```
LEMPEL-ZIV-WELCH-COMPRESS:
  create empty dictionary D;
  dsize := 0;
  for all symbols s in alphabet
    D.insert(s,dsize); dsize := dsize + 1;

  while there is more characters on the input
    s := longest prefix from input
         such that s is in D  (*)
    output D.search(s);
    c := peek next character from input
    D.insert(s+c,dsize);
    dsize := dsize + 1;
```

COCOA_AND_BANANAS Alphabet: 0:_,1:A,2:B,3:C,4:D,5:N,6:O,7:S

Longest prefix	Output code	Entry in dictionary	Dictionary number
C	3	CO	8
O	6	OC	9
CO	8	COA	10
A	1	A_	11
_	0	_A	12
A	1	AN	13
N	5	ND	14
D	4	D_	15
_	0	_B	16
B	2	BA	17
AN	13	ANA	18
ANA	18	ANAS	19
S	7	--	--

```
LEMPEL-ZIV-WELCH-DECOMPRESS:
  create empty dictionary D
  dsize := 0;
  for all symbols s in alphabet
    D.insert(dsize,s); dsize := dsize + 1;

  code := next code from the input
  s := D.search(code); output s
  while there are more codes on the input
    lasts := s
    code := next code from the input
    s := D.search(code); output s;
    D.insert(dsize,lasts+s[1]); dsize := dsize + 1;
```

Alphabet: 0:_,1:A,2:B,3:C,4:D,5:N,6:O,7:S

Code	Decoded string	Entry in dictionary	Dictionary number
3	C	--	--
6	O	CO	8
8	CO	OC	9
1	A	COA	10
0	_	A_	11
1	A	_A	12
5	N	AN	13
4	D	ND	14
0	_	D_	15
2	B	_B	16
13	AN	BA	17
18	????		

```
LEMPEL-ZIV-WELCH-DECOMPRESS:
  create empty dictionary D
  dsize := 0;
  for all symbols s in alphabet
    D.insert(dsize,s); dsize := dsize + 1;

  code := next code from the input
  s := D.search(code); output s
  while there are more codes on the input
    lasts := s
    code := next code from the input
**  if code = dsize then s := lasts + lasts[1];
**  else s := D.search(code);
    output s;
    D.insert(dsize,lasts+s[1]); dsize := dsize + 1;
```

Rozmieňanie peňazí

```
while S>0 do
  c:=value of the largest coin no larger than S;
  num:=S div c;
  pay out num coins of value c;
  S:=S-num*c;
```

```
function coins(i):
  // base cases
  if (i=0) then return 0;

  // recursion:
  min:=infinity;
  for j:=0 to m do
    if (d[j]<=i) then
      smaller_sol:=coins(i-d[j]);
      if smaller_sol<min then min:=smaller_sol;

  return 1+min;

// ------------ main program -----------------
return change_coins(S);
```

```
coins[0]:=0;
for i:=1 to S do
  min:=infinity;
  for j:=1 to m do
    if d[j]<=i and coins[i-d[j]]<min then
      min:=coins[i-d[j]];
  coins[i]:=1+min;

return coins[S];
```

Time: $\Theta(mS)$

```
coins[0]:=0;
for i:=1 to S do
  min:=infinity;
  for j:=1 to m do
    if d[j]<=i and coins[i-d[j]]<min then
      min:=coins[i-d[j]];
*     minchoice:=j;
  coins[i]:=1+min;
* choice[i]:=minchoice;
// ----------- recover solution --------
if coins[S]=infinity then write 'No solution!';
else
  while S>0 do
    write d[choice[S]];
    S:=S-d[choice[S]];
```