```
#include<iostream>
#include <array>
#include "mpi.h"
using namespace std;
int numOfProc, id , numberRecived , portion , array_size ;
int *arr;
int lower , upper;
MPI_Status status;
int allPrimes= 0 ;
int initilization= 0;

void master(){
	// accept user input for boudries
	cout<<"Enter lower : ";
	cin>>lower;
	cout<<"Enter upper : ";
	cin>>upper;
	//calculate the size of the master array
	array_size = upper -lower +1;
	// calculate the portion or array which will be distributed to proccesses
	portion =(array_size / (numOfProc-1));// to eliminate proc 0 from consideration
	//initialize
	arr = new int [array_size];
	//fill array by data between lower and upper bound
	for(int i = 0 ; i < array_size; i++){
		arr[i]= lower;
		lower++;
		//cout<<arr[i];
	}

	// assign tasks to proccesses
	double start_Send = MPI_Wtime();

	for (int p = 1 ; p< numOfProc ; p++){
		MPI_Send(&portion, 1, MPI_INT, p, 1, MPI_COMM_WORLD);
		MPI_Send(&arr[(p-1)*portion], portion, MPI_INT, p, 1, MPI_COMM_WORLD);
	}

	double end_Send = MPI_Wtime();
	// print the time duration in seconds
	std::cout<<"Send to all Process from master take " << end_Send - start_Send << "
seconds" ;

	// recive results from slaves
	double start_Recive = MPI_Wtime();

	for (int p = 1 ; p < numOfProc ; p++){
		MPI_Recv(&numberRecived, 1, MPI_INT, p, 1, MPI_COMM_WORLD,&status);
		allPrimes+=numberRecived;
		if (p+1 == numOfProc)
		cout<<endl<<"All Primes Count is "<<allPrimes;
	}

	double end_Recive = MPI_Wtime();
	// print the time duration in seconds
	std::cout <<endl<<"Recive from all Process in master " << end_Recive -
start_Recive << " seconds" ;
}
// do slave task
void Slave(){
	// start time
	double start_S_Recive = MPI_Wtime();
	// recive portion
	MPI_Recv(&portion, 1, MPI_INT, 0, 1, MPI_COMM_WORLD,&status);
	//define array size
	arr = new int [portion];
	//recive raay content (values) from master task
	MPI_Recv(arr, portion, MPI_INT, 0, 1, MPI_COMM_WORLD,&status);
```

```
	double end_S_Recive = MPI_Wtime();
	std::cout <<"Recive Process "<<id<<" take " << end_S_Recive - start_S_Recive << "
seconds" ;
	//cout << "numberRecived in proc "<<id<< " is "<<numberRecived;
	// this will count the prime number will be found in this slave task
	int primeCount=0;
	for(int i = 0 ; i < portion;i++){
		//cout<<"for"<<arr[i]<<" ";
		int n = arr[i];
	// not include 1 in primes so continue if the value was 1
		if (n==1)
			continue;
		// indicator for value status prime or not 0 for non prime and 1 for prime
		int c = 1;
		// prime extraction process
		for (int j = 2 ; j < n-1;j++){

			if (n%j!=0){
				continue;
			}
			else{
				c = 0 ;
				break;
			}

		}
		//validate on number
		if (c!=0){
			//cout<<"Prime from P "<<id <<" is " <<n;
			primeCount++;
		}
	}
	int newNum = primeCount;
	// send result  back to master
	// start time send
	double start_S_Send = MPI_Wtime();
	MPI_Send(&newNum, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
	// i did not use this but it will return the actual prime number for slave process
	MPI_Send(arr, portion, MPI_INT, 0, 1, MPI_COMM_WORLD);
	// end of time
	double end_S_Send = MPI_Wtime();
	// print the time duration in seconds
	std::cout<<endl<<"Send Process "<<id<<" take " << end_S_Send - start_S_Send << "
seconds" ;
}

int main(int argc, char *argv[])
{
	// initialize argument count and values
	MPI_Init(&argc, &argv);
	// define the whole proccesses numbers will run on this program
	MPI_Comm_size(MPI_COMM_WORLD, &numOfProc);
	//indicator for the current process will be returned into id
	MPI_Comm_rank(MPI_COMM_WORLD, &id);

	if (id == 0 && initilization ==0){
		master();
		initilization++;
		//cout <<"All PrimesCount "<<allPrimes;

	}else {
		Slave();
	}
	// empty ram and kill process
	MPI_Finalize();
}
```