**Question1**.

Part1, file: a2q1.c

Compile Code:

Thoughts:

- Each process creates a local\_data, an array has 3 elements, [0] stores eggholder value, [1] [2] store value of x and y respectively.

- Each process finds out the min value in its own local

- Using MPI\_Gather, each process sends its min value to process 0, then process 0 to find out the min value among all processes.

Output:

A screenshot of a computer screen

Description automatically generated

Part2 files: a1q1p2, currently runTime is 10s for easy testing.   
mpicc -o a1q1p2.x a1q1p2.c -lm

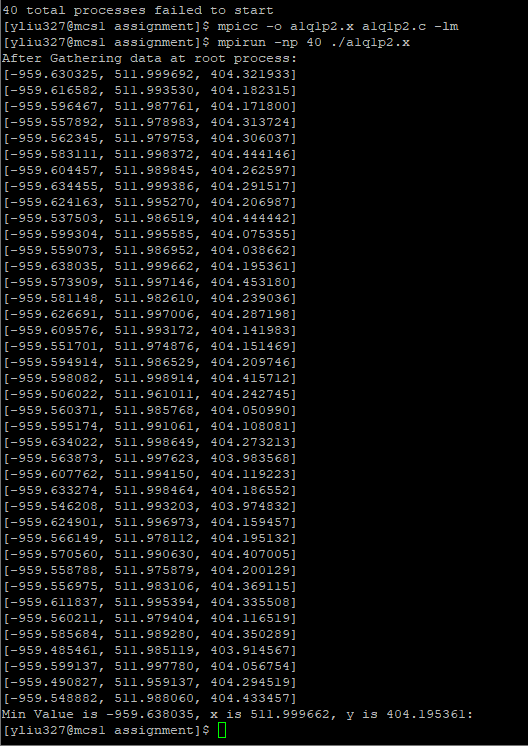
mpirun -np 40 ./a1q1p2.x

Thoughts:

- Base ideas are the same to part, each process finds out min and gather in process0

- gapTime = difftime(current\_time, start\_time); to set the running time

Output:



Part 3, file: a1q1p3.c

mpicc -o a1q1p3.x a1q1p3.c -lm

mpirun -np 40 ./a1q1p3.x

Thoughts:

- Base concepts are the same.

- Add a 5-sec check and flag which tells the loop break or continue

Output:

A screen shot of a computer

Description automatically generated

**Question2**. File: a1q2.c

mpicc -o a1q2.x a1q2.c

mpirun -np 40 ./a1q2.x

In my understanding from the hint and question description, file means in process 0 it can be generating a 16x16 matrix, and send each block, 4x4, to itself and p1, p2 and p3.

Thoughts:

- Cut 16x16 block into 4x4

- Send 4x4 for each process, using code below to get starting row and col.  


- time = n/block\_size to get total looping times, 4 times means there are 4 rows for 4 processes to get 16x16 matrix.

- Use MPI\_Barrier(MPI\_COMM\_WORLD); to sync each process.

Output:

