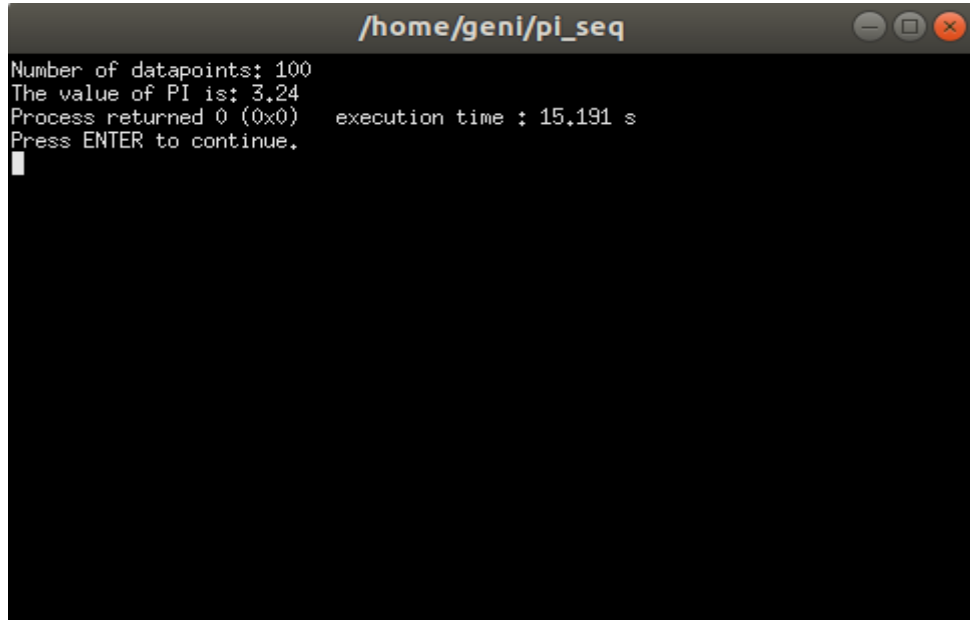


## Exercise 1

**-Write a sequential application pi\_seq in C or C++ that computes  $\pi$  for a given number of samples (command line argument). Test your application for various, large sample sizes to verify the correctness of your implementation.**

Results of three test ran for a number of 100, 1000 and 10 000 data points respectively.

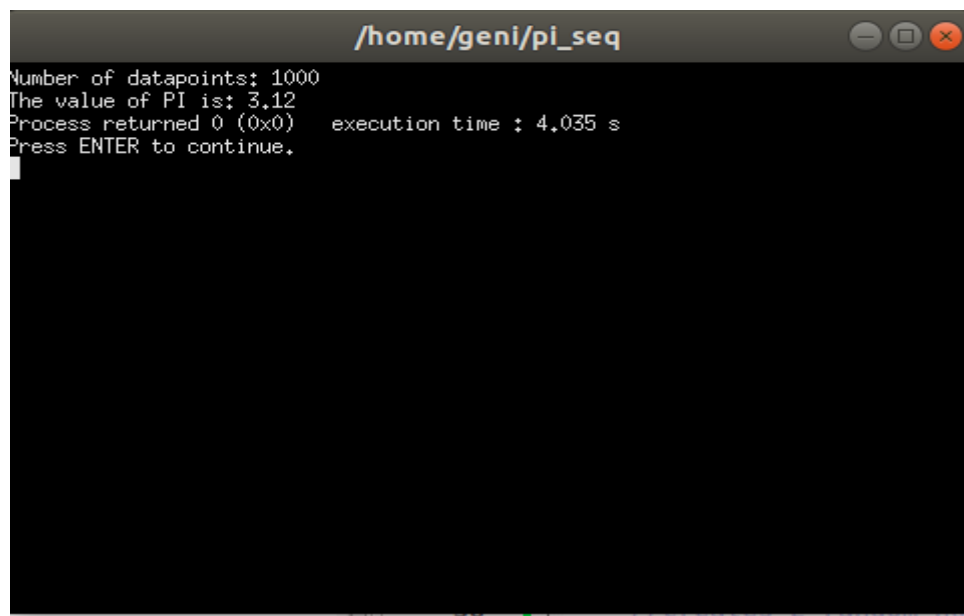
100 data



```
/home/geni/pi_seq
Number of datapoints: 100
The value of PI is: 3.24
Process returned 0 (0x0)   execution time : 15.191 s
Press ENTER to continue.
```

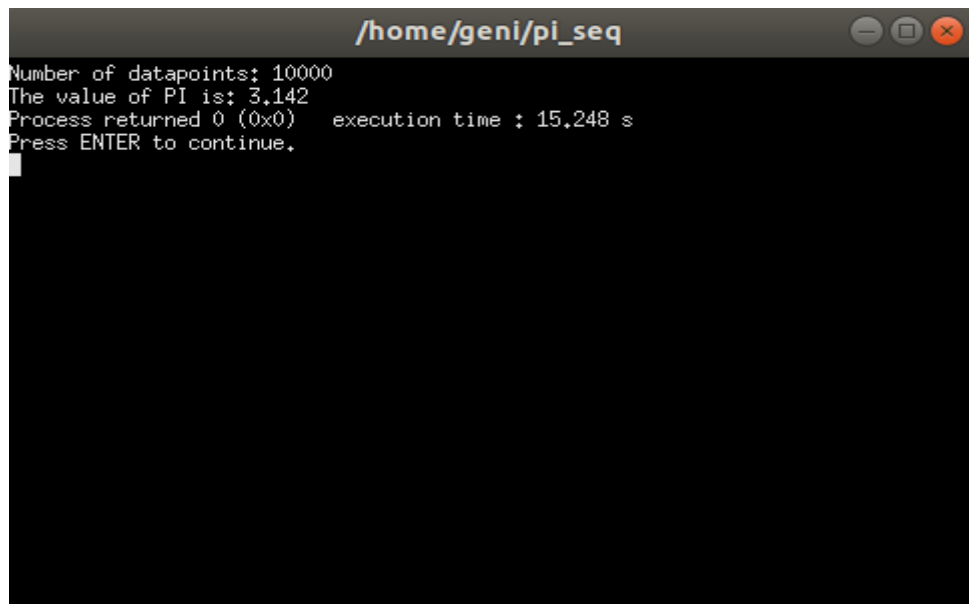
Results for  
points

1000 data



```
/home/geni/pi_seq
Number of datapoints: 1000
The value of PI is: 3.12
Process returned 0 (0x0)   execution time : 4.035 s
Press ENTER to continue.
```

Results for  
points

A terminal window titled `/home/geni/pi_seq` with standard Linux window controls. The terminal output is as follows:

```
Number of datapoints: 10000
The value of PI is: 3.142
Process returned 0 (0x0)   execution time : 15.248 s
Press ENTER to continue.
```

10 000 data

Result for  
points

**-Consider a parallelization strategy using MPI. Which communication pattern(s) would you choose and why?**

In the code, the idea is that each node except for one will have a number of data points and will calculate the number of points inside the circle. The number of points inside the circle and the total number of data points for each node will be sent to the root node, which will accumulate the results received for each node and will deliver a result. In this way, a bigger number of datapoints can be computed in a shorter time comparing to the sequential execution, giving us also a better result.

**-Discuss the effects and implications of your parallelization.**

In the algorithm of Monte-Carlo, it is clear that the more we increase the number of datapoints, the better the value of pi will be approximated. Parallelization, in this case, allows us to compute much more datapoints in a period of time which is comparable with the period of time needed to compute datapoints in the sequential algorithm. Therefore we get a much better result when using parallelization.