Part 1

For the first part of the assignment, I added to the code that was already given with worldsWorstworldsWorstHash1 and worldsWorstworldsWorstHash2. I created a main method that initialised MPI, printed out my name and student number and then checked the size of the world and what rank the node was.

I then created an if/if else/else statement that looked at what rank the node was. If the rank was 0, this was the first job and was given the initial values of 422 and 233. If the rank was 3, this was the last node of the program and so needed to receive the information sent by the other ranks, but did not need to send out information to next node as it is the last. With the nodes in between, they received information from the previous node and sent information to the next node. This then created a synchronised ring.

As can be seen from the results below, I got the same results that were given in the specifications.

Part 1 Results:

```
C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part1\Debug>mpiexec -n 4 "Sarah_Brennan_2962279_Part1.exe"
Student Name: Sarah Brennan
Student Number: 2962279
Student Name: Sarah Brennan
Student Number: 2962279
Student Name: Sarah Brennan
Student Number: 2962279
Student Name: Sarah Brennan
Student Number: 2962279
Rank 0 sending: -108
Rank 1 sending: -238
Rank 2 sending: -140
Result: hash1 = -66, hash2 = 58
```

Part 2

For the second part of the assignment I created a new c++ application. I started off by creating the different methods needed (printArray, countPrime and sumPrime). I then did the functionality for these methods. Once I had the functionality for it, I created a checkPrime method as I was doing the same thing twice for both countPrime and sumPrime.

I then created the main method that initialised MPI, world_siize and world_rank. I also created the if/else statement to call the coordinator for the first node and participant for all other nodes.

The coordinator and participants then work as asked in specifications. I decided to print out all the numbers within the created array and then the total sum of the prime numbers within that array.

Below are the results I printed, as well as tests I did to see the difference between using 1 node and 4 nodes.

Part 2 Results:

```
C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 4 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 100
Student Name: Sarah Brennan
Student Number: 2962279
42, 18, 35, 1, 20, 25, 29, 9, 13, 15, 6, 46, 32, 28, 12, 42, 46, 43, 28, 37, 42, 5, 3, 4, 43, 33, 22, 17, 19, 46,
 48, 27, 22, 39, 20, 13, 18, 50, 36, 45, 4, 12, 23, 34, 24, 15, 42, 12, 4, 19, 48, 45, 13, 8, 38, 10, 24, 42, 30,
 29, 17, 36, 41, 43, 39, 7, 41, 43, 15, 49, 47, 6, 41, 30, 21, 1, 7, 2, 44, 49, 30, 24, 35, 5, 7, 41, 17, 27, 32,
 9, 45, 40, 27, 24, 38, 39, 19, 33, 30, 42
Total sum is 684
```

Above are the results from my project with an array inputted of size 100. It prints out all the numbers within the array and then prints out the total sum of prime numbers.

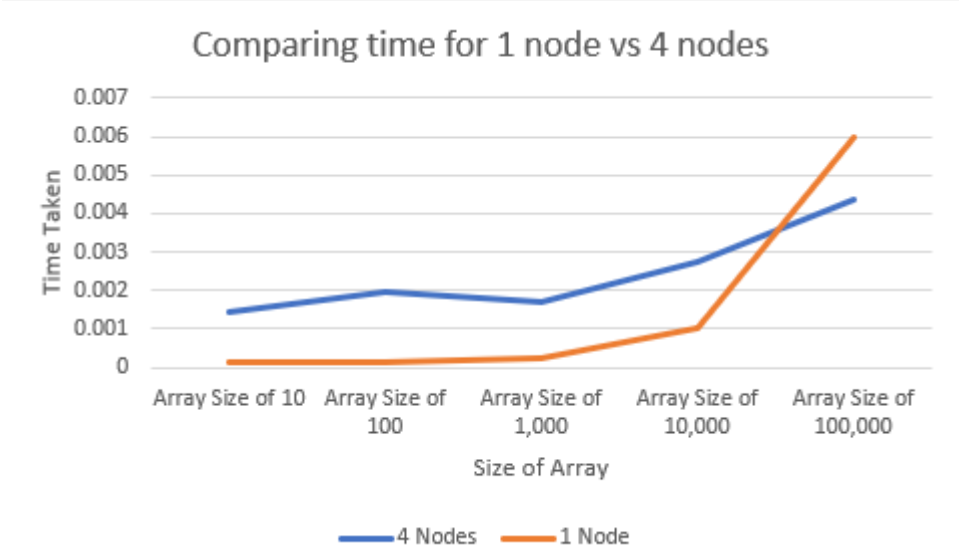
Comparing 4 nodes with 1 node

```
C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 4 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 12
Student Name: Sarah Brennan
Student Number: 2962279
Total sum of prime numbers in array is 42
Elapsed time: 0.0015455

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 1 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 12
Student Name: Sarah Brennan
Student Number: 2962279
Total sum of prime numbers in array is 42
Elapsed time: 0.0001379
```

```
C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 4 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 100
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 684
Elapsed time: 0.0019783

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 1 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 100
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 684
Elapsed time: 0.0001374

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 4 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 1000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 6077
Elapsed time: 0.0016928

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 1 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 1000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 6077
Elapsed time: 0.0002387

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 4 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 10000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 65214
Elapsed time: 0.0027585

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 1 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 10000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 65214
Elapsed time: 0.0010341

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 4 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 100000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 657741
Elapsed time: 0.0043817
```

```
C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 1 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 100000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 657741
Elapsed time: 0.0059962

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 4 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 1000000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 6553641
Elapsed time: 0.0214794

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 1 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 1000000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 6553641
Elapsed time: 0.0644561

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 4 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 10000000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 65633067
Elapsed time: 0.184605

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 1 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 10000000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 65633067
Elapsed time: 0.62788

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 4 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 100000000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 655901674
Elapsed time: 1.74608

C:\Users\sirla\source\repos\Sarah_Brennan_2962279_Part2\Debug>mpiexec -n 1 Sarah_Brennan_2962279_Part2
Please enter the array size evenly divisible by world size: 100000000
Student Name: Sarah Brennan
Student Number: 2962279
Total sum is 655901674
Elapsed time: 6.45089
```

I put the results I received from the above screenshots into the below graph:



As can be seen in the above graph, the crossover point where 4 nodes becomes quicker than 1 node is between 10,000 and 100,000 items in the array. Up until this number there is no need to have 4 nodes available to the user, as it is quicker to use 1 node to do all the calculations. However, when the size of the array gets huge, you can see a big difference in the amount of time it takes for 1 node to work out the calculations compared to 4 nodes. When I tried inputting 100,000,000 items into the array, the computing with 4 nodes took only 1.74608 secs. The computing using 1 node, however, took 6.45089 and so shows it would be much more time efficient to run more nodes the higher the array size is. With the lower array size of 12, however, it can be seen above that the computing using 1 node is much quicker as it does not need to do more work to get something done as it takes 0.0001379 seconds to compute using 1 node and 0.0015455 seconds to compute using 4 nodes.

In conclusion, if the number of items in the array is small and insignificant there is no need to compute using lots of different nodes. However, the bigger the number of items in an array, the more nodes you should use to compute them.

References

- Most work was completed based on MPI_notes.pdf on DS Moodle.
- Timer referenced from https://www.pluralsight.com/blog/software-development/how-to-measure-execution-time-intervals-in-c--