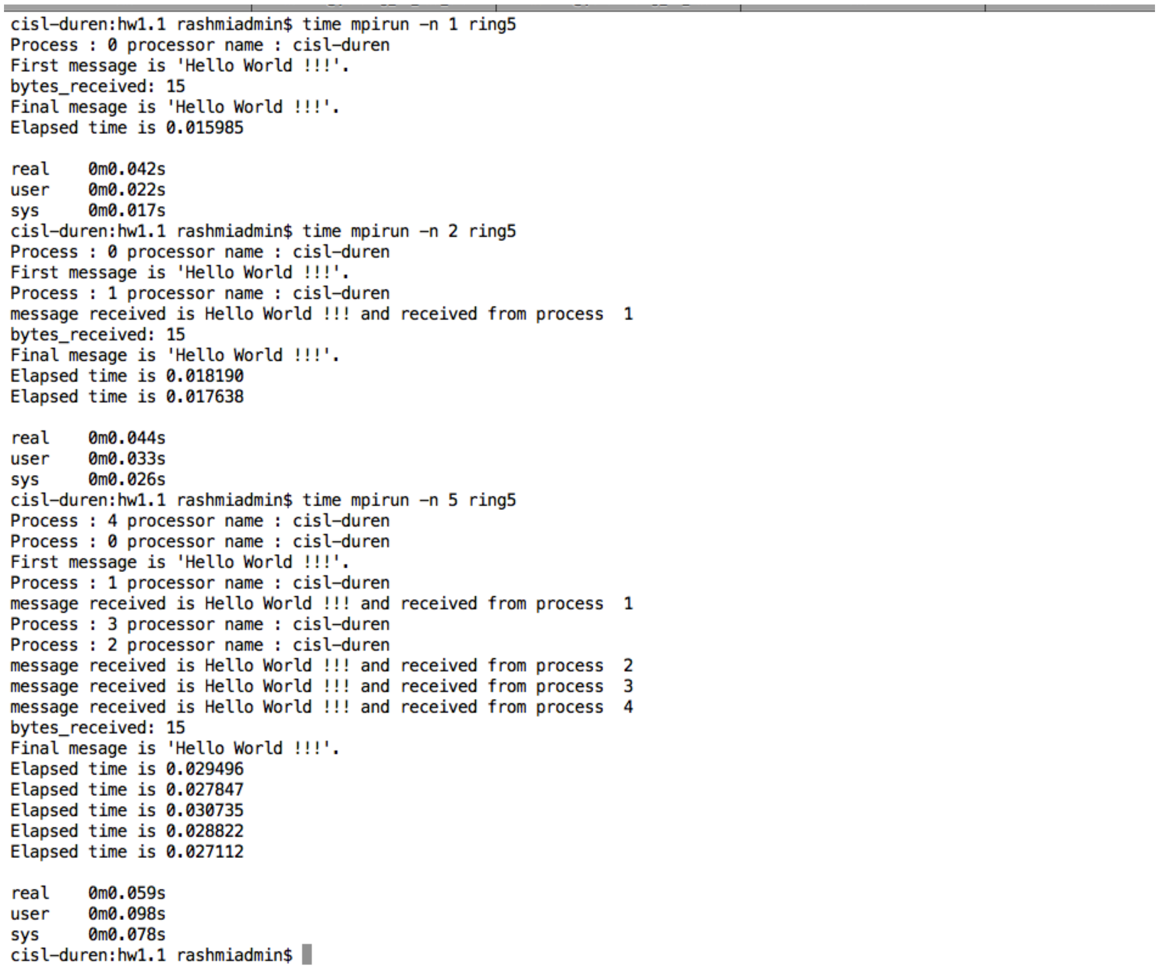
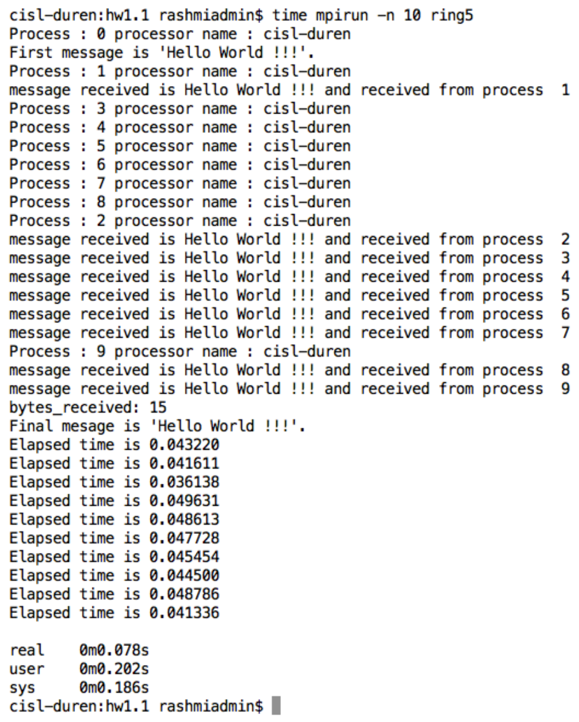
CSCI 5576 – HPSC [Rashmi Oak]

Assignment 1

Question 1

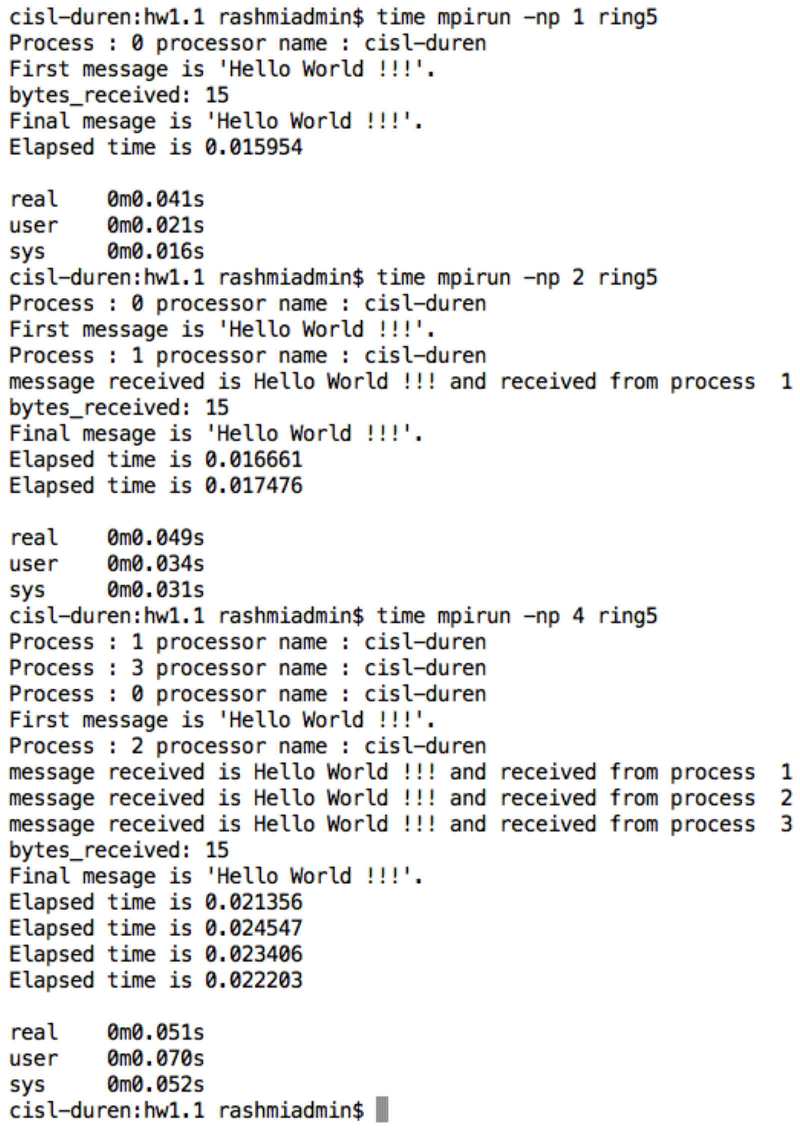


```
cisl-duren:hw1.1 rashmiadmin$ time mpirun -n 1 ring5
Process : 0 processor name : cisl-duren
First message is 'Hello World !!!'.
bytes_received: 15
Final mesage is 'Hello World !!!'.
Elapsed time is 0.015985

real    0m0.042s
user    0m0.022s
sys     0m0.017s
cisl-duren:hw1.1 rashmiadmin$ time mpirun -n 2 ring5
Process : 0 processor name : cisl-duren
First message is 'Hello World !!!'.
Process : 1 processor name : cisl-duren
message received is Hello World !!! and received from process  1
bytes_received: 15
Final mesage is 'Hello World !!!'.
Elapsed time is 0.018190
Elapsed time is 0.017638

real    0m0.044s
user    0m0.033s
sys     0m0.026s
cisl-duren:hw1.1 rashmiadmin$ time mpirun -n 5 ring5
Process : 4 processor name : cisl-duren
Process : 0 processor name : cisl-duren
First message is 'Hello World !!!'.
Process : 1 processor name : cisl-duren
message received is Hello World !!! and received from process  1
Process : 3 processor name : cisl-duren
Process : 2 processor name : cisl-duren
message received is Hello World !!! and received from process  2
message received is Hello World !!! and received from process  3
message received is Hello World !!! and received from process  4
bytes_received: 15
Final mesage is 'Hello World !!!'.
Elapsed time is 0.029496
Elapsed time is 0.027847
Elapsed time is 0.030735
Elapsed time is 0.028822
Elapsed time is 0.027112

real    0m0.059s
user    0m0.098s
sys     0m0.078s
cisl-duren:hw1.1 rashmiadmin$
```



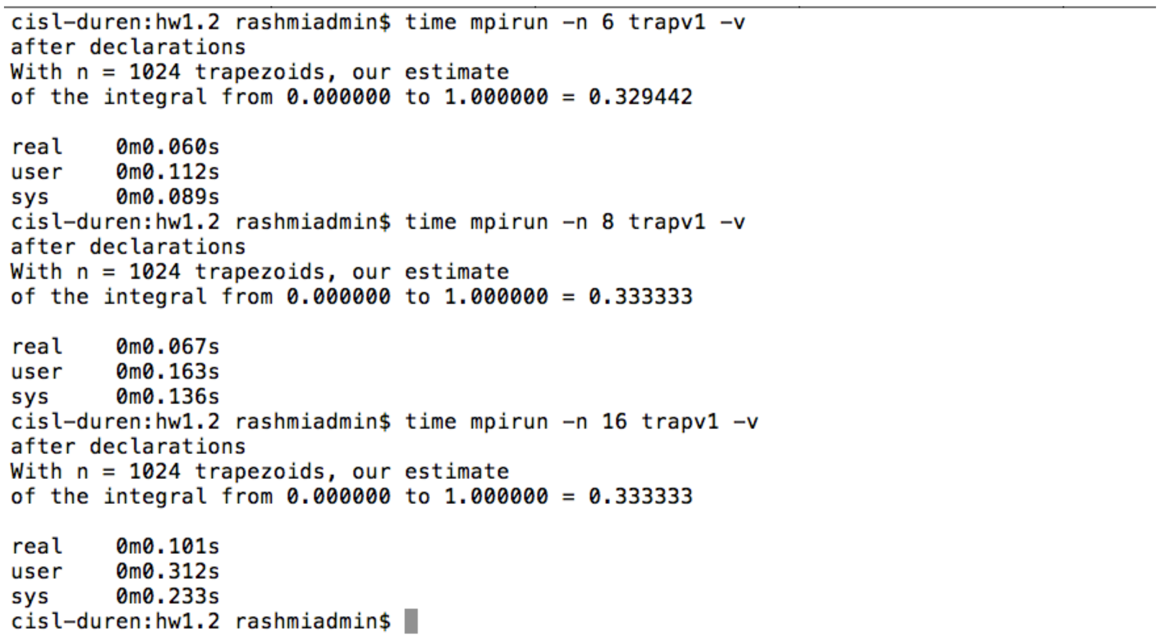
```
cisl-duren:hw1.1 rashmiadmin$ time mpirun -n 10 ring5
Process : 0 processor name : cisl-duren
First message is 'Hello World !!!'.
Process : 1 processor name : cisl-duren
message received is Hello World !!! and received from process  1
Process : 3 processor name : cisl-duren
Process : 4 processor name : cisl-duren
Process : 5 processor name : cisl-duren
Process : 6 processor name : cisl-duren
Process : 7 processor name : cisl-duren
Process : 8 processor name : cisl-duren
Process : 2 processor name : cisl-duren
message received is Hello World !!! and received from process  2
message received is Hello World !!! and received from process  3
message received is Hello World !!! and received from process  4
message received is Hello World !!! and received from process  5
message received is Hello World !!! and received from process  6
message received is Hello World !!! and received from process  7
Process : 9 processor name : cisl-duren
message received is Hello World !!! and received from process  8
message received is Hello World !!! and received from process  9
bytes_received: 15
Final mesage is 'Hello World !!!'.
Elapsed time is 0.043220
Elapsed time is 0.041611
Elapsed time is 0.036138
Elapsed time is 0.049631
Elapsed time is 0.048613
Elapsed time is 0.047728
Elapsed time is 0.045454
Elapsed time is 0.044500
Elapsed time is 0.048786
Elapsed time is 0.041336

real    0m0.078s
user    0m0.202s
sys     0m0.186s
cisl-duren:hw1.1 rashmiadmin$ 
```

1. Every process should receive first except for process 0 which should do MPI_send first. MPI_recv is a blocking call. So every process will block on MPI_recv. As soon as it receives message from its previous process, it will do MPI_send and send message to its next process.
2. Yes processes should receive and then send except for the first process here process 0. Because there has to be some process that should start the communication.
3. If program runs on the one processor, there will not be any communication overhead that occurs because of inter-processor communication. Program might run faster if ran on one single processor. See the following screenshot



```
cisl-duren:hw1.1 rashmiadmin$ time mpirun -np 1 ring5
Process : 0 processor name : cisl-duren
First message is 'Hello World !!!'.
bytes_received: 15
Final mesage is 'Hello World !!!'.
Elapsed time is 0.015954

real    0m0.041s
user    0m0.021s
sys     0m0.016s
cisl-duren:hw1.1 rashmiadmin$ time mpirun -np 2 ring5
Process : 0 processor name : cisl-duren
First message is 'Hello World !!!'.
Process : 1 processor name : cisl-duren
message received is Hello World !!! and received from process  1
bytes_received: 15
Final mesage is 'Hello World !!!'.
Elapsed time is 0.016661
Elapsed time is 0.017476

real    0m0.049s
user    0m0.034s
sys     0m0.031s
cisl-duren:hw1.1 rashmiadmin$ time mpirun -np 4 ring5
Process : 1 processor name : cisl-duren
Process : 3 processor name : cisl-duren
Process : 0 processor name : cisl-duren
First message is 'Hello World !!!'.
Process : 2 processor name : cisl-duren
message received is Hello World !!! and received from process  1
message received is Hello World !!! and received from process  2
message received is Hello World !!! and received from process  3
bytes_received: 15
Final mesage is 'Hello World !!!'.
Elapsed time is 0.021356
Elapsed time is 0.024547
Elapsed time is 0.023406
Elapsed time is 0.022203

real    0m0.051s
user    0m0.070s
sys     0m0.052s
cisl-duren:hw1.1 rashmiadmin$ 
```

