By Jay Lorch

The first step is to decipher the cryptogram, figuring out which token stands for which. For the detailed logical steps to do this, see page 2 of this answer document. Here's the resulting cipher:

Ciphertext	Plaintext
b	#include
a	<
argv	stdio.h
d	>
)	int
"%c%c%c%c%c\n"	main
{	(
char	argc
(	,
return	char

Ciphertext	Plaintext
e	*
/	argv
}	)
1	{
=	a
int	b
*	c
main	d
>	e
,	=

Ciphertext	Plaintext
;	1
-	+
c	;
<	-
printf	/
#include	printf
argc	"%c%c%c%c%c\n"
stdio.h	return
+	}

The next step is to decipher the program, producing the following:

```
#include <stdio.h>

int main (int argc, char **argv)
{
    int a, b, c, d, e;

    a = 1 + 1;
    a = a * a * a * a * a * a + a;
    d = a + (-1 / 1);
    c = a + 1 + 1 + 1;
    b = 1 + 1;
    b = b * b * b * b;
    b = a + b;
    e = (-1 + c + b) / (1 + 1);

    printf("%c%c%c%c%c\n", a, b, c, d, e);
    return 1 - 1;
}
```

Finally, running this program prints the answer **BREAK**, which should seem right since it's also a C token.

By Jay Lorch

The cipher can be deduced as follows.

C programs start with #include statements. Since the only file name token is stdio.h, it starts with #include <stdio.h>. Next comes the preamble, consisting of int main (int argc, char **argv).

The indentation and line breaks are standard, allowing us to learn {, }, and ;.

The first statement is int followed by comma-separated tokens, so those tokens must be variables. We can't tell which token is which variable, but it doesn't matter because the program does the same thing no matter how we name the variables. So we might as well just assign the variables so they appear in alphabetical order.

Next, we figure out which is =, and the obvious choice is the token immediately after variables at the beginnings of statements.

The first assignment has a right-hand side of three tokens, the first and last of which are identical. The middle one must be an operator, and the others must be a constant since no variables have been initialized yet. The string constant makes no sense with any operator, so the constant must be the only numeric constant given in the cipher: 1.

Next, we have to learn which operator is which. We can figure out which is - because it is the only one used as a unary operator. We know * from the preamble. The wrong choice of + vs. / leads to a divide-by-zero error, so we know the other choice is correct.

The printf statement, with its format string, is easy to spot now, and then return is the remaining token.