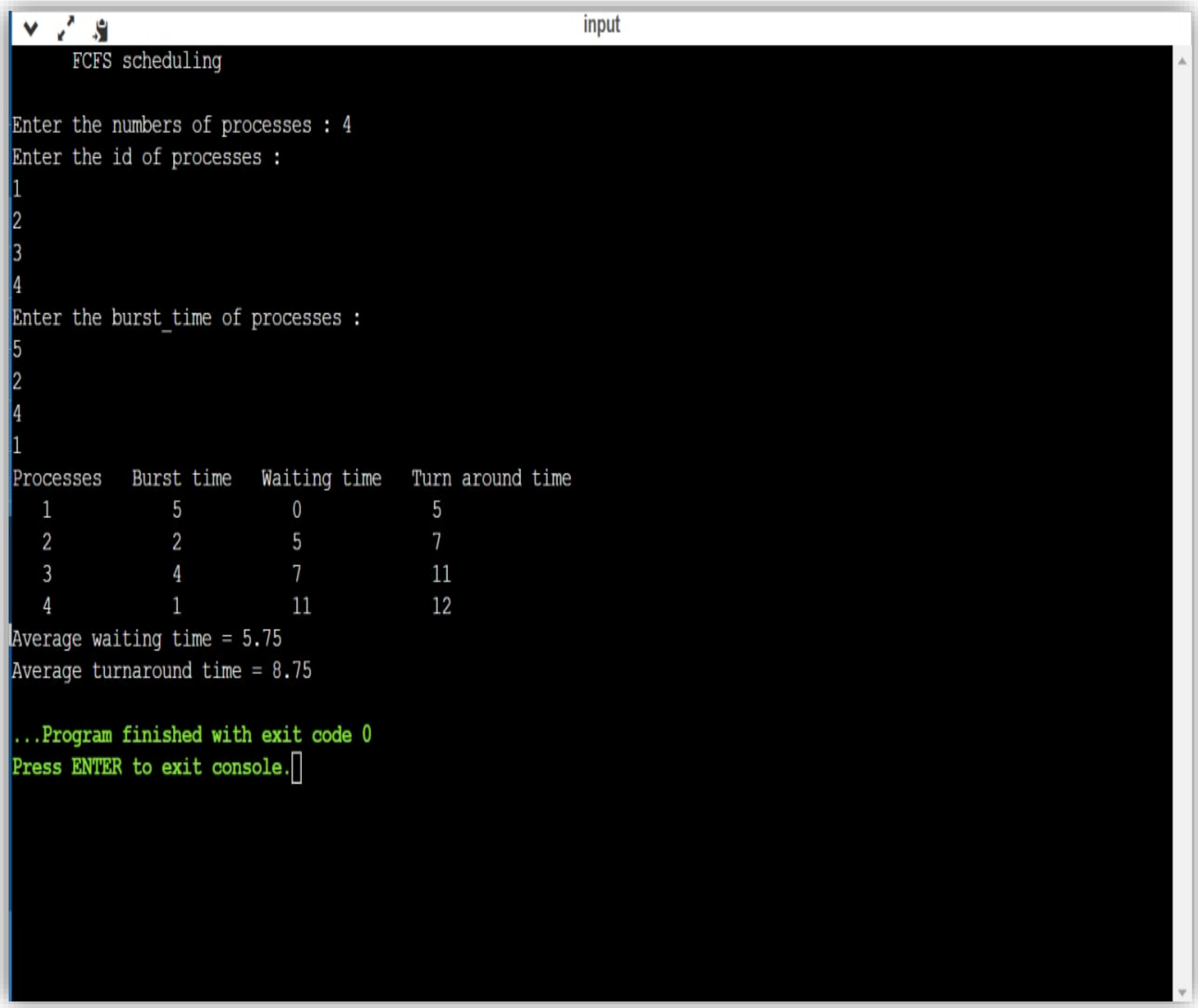
Step10: And Calculate Average TurnAroundTime by dividing the total time by the number of process and display.

Step11: Display the result



```cpp
//Gehad mohammad sabry (G5)

#include<iostream>
using namespace std;

void findWaitingTime(int processes[], int n, int bt[], int wt[])
{
    wt[0] = 0;

    // waiting time
    for (int i = 0; i < n ; i++)
    {
        wt[i] =  bt[i-1] + wt[i-1];
    }
}

void findTurnAroundTime( int processes[], int n, int bt[], int wt[], int tat[])
{
    // turnaround time
    for (int i = 0; i < n ; i++)
    {
        tat[i] = bt[i] + wt[i];
    }
}

void findAverageTime( int processes[], int n, int bt[])
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;
    findWaitingTime(processes, n, bt, wt);

    findTurnAroundTime(processes, n, bt, wt, tat);

    // show processes
    cout << "Processes  "<< " Burst time  "<< " Waiting time  " << " Turn around time\n";

    // calculate total waiting time and total turnaround time
    for (int i = 0; i < n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        cout << "   " << i+1 << "\t\t" << bt[i] <<"\t    "<< wt[i] <<"\t\t  " << tat[i] <<endl;
    }

    cout << "Average waiting time = "<< (float)total_wt / (float)n;
    cout << "\nAverage turnaround time = "<< (float)total_tat / (float)n;
}


int main()
{
    cout<<"      FCFS scheduling    "<<endl<<endl;
    int * processes ;
    cout<<"Enter the numbers of processes : ";
    int n =0;
    cin >> n;
      cout<<"Enter the id of processes : "<<endl;
    processes = new int[n];
   for(int i = 0; i < n; i++){
           cin>>processes[i];
   }
    int * burst_time ;
    cout<<"Enter the burst_time of processes : "<<endl;

   for(int i = 0; i < n; i++){
           cin>>burst_time[i];
   }

    findAverageTime(processes, n,  burst_time);

    return 0;
}
```

