Primer parcial

1. Escribir un programa que lea palabras (letra a letra) del teclado. Cada palabra irá seguida por un número. La lectura terminará cuando se escriba un punto. El programa debe mostrar como salida el número de palabras que van seguidas de un número igual al número de letras de la palabra (palabras bien contadas) y el número de palabras que van seguidas por un número distinto al número de letras de la palabra. Cada palabra tiene como máximo 9 letras. Por ejemplo, para la entrada:

```
Este4es2un5ejemplo7de2varias6palabras9.
```

   Se mostraría la salida:

```
Palabras bien contadas: 5
Palabras mal contadas: 2
```

2. Escribir un programa que lea un número natural y encuentre y muestre el primer natural que es divisible por todos los valores menores o iguales al número leído. Por ejemplo, si el número es el 5, el primer natural que lo cumple es el 60 porque es divisible por 1, 2, 3, 4 y 5; si el número es 15, el programa debe mostrar el número 360360.

3. La conjetura de Collatz dice que, si aplicamos repetidamente el siguiente paso a cualquier número natural:

   - si el número es par, se divide entre 2
   - si el número es impar, se multiplica por 3 y se suma 1

   siempre se acaba repitiendo la secuencia 1 → 4 → 2 → 1, sin importar en qué número se empiece. Otros ejemplos:

   Empezando en n = 6, la secuencia sería: 6 → 3 → 10 → 5 → 16 → 8 → 4 → 2 → 1 (8pasos)
   Empezando en n = 11, la secuencia sería: 11 → 34 → 17 → 52 → 26 → 13 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1(14 pasos)

   Nadie ha encontrado ningún número que incumpla esta conjetura. Escribe un programa que dado un número entero natural leído por teclado, calcule el número de pasos necesarios hasta llegar al 1 por primera vez. Por ejemplo, dado 27, debería imprimir que hay 111 pasos.

Segundo parcial

1. En matemáticas, una potencia perfecta es un número natural que puede ser expresado como una potencia de otro número. Por ejemplo, el 8 es una potencia perfecta porque puede ser expresado como 2^3, 625 es una potencia perfecta porque puede ser expresado como 5^4, 10 no es una potencia perfecta porque no puede ser expresado como potencia de ningún número.

Escribir un subalgoritmo que devuelve al algoritmo principal información de si un número recibido como parámetro es una potencia perfecta. En el caso de que el número recibido sea una potencia perfecta se pasará al algoritmo principal la base y el exponente de los números que hacen esa potencia; si no es una potencia perfecta, estos parámetros no contendrán información.

El número debe leerse en el algoritmo principal y la información sobre si es o no una potencia perfecta y los números con los que se obtiene esa potencia deben mostrarse también en el algoritmo principal.

Para el número 625, se mostrará en el algoritmo principal lo siguiente:

```
625 es una potencia perfecta, se obtiene como 2 elevado a 3
```

Para el número 10, se mostrará en el algoritmo principal:

```
10 no es una potencia perfecta
```

NOTA: Para la resolución de este problema sólo se pueden usar los tipos simples y debe escribirse el subalgoritmo que se indica. Se puede usar la función *pow(a, b)* que devuelve el valor de a^b y la función *sqrt(a)* que devuelve la raíz cuadrada de a. Para ello hay que incluir la librería `cmath`.

2. Los números poderosos, son aquellos que cumplen que si un número primo p es divisor suyo, también lo es su cuadrado p^2. Por ejemplo, el 36 es un número poderoso ya que sus divisores primos son 2 y 3, y sus cuadrados, 4 y 9, también son divisores de 36. El número 3 no lo es ya que su divisor primo es 3, pero el cuadrado de este divisor primo, 9, no divide a 3. Se pide:

a) Escribir un subalgoritmo que devuelva un valor verdadero si el número que se le pasa como parámetro es poderoso.

b) Escribir un programa que pida dos valores de tipo natural y muestre todos los números poderosos que hay entre esos valores y muestre cúantos hay en total en ese intervalo.

Por ejemplo, si se escribieran 1 y 100 se mostraría:

```
1 4 8 9 16 25 27 32 36 49 64 72 81 100
Hay 14 números poderosos en ese intervalo
```

Nota: Se pueden escribir otros subalgoritmos si se considera necesario. Sólo pueden utilizarse tipos simples.