EJERCICIO 3:

```
[U214974@ohpc 3_primes]$ export OMP_NUM_THREADS=1
[U214974@ohpc 3_primes]$ ./primes

  Number of processors available = 40
  Number of threads =              1

                      Default        Static       Dynamic
        N     Pi(N)       Time          Time          Time

        1         0     0.000007      0.000001      0.000001
        2         1     0.000000      0.000000      0.000001
        4         2     0.000000      0.000000      0.000000
        8         4     0.000000      0.000000      0.000001
       16         6     0.000001      0.000001      0.000001
       32        11     0.000001      0.000001      0.000001
       64        18     0.000003      0.000003      0.000003
      128        31     0.000007      0.000007      0.000007
      256        54     0.000021      0.000020      0.000021
      512        97     0.000070      0.000069      0.000077
     1024       172     0.000237      0.000237      0.000240
     2048       309     0.000838      0.000835      0.000843
     4096       564     0.003074      0.003041      0.003076
     8192      1028     0.011011      0.010999      0.011031
    16384      1900     0.040839      0.040845      0.041098
    32768      3512     0.150577      0.150611      0.152572
    65536      6542     0.562433      0.562411      0.563653
export     131072     12251      2.114725      2.114760      2.116249

[U214974@ohpc 3_primes]$ export OMP_NUM_THREADS=2
[U214974@ohpc 3_primes]$ ./primes

  Number of processors available = 40
  Number of threads =              2

                      Default        Static       Dynamic
        N     Pi(N)       Time          Time          Time

        1         0     0.000049      0.000002      0.000004
        2         1     0.000001      0.000001      0.000002
        4         2     0.000001      0.000001      0.000001
        8         4     0.000001      0.000001      0.000002
       16         6     0.000001      0.000001      0.000002
       32        11     0.000002      0.000002      0.000003
       64        18     0.000002      0.000002      0.000004
      128        31     0.000006      0.000004      0.000006
      256        54     0.000015      0.000012      0.000015
```

```
   512       97     0.000054     0.000038     0.000041
  1024      172     0.000174     0.000120     0.000128
  2048      309     0.000602     0.000430     0.000437
  4096      564     0.002206     0.001543     0.001550
  8192     1028     0.007973     0.005502     0.005568
 16384     1900     0.029786     0.020524     0.020553
 32768     3512     0.109788     0.075521     0.075856
 65536     6542     0.413548     0.281702     0.281638
131072    12251     1.552137     1.062577     1.059529

[U214974@ohpc 3_primes]$ export OMP_NUM_THREADS=4
[U214974@ohpc 3_primes]$ ./primes

 Number of processors available = 40
 Number of threads =              4

                  Default        Static       Dynamic
    N     Pi(N)      Time          Time         Time

    1         0     0.000122     0.000002     0.000003
    2         1     0.000002     0.000001     0.000002
    4         2     0.000002     0.000002     0.000003
    8         4     0.000002     0.000002     0.000003
   16         6     0.000002     0.000002     0.000003
   32        11     0.000002     0.000002     0.000004
   64        18     0.000002     0.000002     0.000004
  128        31     0.000003     0.000003     0.000006
  256        54     0.000009     0.000007     0.000010
  512        97     0.000030     0.000024     0.000025
 1024       172     0.000096     0.000063     0.000077
 2048       309     0.000342     0.000220     0.000223
 4096       564     0.001270     0.000793     0.000804
 8192      1028     0.004747     0.002849     0.002798
16384      1900     0.017582     0.010415     0.010441
32768      3512     0.064142     0.048370     0.041458
65536      6542     0.239564     0.164446     0.144253
131072    12251     0.895219     0.539886     0.540896
```

1. Run the code with 1, 2, and 4 threads. Which scheduling technique scales the best?
2. Why? Discuss the results.

La tècnica per defecte no sembla escalar eficientment quan augmentem el nombre de threads. A mesura que augmentem la N, el temps d'execució s'incrementa de manera significativa. Això es deu ja que, aquesta tècnica no distribueix de manera òptima la càrrega de treball entre els threads.

D’altra banda, la tècnica estàtica sembla que té un major rendiment que la tècnica per defecte. Igual que abans, observem que a mesura que augmentem la N, el temps d'execució també ho fa. Però aquest increment és més progressiu i controlat, fet que indica una millor distribució de la càrrega de treball. Això es deu al fet que la programació estàtica divideix les iteracions en fragments de mida igual on comença el paral·lelisme, per això veiem un rendiment millor que la programació per defecte però pitjor que la dinàmica la qual té en compte la càrrega de treball variable.

Finalment, la tècnica dinàmica sembla que escala de manera més eficient els resultats en comparació amb les altres dues tècniques. Això ho veiem en la forma en què el temps d’execució augmenta de manera més suau a mesura que la N augmenta. Aquesta tècnica és més eficient quan treballem amb càrregues de treball variables, ja que distribueix la càrrega de treball de manera més equitativa entre threads. En el nostre cas la càrrega de treball varia en funció de l’entrada (N), llavors la tècnica dinàmica ens proporciona un major rendiment doncs una millor escalació.