Output:



```
      FCFS scheduling

Enter the numbers of processes : 4
Enter the id of processes :
1
2
3
4
Enter the burst_time of processes :
5
2
4
1
Processes   Burst time   Waiting time   Turn around time
   1           5            0              5
   2           2            5              7
   3           4            7              11
   4           1            11             12
Average waiting time = 5.75
Average turnaround time = 8.75

...Program finished with exit code 0
Press ENTER to exit console.
```

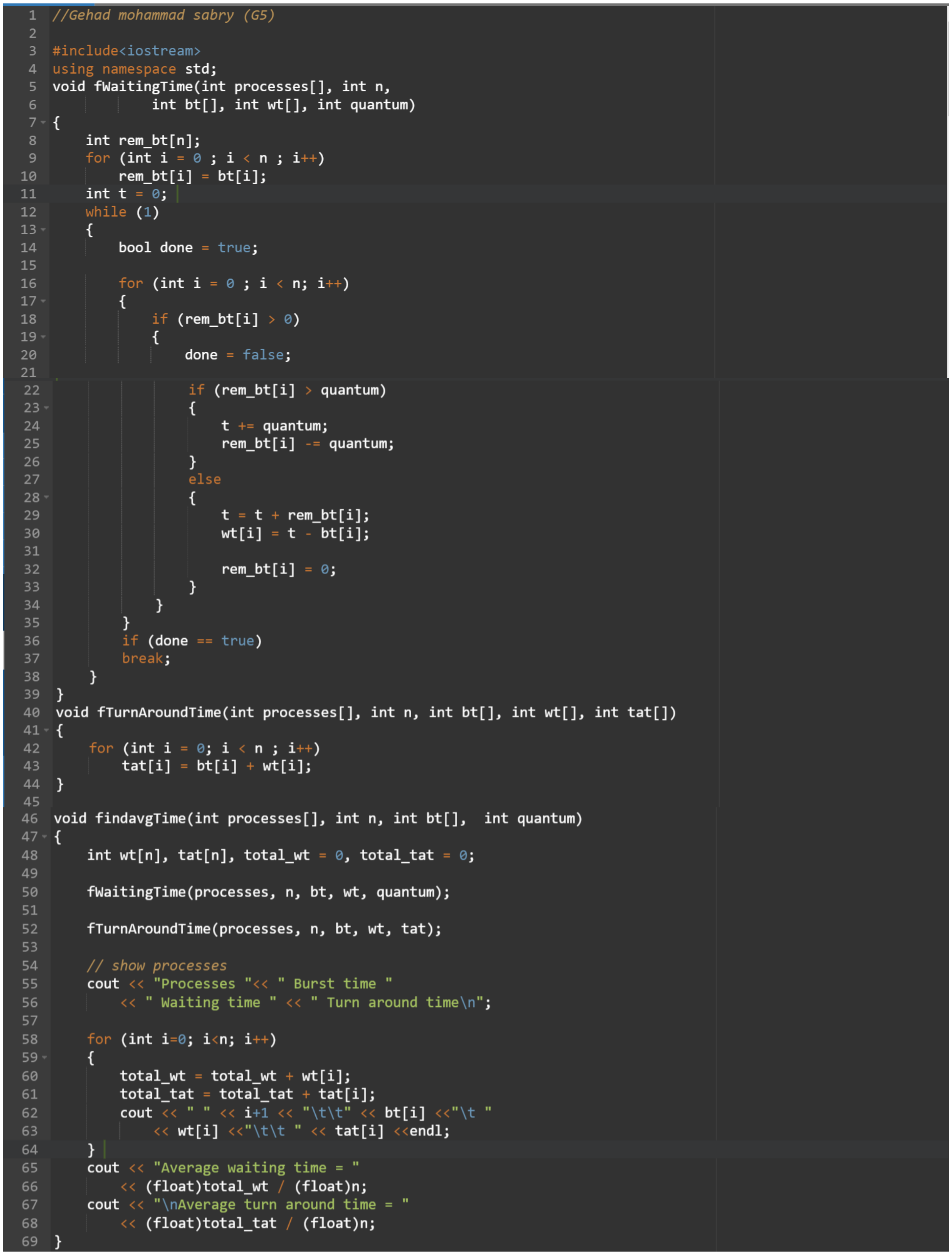
Question 2:

1- Round Robin (preemptive)

