Problem 3

Find the largest prime factor of a composite number

Let the given number be *n* and let *k* = 2, 3, 4, 5, ... For each *k*, if it is a factor of *n* then we divide *n* by *k* and completely divide out each *k* before moving to the next *k*. It can be seen that when *k* is a factor it will necessarily be prime, as all smaller factors have been removed, and the final result of this process will be *n* = 1.
So a very crude version might be:

```
//if you had some trouble getting the number into a variable
//see the note at the end on page 2.
n=”the evil big number”
factor=2
lastFactor=1
while n>1
   if n mod factor=0
    then
      lastFactor=factor
      n=n div factor
      while n mod factor=0
         n=n div factor
   factor=factor+1
output lastFactor
```

This works for the given number, but can be improved in several ways.
First of all: 2 is the only even prime, so if we treat 2 separately we can increase `factor` with 2 every step.

```
n=”the evil big number”
if n mod 2=0
 then
   n=n div 2
   lastFactor=2
   while n mod 2=0
      n=n div 2
  else
    lastFactor=1
factor=3
while n>1
   if n mod factor=0
      then
         n=n div factor
         lastFactor=factor
         while n mod factor=0
            n=n div factor
   factor=factor+2
output lastFactor
```


Author: hk

The largest primefactor in the number you were asked was not very large.
Suppose now that the number to be factored is 2*1009*(some big prime) then it would take quite a time for `factor` to reach that prime. This can be improved upon by the following key realisation: every number n can at most have one prime factor greater than $\sqrt{n}$. If we, after dividing out some prime factor, calculate the square root of the remaining number we can use that square root as upper limit for `factor`. If `factor` exceeds this square root we know the remaining number is prime.
An implementation of this would be:

```
n=”the evil big number”
if n mod 2=0
 then
   lastFactor=2
   n=n div 2
   while n mod 2=0
      n=n div 2
 else
    lastFactor=1
factor=3
maxFactor=√n
while n>1 and factor<=maxFactor
    if n mod factor=0
      then
         n=n div factor
         lastFactor=factor
         while n mod factor=0
            n=n div factor
         maxFactor=√n
   factor=factor+2
if n=1
   then
     output lastFactor
   else
     output n
```

Note

The number 600851475143 you were asked to factor is larger than 2^{31}-1= 2,147,483,647.
In typed languages like C and derivatives , Pascal and derivatives and Visual Basic the data type int, or integer usually is a 32 bit signed integer and cannot hold numbers larger than 2^{31}-1.
You must then use another datatype that can hold larger numbers. Names for these types are typically: long, long long or int64 or _int64. If you declared your variable in that way it might also be necessary to place a suffix after the number like 600851475143L.
As this differs from language to language and from implementation to implementation you are advised to look up the ranges and conventions for the different datatypes in your documentation. You will need this knowledge on several occasions in the problems to come.

Special care is also required when adding a great many numbers so that the sum of those numbers exceeds the 32 bit signed integer limit or when multiplying two numbers whose product exceeds that limit. In that last case a so called *typecast* might be necessary.
But this overview is not a programming tutorial so you should study those details about your programming language carefully before proceeding. In numerous cases help on a problem was asked that would not have been necessary if this information was gathered beforehand.

Author: hk