```
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;
```

```
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;
```

```
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;
```

```
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]];
```