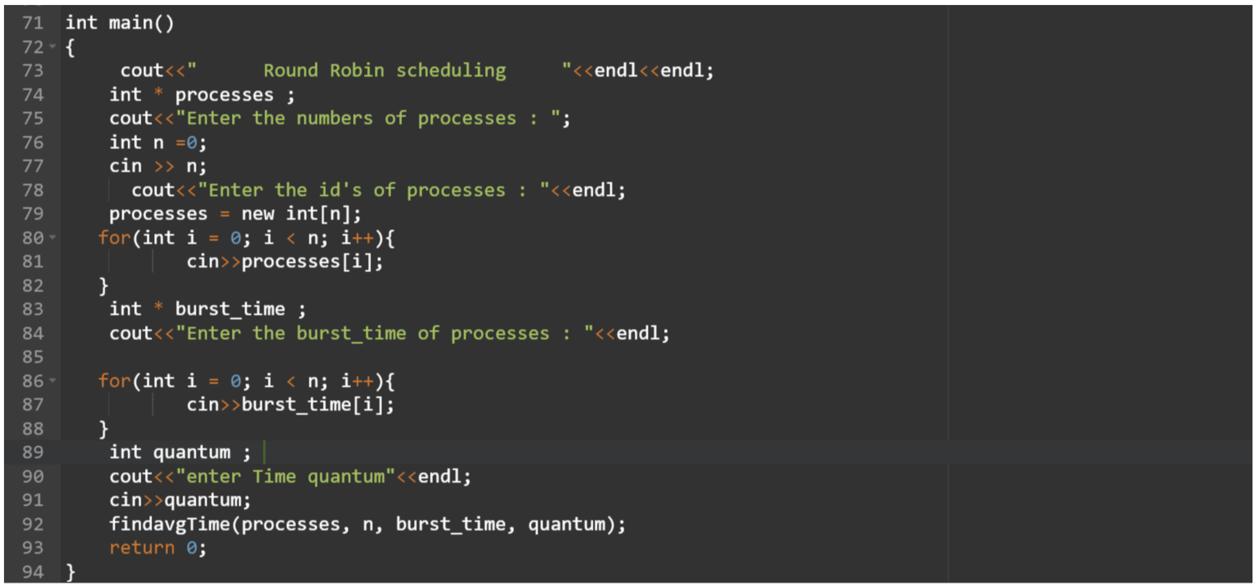
the Pseudo Code of Round Robin scheduling

Step1: **Take the process which occurs first and start executing the process(for quantum time only).**
Step2: **After the quantum time has passed, check for any processes in the Ready queue. If the ready queue is empty then continue the current process. If the queue not empty and the current process is not complete, then add the current process to the end of the ready queue.**
Step3: **If the process is complete and the ready queue is empty then the task is complete**
Step4: **After all these we get the three times which are:**

Step5: **Waiting Time: total time waiting for their complete execution. (turn around time – burst time).**
Step6: **Turn Around Time: total time the process exists in the system. (completion time – arrival time).**
Step7: **Display the result**

