Do you know how to print a diamond?

I mean printing letters in a certain fashion so that the sequence of letters looks like a diamond.
Like this:

```
  A
 B B
C   C
 B B
  A
```

This was a diamond made with letters from `A` to `C`. Here is one with letters from `A` to `F`:

```
     A
    B B
   C   C
  D     D
 E       E
F         F
 E       E
  D     D
   C   C
    B B
     A
```

We launch *gforth* with our program, like this:, and then we type something like:

```
gforth Diamond.fs ↩
```

And once *gforth* has started, we enter `DIAMOND` commands like this:

```
CHAR C DIAMOND ↩
CHAR F DIAMOND ↩
```

And the program should print diamonds like the ones above.

Let's explore the problem a bit. Do we know how to print letters?

What about spaces?

How many spaces do you need before printing `A` in the first line of the diamond?

I'm not sure. What do you mean by *print a diamond*?

I see.

Okay. How do you tell the computer you want a diamond printed?

Understood. Let's write the `Diamond.fs` program!
But where do we start?

That's easy. Here's how to print letters:

```
CHAR A EMIT ↩ A ok
CHAR F EMIT ↩ F ok
```

We can print any amount of spaces we want:

```
3 SPACES ↩    ok
8 SPACES ↩        ok
```

It depends on the size of the diamond we want to print.

How do we know that size?

It depends on the *char* parameter given to our program. The sequence `CHAR A` will leave the number `65` on the parameter stack.
The sequence `CHAR F` will leave `70`, etc.

Let's start our program by creating a word, `LETTER#` wich given a *char*, will produce the number of that letter.

→ `CHAR A LETTER#` should leave `1` on the stack,

→ `CHAR B LETTER#` should leave `2`,

→ `CHAR C LETTER#` should leave `3`,

→ and so on.

Ok.

```
\ Diamond.fs

CHAR A CONSTANT START

: LETTER#     \ c -- l#
   START - 1+ ;
```

Diamond.fs

```
gforth Diamond.fs ↩
CHAR A LETTER# . ↩ 1 ok
CHAR B LETTER# . ↩ 2 ok
CHAR F LETTER# . ↩ 3 ok
CHAR Z LETTER# . ↩ 26 ok
```

In a diamond containing a line with the letters `B`, how many central spaces do we need to print between these letters on that line?

1.

In a diamond containing a line with the letters `C`, how many central spaces do we need to print between these letters on that line?

3.

What about the letters `F`?

9. This number doesn't depend on the size of diamond, it depends only on the specific letter.

Write a word `CENTRAL` that given a letter number, calculate the number of central spaces to print.

It's easy:

```
: CENTRAL     \ l# -- n
   2* 3 - ;
```

```
gforth Diamond.fs ↩
6 CENTRAL . ↩ 9 ok
5 CENTRAL . ↩ 7 ok
3 CENTRAL . ↩ 3 ok
1 CENTRAL . ↩ -1 ok
```

Of course it doesn't work for the letter `A`.

What should we print when the number of central spaces is -1?

We should print a *backspace*.

What would be the effect of printing a backspace between two letters?

Only one letter would appear on the terminal line!

Create a word `.SPACES` which when given a positive number, acts like exactly like `SPACES`, and when given -1 sends a backspace to the terminal.
The character code for a backspace is 8.

```
: BACKSPACE
    8 EMIT ;

: .SPACES     \ n --
    DUP 0>= IF SPACES
           ELSE DROP BACKSPACE
           THEN ;
```

```
gforth Diamond.fs ↩
42 EMIT 3 .SPACES 42 EMIT CR ↩ *   *
ok
42 EMIT -1 .SPACES 42 EMIT CR ↩ *
ok
```

It works!

Let's write a word `LETTERS` which given a letter number, print the character corresponding to that number, the central spaces, and that character again.

OK!

```
: LETTER      \ l# -- c
    START 1- + EMIT ;

: LETTERS     \ l# --
    DUP LETTER
    DUP CENTRAL .SPACES
        LETTER ;
```

```
gforth Diamond.fs ↩
1 LETTERS ↩ A ok
2 LETTERS ↩ B B ok
3 LETTERS ↩ C   C ok
4 LETTERS ↩ D     D ok
```

It works!