Question 2



```
cisl-duren:hw1.2 rashmiadmin$ time mpirun -n 6 trapv1 -v
after declarations
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 1.000000 = 0.329442

real    0m0.060s
user    0m0.112s
sys     0m0.089s
cisl-duren:hw1.2 rashmiadmin$ time mpirun -n 8 trapv1 -v
after declarations
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 1.000000 = 0.333333

real    0m0.067s
user    0m0.163s
sys     0m0.136s
cisl-duren:hw1.2 rashmiadmin$ time mpirun -n 16 trapv1 -v
after declarations
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 1.000000 = 0.333333

real    0m0.101s
user    0m0.312s
sys     0m0.233s
cisl-duren:hw1.2 rashmiadmin$
```

```
/* Add up the integrals calculated by each process */
  if(my_rank == p-1){
    total = integral;
    printf("after declarations\n");
      //printf("integral equals %f\n",total);
    MPI_Send(&total, 1, MPI_FLOAT, next, 0, MPI_COMM_WORLD );
  }
  if(my_rank != 0 && my_rank != p-1){

    total = integral;
    MPI_Recv( &recv_integral, 1, MPI_FLOAT, prev, 0, MPI_COMM_WORLD, &status );
    total = total + recv_integral;
    MPI_Send(&total, 1, MPI_FLOAT, next, 0, MPI_COMM_WORLD );
      //as soon as you receive from previous process send it to next process

  }

  if (my_rank == 0) {
    total = integral;
    //for (source = 1; source < p; source++) {
      MPI_Recv(&recv_integral, 1, MPI_FLOAT, prev, tag,
        MPI_COMM_WORLD, &status);
      total = total + recv_integral;
    //}*/
  //} else {
      MPI_Send(&total, 1, MPI_FLOAT, dest,
      tag, MPI_COMM_WORLD);
  //}
```

}

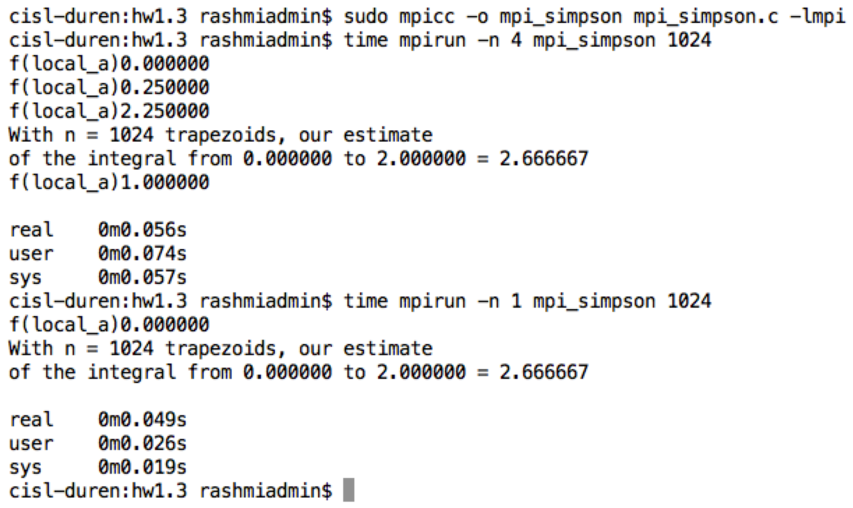
This is the code snippet for parallel trapezoidal rule.
Here for process P-1 MPI_send should be performed first. For rest of the process following should be the set of steps:

1. calculate own integral
1. MPI_recv: Receive integral value from previous process (here my_rank +1)
2. MPI_send : send this value to (my_rank +1)

Process 0 has to calculate its own integral, receive integral from process 1 and then add that value to its own integral and display the final result.

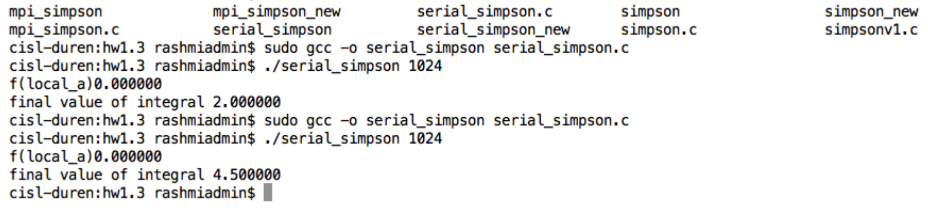
Question 3

For simpson's rule I ran into an issue which I had posted on moodle. I was not sure if I have to use simpson's rule on every integral and then add all the values from processes and then process 0 will give out final result or I have find a distribution scheme for example, [1 4 2 4 2][4 2 4 2][4 2 4 1] and then calculate simpson's integral value.
I am still not sure what exactly to do.
Due to time constraint, I did not get the first part up and running and hence I have implemented latter version of simpson's rule.

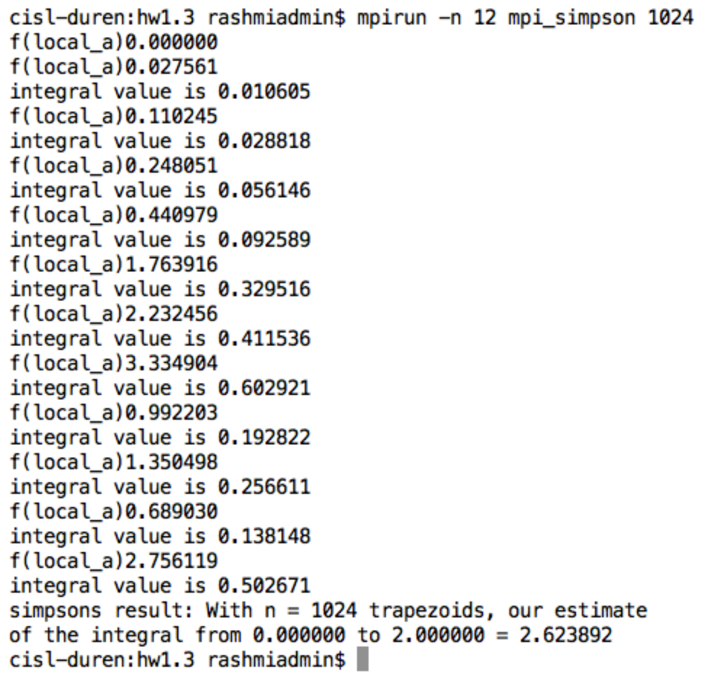


```
cisl-duren:hw1.3 rashmiadmin$ sudo mpicc -o mpi_simpson mpi_simpson.c -lmpi
cisl-duren:hw1.3 rashmiadmin$ time mpirun -n 4 mpi_simpson 1024
f(local_a)0.000000
f(local_a)0.250000
f(local_a)2.250000
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 2.000000 = 2.666667
f(local_a)1.000000

real	0m0.056s
user	0m0.074s
sys	0m0.057s
cisl-duren:hw1.3 rashmiadmin$ time mpirun -n 1 mpi_simpson 1024
f(local_a)0.000000
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 2.000000 = 2.666667

real	0m0.049s
user	0m0.026s
sys	0m0.019s
cisl-duren:hw1.3 rashmiadmin$ 
```

The serial version of the code runs faster since it is run on one single processor. The parallel version runs slower since it involves inter-process or inter-processor communication.