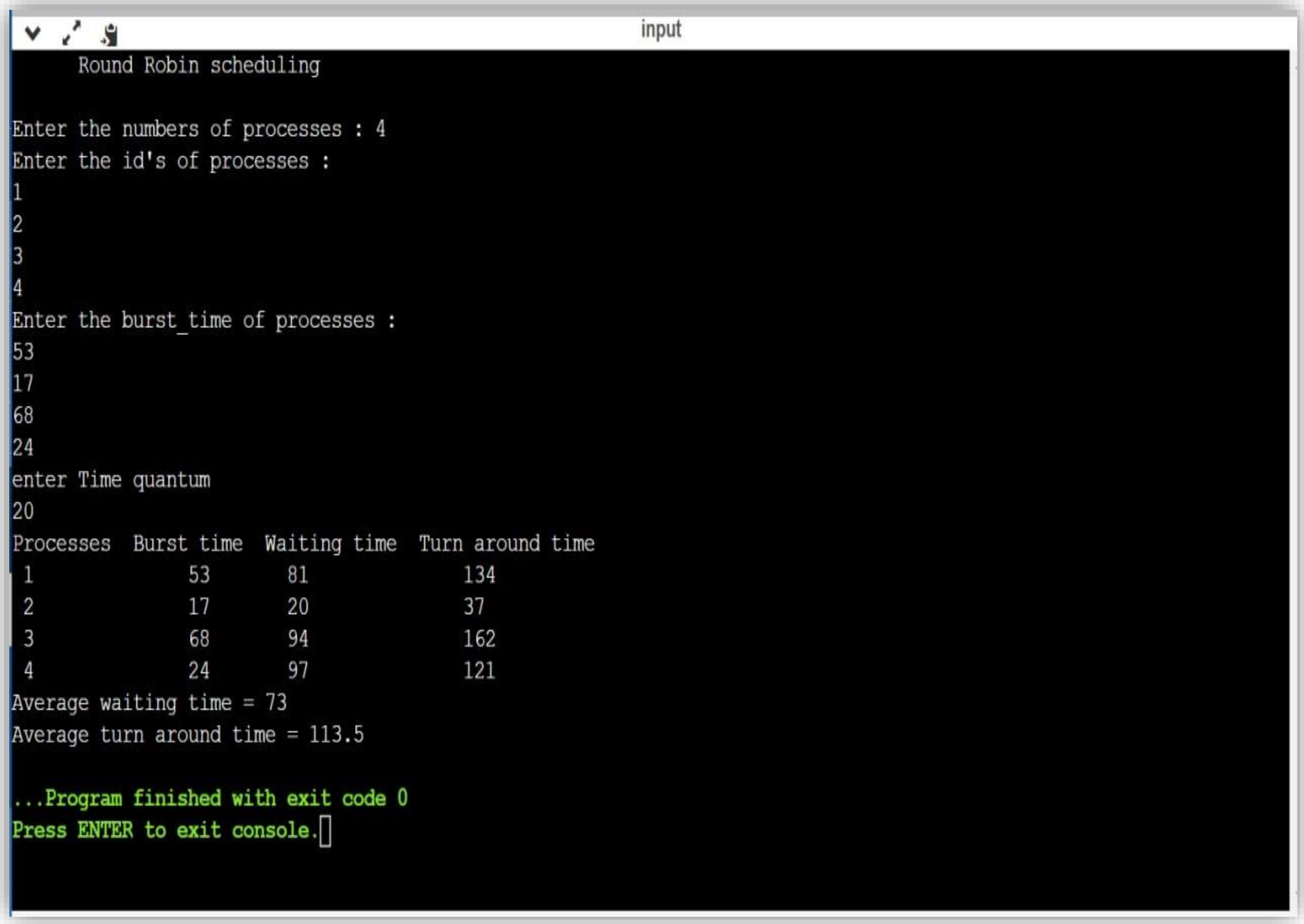


```cpp
//Gehad mohammad sabry (G5)

#include<iostream>
using namespace std;
void fWaitingTime(int processes[], int n,
            int bt[], int wt[], int quantum)
{
    int rem_bt[n];
    for (int i = 0 ; i < n ; i++)
        rem_bt[i] = bt[i];
    int t = 0;
    while (1)
    {
        bool done = true;

        for (int i = 0 ; i < n; i++)
        {
            if (rem_bt[i] > 0)
            {
                done = false;

                if (rem_bt[i] > quantum)
                {
                    t += quantum;
                    rem_bt[i] -= quantum;
                }
                else
                {
                    t = t + rem_bt[i];
                    wt[i] = t - bt[i];

                    rem_bt[i] = 0;
                }
            }
        }
        if (done == true)
        break;
    }
}
void fTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
{
    for (int i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}

void findavgTime(int processes[], int n, int bt[],  int quantum)
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    fWaitingTime(processes, n, bt, wt, quantum);

    fTurnAroundTime(processes, n, bt, wt, tat);

    // show processes
    cout << "Processes "<< " Burst time "
        << " Waiting time " << " Turn around time\n";

    for (int i=0; i<n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        cout << " " << i+1 << "\t\t" << bt[i] <<"\t "
            << wt[i] <<"\t\t " << tat[i] <<endl;
    }
    cout << "Average waiting time = "
        << (float)total_wt / (float)n;
    cout << "\nAverage turn around time = "
        << (float)total_tat / (float)n;
}
```



```cpp
int main()
{
     cout<<"       Round Robin scheduling      "<<endl<<endl;
    int * processes ;
    cout<<"Enter the numbers of processes : ";
    int n =0;
    cin >> n;
      cout<<"Enter the id's of processes : "<<endl;
    processes = new int[n];
   for(int i = 0; i < n; i++){
           cin>>processes[i];
   }
    int * burst_time ;
    cout<<"Enter the burst_time of processes : "<<endl;

   for(int i = 0; i < n; i++){
           cin>>burst_time[i];
   }
    int quantum ;
    cout<<"enter Time quantum"<<endl;
    cin>>quantum;
    findavgTime(processes, n, burst_time, quantum);
    return 0;
}
```

Output:



