**Problema de programacio´n paralela**

La suma de los nu´meros primos menores a 10 es:

2 + 3 + 5 + 7 = 17

Utilizando el lenguaje de programaci´on indicado por tu profesor (Scheme, Racket, Clojure), escribe dos versiones de un programa que calcule la suma de todos los nu´meros primos menores a 5*,*000*,*000 (cinco millones):

• La primera versi´on debe ser una implementacio´n convencional que realice el c´omputo de manera secuencial. • La segunda versi´on debe realizar el c´omputo de manera paralela a trav´es de los mecanismos provistos por el lenguaje siendo utilizado (por ejemplo *places* o la funci´on pmap). Debes procurar paralelizar el c´odigo aprovechando todos los nu´cleos disponibles en tu sistema.

Ambas versiones del programa deben dar 838*,*596*,*693*,*108 como resultado.

Con el fin de que el proceso de c´omputo sea m´as intenso para el CPU, utiliza el siguiente algoritmo:

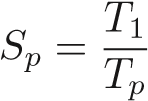
**Algoritmo para determinar si** *n* **es un nu´mero primo.** Devuelve *verdadero* o *falso*.

1. Si *n* es menor que 2, el algoritmo termina devolviendo *falso*.
2. Para *i* desde 2 hasta ⌈√*n* ⌉, realiza lo siguiente:

– El algoritmo termina devolviendo *falso* si *n* es divisible entre *i* de manera exacta, de otra se repite el ciclo con el siguiente valor de *i*.

1. El algoritmo termina devolviendo *verdadero* si el ciclo del punto anterior conluy´o de manera normal.

Mide el tiempo en que tarda en ejecutar cada versi´on del programa y calcula el *speedup* obtenido usando la siguiente f´ormula:



En donde:

* *p* es el nu´mero de procesadores (o nu´cleos).
* *T*1 es el tiempo que tarda en ejecutarse la versi´on secuencial del programa.
* *Tp* es el tiempo que tarda en ejecutarse la versi´on paralela del programa utilizando *p* procesadores.
* *Sp* es el *speedup* obtenido usando *p* procesadores.

Escribe un breve documento en donde reportes los resultados obtenidos y entr´egalo junto con el c´odigo fuente de tus implementaciones.