In a diamond using the letters `A` to `F`, on the lines containing `A`, how many spaces do you need to print before printing `A`?

5 spaces.

In a diamond using the letters `A` to `F`, on the lines containing `B`, how many spaces do you need to print before printing `B`?

4 spaces.

In a diamond using the letters `A` to `C`, on the lines containing `C`, how many spaces do you need to print before printing `C`?

zero.

What are your conclusions?

The number of spaces before a letter is a function of that letter number and the maximum letter number.

How would you calculate this number of spaces?

Subtracting the letters numbers. Easy.

Write a word **LINE** which given the maximum letter number and the current letter number, prints the diamond line corresponding to that current letter.

Here you go:

```
: LINE        \ max#,l# --
   DUP -ROT - SPACES LETTERS ;
```

```
gforth Diamond.fs ↩
6 1 LINE ↩       A ok
6 2 LINE ↩     B B ok
6 5 LINE ↩  E       E ok
6 6 LINE ↩ F         F ok
```

It works!

What does our program need to do print the first half of a diamond from letters **A** to **F** ?

Looping from 1 to 6, and calling our word **LINE** with 6 as a first parameter, and the current loop indice as the second parameter.

Write a word **1ST-HALF** which given the maximum letter number prints the first half of the diamond.

```
: 1ST-HALF   \ max# --
   DUP 1+ 1 DO
            DUP I CR LINE
            LOOP DROP ;
```

```
gforth Diamond.fs ↩
6 1ST-HALF ↩
     A
    B B
   C   C
  D     D
 E       E
F         F ok
```

What does our program need to do print the second half of a diamond from letters **A** to **F** ?

Looping from 2 to 6, and calling our word **LINE** with 6 as a first parameter, and the proper letter number as the second parameter.

How is that letter number calculated?

When the loop indice varies from 2 to 6, the letter number should vary from 5 to 1, so subtracting **I** from 7 is the way to obtain the proper letter number.

Write the word **2ND-HALF** which given the maximum letter number prints the sedond half of the diamond.

```
: 2ND-HALF   \ max# --
   DUP 1+ 2 DO
            DUP  DUP 1+ I -  CR LINE
            LOOP DROP ;
```

```
gforth Diamond.fs ↩
6 2ND-HALF ↩
 E       E
  D     D
   C   C
    B B
     A ok
```

Write the word **DIAMOND** which when given a char corresponding to the maximum letter, prints a diamond.

```
: DIAMOND    \ c --
    LETTER#
    DUP 1ST-HALF
        2ND-HALF CR ;
```

```
gforth Diamond.fs ↩
CHAR F DIAMOND ↩
     A
    B B
   C   C
  D     D
 E       E
F         F
 E       E
  D     D
   C   C
    B B
     A
 ok
```

It works!

Here's the program

```
\ Diamond.fs

CHAR A CONSTANT START

: LETTER#     \ c -- l#
    START - 1+ ;

: CENTRAL     \ l# -- n
    2* 3 - ;

: BACKSPACE
    8 EMIT ;

: .SPACES     \ n --
    DUP 0>= IF SPACES
          ELSE DROP BACKSPACE
          THEN ;

: LETTER      \ l# -- c
    START 1- + EMIT ;

: LETTERS     \ l# --
    DUP LETTER
    DUP CENTRAL .SPACES
        LETTER ;

: LINE        \ max#,l# --
    DUP -ROT - SPACES LETTERS ;

: 1ST-HALF    \ max# --
    DUP 1+ 1 DO
              DUP I CR LINE
              LOOP DROP ;

: 2ND-HALF    \ max# --
    DUP 1+ 2 DO
              DUP  DUP 1+ I -  CR LINE
              LOOP DROP ;

: DIAMOND     \ c --
    LETTER#
    DUP 1ST-HALF
        2ND-HALF CR ;
```

Nice!

```
gforth Diamond.fs ↩
CHAR Z DIAMOND ↩
```