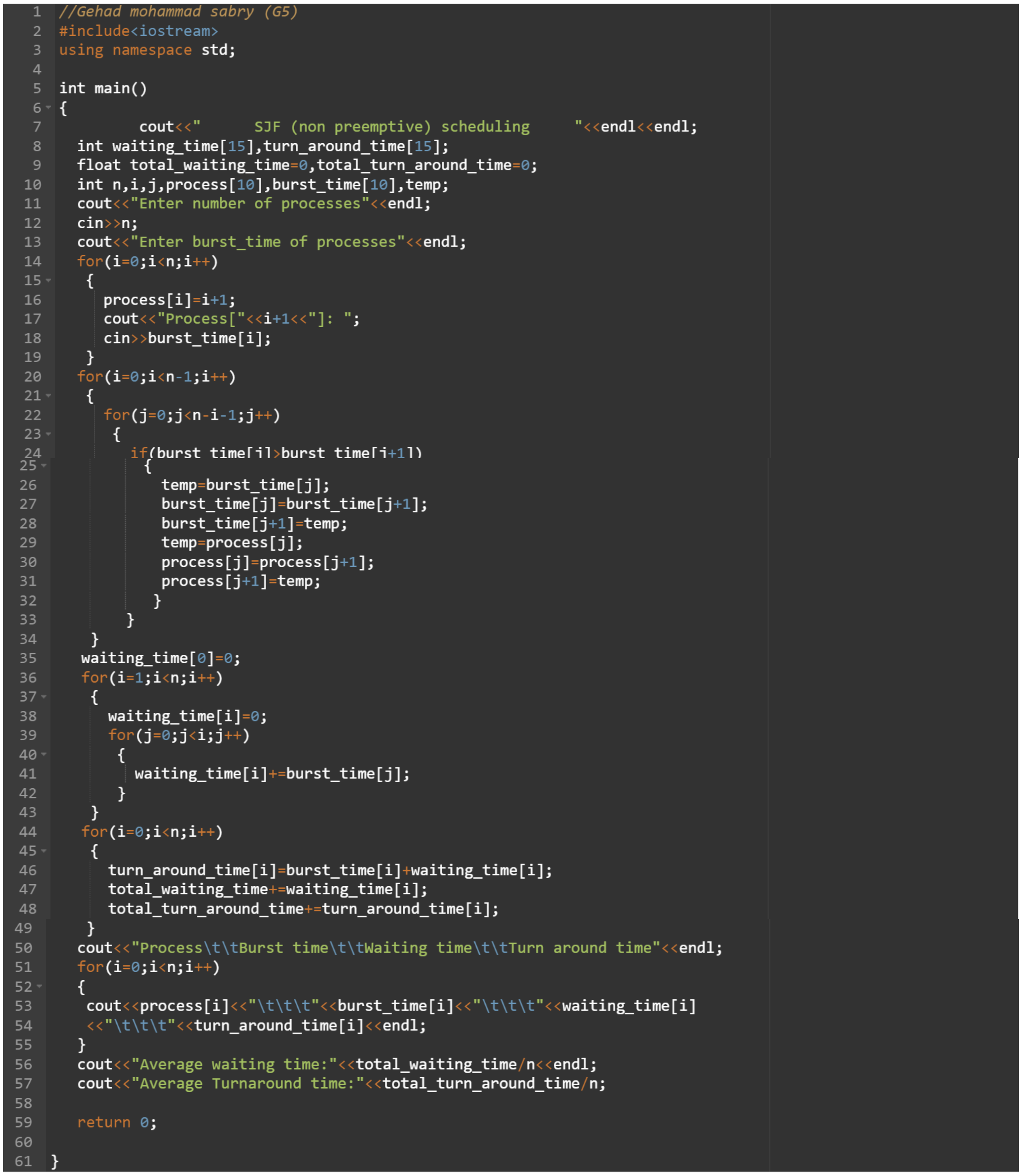
```
      Round Robin scheduling

Enter the numbers of processes : 4
Enter the id's of processes :
1
2
3
4
Enter the burst_time of processes :
53
17
68
24
enter Time quantum
20
Processes  Burst time  Waiting time  Turn around time
 1          53       81           134
 2          17       20           37
 3          68       94           162
 4          24       97           121
Average waiting time = 73
Average turn around time = 113.5

...Program finished with exit code 0
Press ENTER to exit console.
```

Question 3:

1- SJF (non preemptive)

