Problem D

Growing trees from numbers

Trees in this problem are abstract entities, not biological species. Sorry about that. We shall consider only so called rooted trees which consist of a set of nodes connected by links in such
a way that each node, except one (called root node), has exactly one predecessor (called parent node), and any number of successor nodes (called child nodes). If a node has no children it is called terminal node. We can also define a rooted tree recursively as the root node and a number of subtrees connected to the root.
Suppose we have a rooted tree T. We can encode the tree into a positive integer G(T) as follows (Goebel encoding).

1. If the tree is trivial, i.e. contains only the root:

   G(T) = 1

2. If T has the form of T = root(T1, ... ,Tk), where T1,..., Tk are subtrees connected to the root:

   G(T) = P(G(T1)) * ... * P(G(Tk))

   P(i) above denotes i-th prime number; asterisk '*' denotes multiplication. Recall, P(1)=2,
   P(2)=3, P(3)=5, P(4)=7, and so on.

Input
You are asked to write a program that reads a sequence of positive integers from the standard input stream, considers each input number as encoded representation of a rooted tree, and generates the textual form of the tree encoding process. Last number in the input sequence is 0; it will terminate the program execution. Input numbers are in the range 0..65535 (16 bits unsigned).

Output
To be specific, the following example gives the expected output of the program. The root node for a tree encoded by number n has always the form ':n>'. Other nodes are printed as ''. Tree structure is represented by simple horizontal
and vertical connectors.
EXAMPLE

Input

1 3 35 999 0

Output

```
:1>

:3>--<3:2>--<2:1>
```

```
:35>--<5:3>--<3:2>--<2:1>
   |
   |--<7:4>--<2:1>
          |
          |--<2:1>

:999>--<3:2>--<2:1>
    |
    |--<3:2>--<2:1>
    |
    |--<3:2>--<2:1>
    |
    |--<37:12>--<2:1>
             |
             |--<2:1>
             |
             |--<3:2>--<2:1>
```

Solution

```
                             TEST
input        1 3 35 999 470 540 71 0
output

:1>

:3>--<3:2>--<2:1>

:35>--<5:3>--<3:2>--<2:1>
   |
   |--<7:4>--<2:1>
          |
          |--<2:1>

:999>--<3:2>--<2:1>
    |
    |--<3:2>--<2:1>
    |
    |--<3:2>--<2:1>
    |
    |--<37:12>--<2:1>
             |
             |--<2:1>
             |
             |--<3:2>--<2:1>

:470>--<2:1>
    |
    |--<5:3>--<3:2>--<2:1>
    |
    |--<47:15>--<3:2>--<2:1>
             |
             |--<5:3>--<3:2>--<2:1>

:540>--<2:1>
    |
    |--<2:1>
    |
    |--<3:2>--<2:1>
    |
    |--<3:2>--<2:1>
    |
    |--<3:2>--<2:1>
    |
    |--<5:3>--<3:2>--<2:1>
```

```
:71>--<71:20>--<2:1>
            |
            |--<2:1>
            |
            |--<5:3>--<3:2>--<2:1>
```

Listing

```
#include
#define LMX 11
#define PMX 6543

/* PROTOTYPY FUNKCJI */
void     GenTree(unsigned n,int cp);
void     VerticalLines(void);
void     np(unsigned n, unsigned *q, unsigned *i);
void     generate(int i);
unsigned prime(int i);
int tv[LMX];
int mtv=0;
unsigned tp[PMX] = {1,2,3,5};
int mtp = 3;

unsigned prime(int i)
{ unsigned pp, k, n=tp[mtp];

  if(i<1 || i>=PMX) return 1;

  while (mtp",n);
  if (n==1) putchar('\n');
  else
  { np(n,&q;,&i;);
    tv[++mtv]=cp+l;
    do
    { l1 = printf("--<%u",q);
      if(q==n) --mtv;
      GenTree(i,cp+l+l1);
      n/=q;
      if(n>1)
      { np(n,&q;,&i;);
        VerticalLines(); putchar('\n');
        if(n!=q) VerticalLines(); else VerticalLines();
      }
    } while (n>1);
  }
}

void VerticalLines()
{ int i,j,k;

  for(j=0,i=1; i<=mtv; i++)
  { k=tv[i];
    printf("%*c",k-j,'|');
    j=k;
  }
}

void np(unsigned n,unsigned *q,unsigned *i)
{ *i=1; *q=2;
  while(n % *q != 0) *q=prime(++(*i));
}
```

� *zwir*, *wierzej*, Mon Oct 28 23:01:26 MET DST 1996