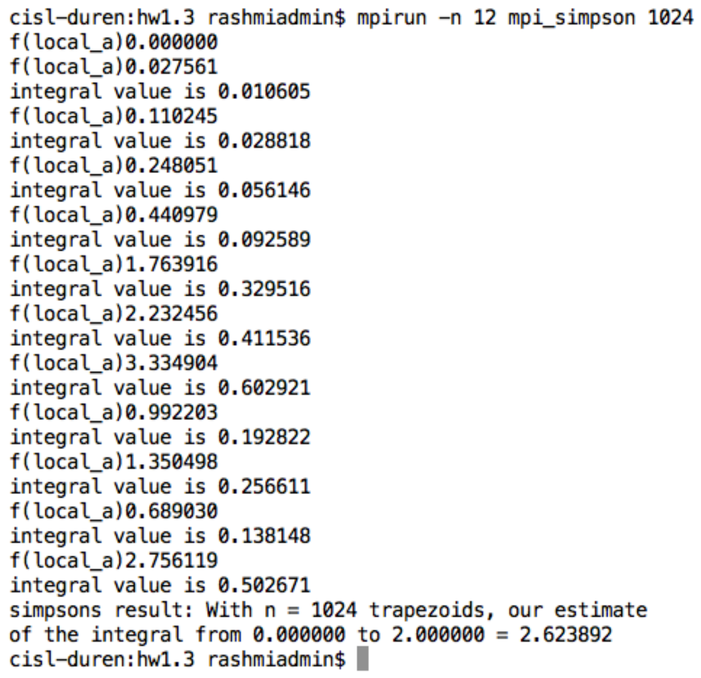


```
cisl-duren:hw1.3 rashmiadmin$ ls
mpi_simpson            mpi_simpson_new        serial_simpson.c       simpson                simpson_new
mpi_simpson.c          serial_simpson         serial_simpson_new     simpson.c              simpsonv1.c
cisl-duren:hw1.3 rashmiadmin$ sudo gcc -o serial_simpson serial_simpson.c
cisl-duren:hw1.3 rashmiadmin$ ./serial_simpson 1024
f(local_a)0.000000
final value of integral 2.000000
cisl-duren:hw1.3 rashmiadmin$ sudo gcc -o serial_simpson serial_simpson.c
cisl-duren:hw1.3 rashmiadmin$ ./serial_simpson 1024
f(local_a)0.000000
final value of integral 4.500000
cisl-duren:hw1.3 rashmiadmin$ 
```

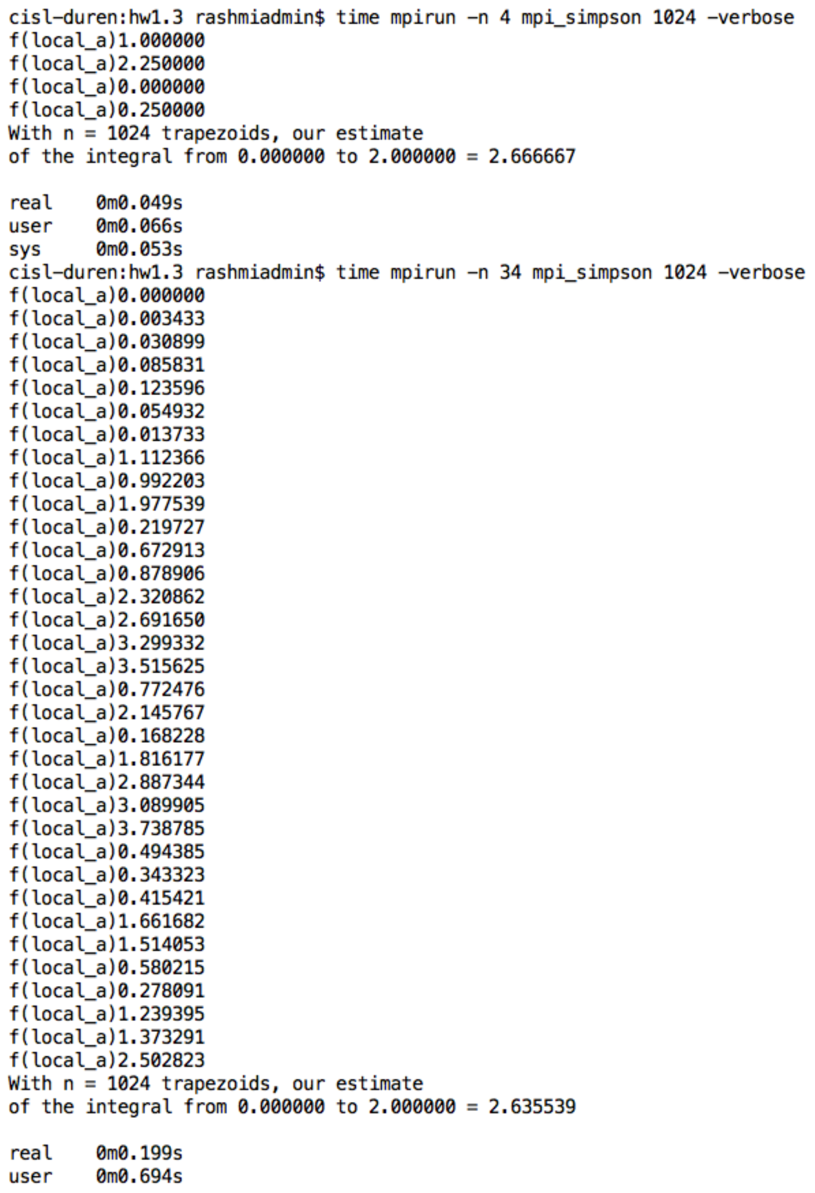


```
cisl-duren:hw1.3 rashmiadmin$ mpirun -n 12 mpi_simpson 1024
f(local_a)0.000000
f(local_a)0.027561
integral value is 0.010605
f(local_a)0.110245
integral value is 0.028818
f(local_a)0.248051
integral value is 0.056146
f(local_a)0.440979
integral value is 0.092589
f(local_a)1.763916
integral value is 0.329516
f(local_a)2.232456
integral value is 0.411536
f(local_a)3.334904
integral value is 0.602921
f(local_a)0.992203
integral value is 0.192822
f(local_a)1.350498
integral value is 0.256611
f(local_a)0.689030
integral value is 0.138148
f(local_a)2.756119
integral value is 0.502671
simpsons result: With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 2.000000 = 2.623892
cisl-duren:hw1.3 rashmiadmin$ 
```