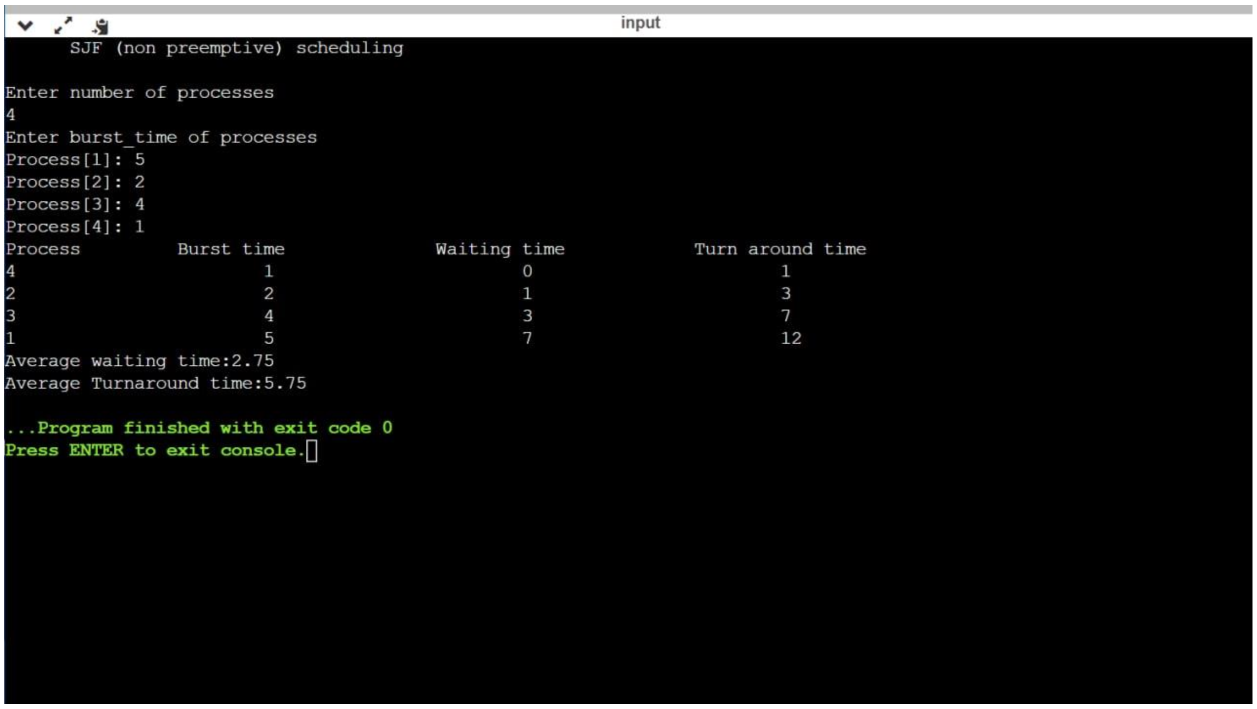
the Pseudo Code of SJF (non preemptive) scheduling

```
Step1: Firstly Start process.
Step2: Declare array size
Step3: Take number of elements to be inserted.
Step4: Select process which have shortest burst time
Among all process will execute first.
Step5: If process have same burst time length then FCFS (
First come First Serve ) scheduling algorithm used.
Step6: Make average waiting time length of next process.
Step7: Start with first process, selection as above and other
processes are to be in queue.
Step8: Calculates Burst total number of time.
Step9: Display the result
Step10: Now Close / Stop process.
```



```cpp
//Gehad mohammad sabry (G5)
#include<iostream>
using namespace std;

int main()
{
         cout<<"      SJF (non preemptive) scheduling     "<<endl<<endl;
  int waiting_time[15],turn_around_time[15];
  float total_waiting_time=0,total_turn_around_time=0;
  int n,i,j,process[10],burst_time[10],temp;
  cout<<"Enter number of processes"<<endl;
  cin>>n;
  cout<<"Enter burst_time of processes"<<endl;
  for(i=0;i<n;i++)
   {
     process[i]=i+1;
     cout<<"Process["<<i+1<<"]: ";
     cin>>burst_time[i];
   }
  for(i=0;i<n-1;i++)
   {
     for(j=0;j<n-i-1;j++)
      {
        if(burst_time[j]>burst_time[j+1])
         {
           temp=burst_time[j];
           burst_time[j]=burst_time[j+1];
           burst_time[j+1]=temp;
           temp=process[j];
           process[j]=process[j+1];
           process[j+1]=temp;
          }
      }
   }
   waiting_time[0]=0;
   for(i=1;i<n;i++)
    {
      waiting_time[i]=0;
      for(j=0;j<i;j++)
       {
         waiting_time[i]+=burst_time[j];
       }
    }
   for(i=0;i<n;i++)
    {
      turn_around_time[i]=burst_time[i]+waiting_time[i];
      total_waiting_time+=waiting_time[i];
      total_turn_around_time+=turn_around_time[i];
    }
   cout<<"Process\t\tBurst time\t\tWaiting time\t\tTurn around time"<<endl;
   for(i=0;i<n;i++)
   {
    cout<<process[i]<<"\t\t\t"<<burst_time[i]<<"\t\t\t"<<waiting_time[i]
    <<"\t\t\t"<<turn_around_time[i]<<endl;
   }
   cout<<"Average waiting time:"<<total_waiting_time/n<<endl;
   cout<<"Average Turnaround time:"<<total_turn_around_time/n;

   return 0;

}
```

Output:



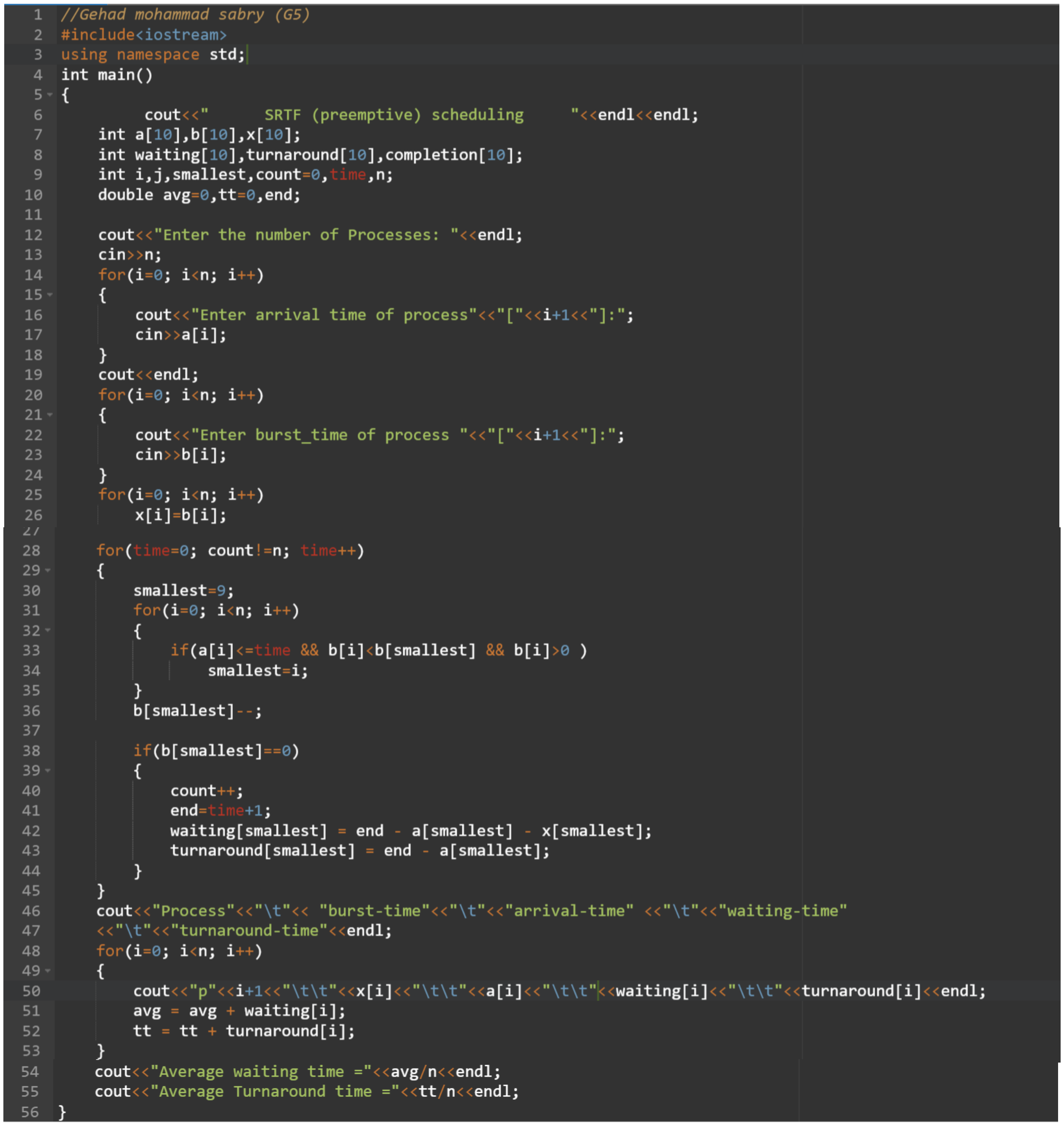
```
      SJF (non preemptive) scheduling

Enter number of processes
4
Enter burst_time of processes
Process[1]: 5
Process[2]: 2
Process[3]: 4
Process[4]: 1
Process          Burst time               Waiting time             Turn around time
4                        1                        0                        1
2                        2                        1                        3
3                        4                        3                        7
1                        5                        7                        12
Average waiting time:2.75
Average Turnaround time:5.75

...Program finished with exit code 0
Press ENTER to exit console.
```

Question 3:

SRTF (preemptive)

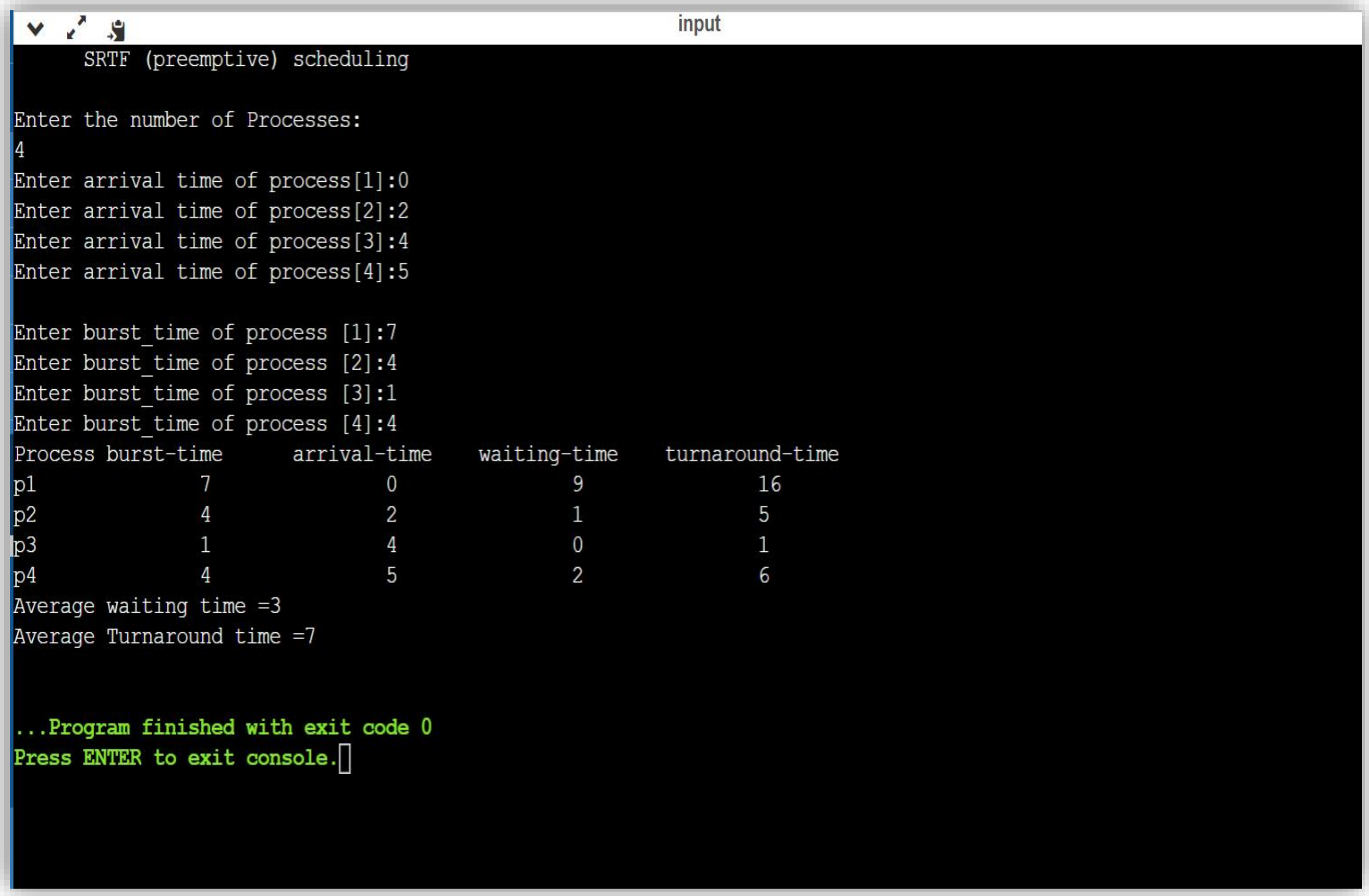
the Pseudo Code of SRTF (preemptive) scheduling

