Algorithm 1 Power_of_2(natural number **k)**
Input: ***k*** , a natural number
Output: ***k***-th power of ***2***

Algorithm:
if ***k* = *0*, then** return ***1***;
else return **2*Power_of_2(*k* - *1*) .**

By way of comparison, let us see how the same problem can be solved by an iterative algorithm.

Algorithm 2 Power_of_2(natural number **k)**
Input: ***k*** , a natural number
Output: ***k***-th power of ***2***

Algorithm:
int *i, power*;
***i* := *0*;**
***power* := *1*;**
while(*i* < *k*) {
    ***power* := *power* * *2*;**
    ***i* := *i* + *1*;**
}
return ***power* .**

Algorithm 3: **Even(**positive integer ***k*****)**
Input: ***k***, a positive integer
Output: ***k***-th even natural number (the first even being ***0***)
Algorithm:
int ***i, even*;**
***i* := *1*;**
***even* := *0*;**
while(***i* < *k*** **) {**
    ***even* := *even* + *2*;**
    ***i* := *i* + *1*;**
}
return ***even*** **.**

Algorithm 4 **Natural(**a number ***x*****)**
Input: A number ***x***
Output: "**Yes**" if ***x*** is a natural number, else "**No**"

Algorithm:
if ***x* < *0*,** **then** return "**No**"
else
  if ***x* = *0*,** **then** return "**Yes**"
  else return **Natural(** ***x* - *1*** **)**