```
#include[redacted]
#include[redacted]
#include[redacted]
#include[redacted]

#define ROOT 0
#define NOT_PRIME 1
#define PRIME 0
#define MAX_PRIME 3571



//calloc(number_of_elements_to_allocate, sizeof each element)
void print_primes(int from, int n, int* primes);


int main(int argc, char *argv[]){
    const int n = (argc ==2)? atoi(argv[1]) : MAX_PRIME;
    int sqrt_n = ceil(sqrt(n)) ;//based on the prime factor test
    //msg[0] = last number checked, msg[1] = increment
    int i, j, comm_sz, my_rank, loc_n, msg[2];
    double start_t, end_t;


    /*startup MPI */
    MPI_Init(NULL,NULL);

    /*get my rank among all the processes */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    /*get number of processes */
    MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);

    /*range of n per process */
    loc_n = n/comm_sz;
    int *local_primes = (int*) calloc(loc_n, sizeof(int));
    if(my_rank == ROOT){
        printf([redacted], loc_n);
        start_t = MPI_Wtime();
        local_primes[0] = NOT_PRIME; //1 is not prime
        for(i = 1; (i+1) <= sqrt_n; i++){

            if(!local_primes[i]){//not_marked
                j = i;
                do{
                    j = j+(i+1);
                    local_primes[j] = 1;
                }while(j+(i+1) < loc_n-1);

                msg[0] = j;//next number to mark
                msg[1] = i;//current increment
                if(comm_sz-1)//more than 1 node
                    MPI_Send(msg, 2, MPI_INT, my_rank+1, 0,
MPI_COMM_WORLD);

            }

        }
        msg[0] = j+(i);//next number to mark
        msg[1] = i;//current increment
        if(comm_sz-1)//more than 1 node
                    MPI_Send(msg, 2, MPI_INT, my_rank+1, 0,
MPI_COMM_WORLD);

    }else if(my_rank < (comm_sz-1)){//not root && not last node
        do{
            MPI_Recv(msg, 2, MPI_INT, my_rank-1, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
            i = msg[1];//current_increment
            j = msg[0];//next_number_to_mark

            for( j+=i+1; (j)<loc_n*(my_rank+1) ; j += (i+1)){//mark multiples of j

                local_primes[(j+1)-(loc_n*my_rank)] = NOT_PRIME;
            }
            msg[0] = j-(i+1);
            msg[1] = i;
            MPI_Send(msg, 2, MPI_INT, my_rank+1, 0, MPI_COMM_WORLD);

        }while((i+1) <= sqrt_n);

    }else{//if i am the last node
        do{
            MPI_Recv(msg, 2, MPI_INT, my_rank-1, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
            i = msg[1];//current_increment
            j = msg[0];//next_number_to_mark


            for(j+=i+1 ;  (j) <= loc_n*(my_rank+1) ; j += (i+1)){//mark multiples of j

                local_primes[(j+1)-(loc_n*my_rank)] = NOT_PRIME;
            }

        }while((i+1) <= sqrt_n);
    }
    ;

    //after this all primes are marked for this range.
    MPI_Barrier(MPI_COMM_WORLD);
    end_t = MPI_Wtime();
```

```
    if(my_rank != ROOT){//send array of marked primes
        MPI_Send(local_primes,loc_n, MPI_INT, 0, 0, MPI_COMM_WORLD);

    }else{//if i am the root
        print_primes(1, loc_n-1, local_primes);
        putchar('\n');
        for(i=1 ; i < comm_sz; i++){
            MPI_Recv(local_primes, loc_n, MPI_INT, i, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
            print_primes(i*(loc_n), loc_n+1, local_primes);
            putchar('\n');
        }
        end_t = MPI_Wtime();
        printf([redacted],comm_sz,n,
end_t-start_t);


    }/*
  if(my_rank == ROOT)
        printf("using %d procs, input was %d, time was  %lf \n",comm_sz,n,end_t-start_t);
    */

    //free(local_primes);

    MPI_Finalize();

    return 0;
}//end main

void print_primes(int from, int n, int* primes){
    int prime_counter=0,i;
    for(i = 1 ;i < n ; i++)
        if(!primes[i]){//not marked
            printf([redacted], from+i);
            //if(!(++prime_counter%20))
                //putint('\n');
            //++prime_counter;
        }
    //printf("# of primes in (1,%d) = %d", n,prime_counter);
}
```