Step1: Traverse until all process gets completely executed.
Step2: Find process with minimum remaining time at every single time .
Step3: Reduce its time by 1.
Step4: Check if its remaining time becomes 0
Step5: Increment the counter of process completion.
Step6: Completion time of current process = current_time + 1;
Step7: Calculate waiting time for each completed process.
wt[i]= Completion time – arrival_time-burst_time
Step8: Increment time lap by one.
Step9: Find turnaround time (waiting_time + burst_time).



```cpp
//Gehad mohammad sabry (G5)
#include<iostream>
using namespace std;
int main()
{
         cout<<"      SRTF (preemptive) scheduling     "<<endl<<endl;
    int a[10],b[10],x[10];
    int waiting[10],turnaround[10],completion[10];
    int i,j,smallest,count=0,time,n;
    double avg=0,tt=0,end;

    cout<<"Enter the number of Processes: "<<endl;
    cin>>n;
    for(i=0; i<n; i++)
    {
        cout<<"Enter arrival time of process"<<"["<<i+1<<"]:";
        cin>>a[i];
    }
    cout<<endl;
    for(i=0; i<n; i++)
    {
        cout<<"Enter burst_time of process "<<"["<<i+1<<"]:";
        cin>>b[i];
    }
    for(i=0; i<n; i++)
        x[i]=b[i];

    for(time=0; count!=n; time++)
    {
        smallest=9;
        for(i=0; i<n; i++)
        {
            if(a[i]<=time && b[i]<b[smallest] && b[i]>0 )
                smallest=i;
        }
        b[smallest]--;

        if(b[smallest]==0)
        {
            count++;
            end=time+1;
            waiting[smallest] = end - a[smallest] - x[smallest];
            turnaround[smallest] = end - a[smallest];
        }
    }
    cout<<"Process"<<"\t"<< "burst-time"<<"\t"<<"arrival-time" <<"\t"<<"waiting-time"
    <<"\t"<<"turnaround-time"<<endl;
    for(i=0; i<n; i++)
    {
        cout<<"p"<<i+1<<"\t\t"<<x[i]<<"\t\t"<<a[i]<<"\t\t"<<waiting[i]<<"\t\t"<<turnaround[i]<<endl;
        avg = avg + waiting[i];
        tt = tt + turnaround[i];
    }
    cout<<"Average waiting time ="<<avg/n<<endl;
    cout<<"Average Turnaround time ="<<tt/n<<endl;
}
```

output:



```
      SRTF (preemptive) scheduling

Enter the number of Processes:
4
Enter arrival time of process[1]:0
Enter arrival time of process[2]:2
Enter arrival time of process[3]:4
Enter arrival time of process[4]:5

Enter burst_time of process [1]:7
Enter burst_time of process [2]:4
Enter burst_time of process [3]:1
Enter burst_time of process [4]:4
Process burst-time      arrival-time    waiting-time    turnaround-time
p1              7               0               9               16
p2              4               2               1               5
p3              1               4               0               1
p4              4               5               2               6
Average waiting time =3
Average Turnaround time =7


...Program finished with exit code 0
Press ENTER to exit console.
```