Practice with the third and fourth loopy questions

**Introduction**

The third and fourth loop questions are:

3. Does the repetend make progress toward termination? To see this, we generally give an expression whose value should decrease during execution of the repetend.

4. Does the repetend keep P true: Is \( \{B \&\& P\} S \{P\} \) true?

**Summation**

We develop the repetend of a loop that adds the values in range \( m..n-1 \). Here is the relevant information:

- **P**: \( s = \text{sum of } m..k-1 \text{ and } m \leq k \leq n \)
- **B**: \( k < n \)
- Progress: Decrease the expression \( n - k \)

The way to decrease the expression is to add 1 to \( k \): This means that one more value has to be added to \( s \). Since \( s \) contains the sum of \( m..k-1 \), the next value to add is \( k \). The repetend is:

\[
\{B \&\& P\} \ s = s + k; \ k = k+1; \ \{P\}
\]

Here are two more exercises for you to do. The answers can be found at the end of the script for this video. Please stop the video and do them — be careful.

1. **P**: \( s = \text{sum of } k..n-1 \text{ and } 0 \leq k \leq n \)
   - **B**: \( k > m \)
   - Progress: decrease \( k \)

2. **P**: \( v = \text{minimum of } b[0..k] \) and \( 0 < k \leq n \)
   - **B**: \( k < n \)
   - Progress: increase \( k \)

We developed the repetend in an informal fashion. A later video shows how this repetend can almost be calculated.

**Exponentiation**

We are working on a loop to calculate \( b^c \) (\( b \) to the power \( c \)) for \( c \geq 0 \). Here are the invariant, the loop condition, and our way of getting closer to termination:

- **P**: \( b^c = z \times x^y \) and \( y \geq 0 \)
- **B**: \( 0 < y \)
- Progress: Decrease \( y \)

The simplest \( y \) to decrease \( y \) is to subtract 1 from it. To see how to maintain the invariant when subtracting 1 from \( y \), we rewrite use this property of exponentiation: \( x^y = x \times x^{y-1} \). Therefore,

\[
z \times x^y = z \times x \times x^{y-1}.
\]

Thus, if we subtract 1 from \( y \), we can maintain the invariant by storing \( z \times x \) in \( z \), yielding the repetend:

\[
\{B \&\& P\} \ z = z \times x; \ y = y-1; \ \{P\}
\]

Earlier, we developed the initialization and loop condition, and the final algorithm is shown below. It makes \( c \) iterations. In lecture, we will see how this can be reduced greatly.

```c
x = b; y = c; while (y != 0) {z = z \times x; y = y-1; }
```

**Answer to exercises**

1. To make progress toward termination, use \( k = k-1 \); To keep invariant \( P \) true, a value has to be added to \( s \). We use

\[
s = s + k-1; k = k-1; \quad \text{or} \quad k = k-1; s = s + k;
\]

2. To make progress toward termination, use \( k = k+1 \); For \( P \) to be true after that, we need \( v = \text{minimum of } b[0..k] \). We can use either

\[
v = \text{Math.min}(v, b[k+1]); k = k+1; \quad \text{or} \quad k = k+1; v = \text{Math.min}(v, b[k]);
\]