```c
#include <stdio.h>
#include <math.h>
#include "mpi.h"
#include <stdlib.h>


#define myRank0 0
#define myRank1 1
#define myRank2 2
#define myRank3 3
#define myRank4 4
#define myRank5 5

main(int argc, char **argv)

{
    //---------------MPI
    int i,
        j,
        myRank,
        procN,
        source,
        tag=1,
        dest,
        nWorkers;

        MPI_Status status;
        nWorkers = procN-1;

        FILE *filePrime;
        FILE *fileVector;
    //---------------MPI

    //---------------Siede de Arethosthenes
    int sizeVector = 100000,    //tamanho do vetor a ser alocado
    *vectorN,               // vetor que irá armazenar os dados
    *primes,            // vetor para armazenar os primos
    numbers = 100,
    primoCont = 0;
    //---------------Siede de Arethosthenes


    srand (time (NULL));

    vectorN = (int*) malloc( sizeVector * sizeof (int) );


    MPI_Init (&argc , &argv);


    MPI_Comm_rank(MPI_COMM_WORLD, &myRank);
    MPI_Comm_size(MPI_COMM_WORLD, &procN);

    for ( j = 1; j < 11; j++)
    {
        if (myRank == myRank0)
        {

            fileVector = fopen("vetor_gerado.txt","w");

            for (i=0; i<sizeVector; i++)
            {
                vectorN[i] = (rand() %100)+2;
                fprintf(fileVector, "%i\n", vectorN[i]);
            }

            fclose(fileVector);

            MPI_Send (vectorN, sizeVector, MPI_INT, myRank1, tag, MPI_COMM_WORLD);

        }
        else
        {
            if (myRank == myRank5)
            {

                MPI_Recv(vectorN, sizeVector, MPI_INT, myRank4, tag, MPI_COMM_WORLD, &status);

                filePrime = fopen("primos.txt","w");

                for (i=0; i<sizeVector; i++)
                {

                    if (vectorN[i] != 0)
                    {
                    primoCont++;
                    fprintf(filePrime, "%i\n", vectorN[i]);

                    }
                }

                fclose(filePrime);

                printf("Números de primos encontrados: %d\n", primoCont);
                printf("Contador do Loop: %d\n", j);
                primoCont=0;
            }
            else if (myRank == myRank1)
            {

                MPI_Recv(vectorN, sizeVector, MPI_INT, myRank0, tag, MPI_COMM_WORLD, &status);

                for (i=0; i<sizeVector; i++)
                {
                    if ( vectorN[i] != 2 && (vectorN[i] % 2) == 0 )
                    {
                        vectorN[i] = 0;
                    }
                }

                MPI_Send (vectorN, sizeVector, MPI_INT, myRank2, tag, MPI_COMM_WORLD);

            }

            else if (myRank == myRank2)
            {
                MPI_Recv(vectorN, sizeVector, MPI_INT, myRank1, tag, MPI_COMM_WORLD, &status);

                for (i=0; i<sizeVector; i++)
                {
                    if ( vectorN[i] != 3 && (vectorN[i] % 3) == 0 )
                    {
                        vectorN[i] = 0;
                    }
                }

                MPI_Send (vectorN, sizeVector, MPI_INT, myRank3, tag, MPI_COMM_WORLD);

            }

            else if (myRank == myRank3)
            {
                MPI_Recv(vectorN, sizeVector, MPI_INT, myRank2, tag, MPI_COMM_WORLD, &status);

                for (i=0; i<sizeVector; i++)
                {
                    if ( vectorN[i] != 5 && (vectorN[i] % 5) == 0 )
                    {
                        vectorN[i] = 0;
                    }
                }

                MPI_Send (vectorN, sizeVector, MPI_INT, myRank4, tag, MPI_COMM_WORLD);
```

```c
            }

            else if (myRank == myRank4)
            {
                MPI_Recv(vectorN, sizeVector, MPI_INT, myRank3, tag, MPI_COMM_WORLD, &status);

                for (i=0; i<sizeVector; i++)
                {
                    if ( vectorN[i] != 7 && (vectorN[i] % 7) == 0 )
                    {
                        vectorN[i] = 0;
                    }
                }

                MPI_Send (vectorN, sizeVector, MPI_INT, myRank5, tag, MPI_COMM_WORLD);
            }

        }
    }

    free(vectorN);
    MPI_Finalize();

}
```