output that includes integral calculated from each process:



```
cisl-duren:hw1.3 rashmiadmin$ mpirun -n 12 mpi_simpson 1024
f(local_a)0.000000
f(local_a)0.027561
integral value is 0.010605
f(local_a)0.110245
integral value is 0.028818
f(local_a)0.248051
integral value is 0.056146
f(local_a)0.440979
integral value is 0.092589
f(local_a)1.763916
integral value is 0.329516
f(local_a)2.232456
integral value is 0.411536
f(local_a)3.334904
integral value is 0.602921
f(local_a)0.992203
integral value is 0.192822
f(local_a)1.350498
integral value is 0.256611
f(local_a)0.689030
integral value is 0.138148
f(local_a)2.756119
integral value is 0.502671
simpsons result: With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 2.000000 = 2.623892
cisl-duren:hw1.3 rashmiadmin$ 
```


```
cisl-duren:hw1.3 rashmiadmin$ time mpirun -n 4 mpi_simpson 1024 -verbose
f(local_a)1.000000
f(local_a)2.250000
f(local_a)0.000000
f(local_a)0.250000
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 2.000000 = 2.666667

real	0m0.049s
user	0m0.066s
sys	0m0.053s
cisl-duren:hw1.3 rashmiadmin$ time mpirun -n 34 mpi_simpson 1024 -verbose
f(local_a)0.000000
f(local_a)0.003433
f(local_a)0.030899
f(local_a)0.085831
f(local_a)0.123596
f(local_a)0.054932
f(local_a)0.013733
f(local_a)1.112366
f(local_a)0.992203
f(local_a)1.977539
f(local_a)0.219727
f(local_a)0.672913
f(local_a)0.878906
f(local_a)2.320862
f(local_a)2.691650
f(local_a)3.299332
f(local_a)3.515625
f(local_a)0.772476
f(local_a)2.145767
f(local_a)0.168228
f(local_a)1.816177
f(local_a)2.887344
f(local_a)3.089905
f(local_a)3.738785
f(local_a)0.494385
f(local_a)0.343323
f(local_a)0.415421
f(local_a)1.661682
f(local_a)1.514053
f(local_a)0.580215
f(local_a)0.278091
f(local_a)1.239395
f(local_a)1.373291
f(local_a)2.502823
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 2.000000 = 2.635539

real	0m0.199s
user	0m0.694s
```

The above screenshot shows how the area is divided into a number of intervals and still the final result is same. So this shows that the code works.

Note:
I did use MPI_Wtime but it gives elapsed time for each process separately. Hence I used linux time command to time the execution of the program. I have used MPI_Wtime in second question to compare performance of both the linear reduce and all to one implementation of trapezoidal rule.