1. What Does This Program Do? (BASIC)
When the following program is run, what is the final value of B?

```
10 A = 1 - 9 * 9 + 3
20 B = 1 - 9 * (9 + 4)
30 B = A - B
40 END
```

2. What Does This Program Do? (BASIC)
Find the initial value of k for which the program prints “1101,” the binary value of variable X. Assume that k is a power of 2.

```
10 X = 13
20 K = ?
30 REM
40 IF X / K < 1 THEN PRINT "0" ELSE PRINT "1"
50 X = X - K*INT(X/K)
60 K = K / 2
70 IF K >= 1 THEN GOTO 30
80 END
```

3. Computer Number Systems
Express the sum of the hex numbers A23C and 4D5E6 in hex.

4. Computer Number Systems
Express 32411648 in hex.

5. Computer Number Systems
Find the decimal value of A42E16.
1. Line 10 sets \( A = -77 \), line 20 sets \( B = -116 \), and line 30 computes the value of \( A - B \) and stores it into \( B \).

2. Variable \( K \) should be initialize to the smallest power of 2 less than or equal to \( X \). One key to this problem is to note that the first character output is a “1”. For that to be true, \( \frac{X}{K} \geq 1 \) must be false, so \( K \) must be a power of 2 smaller than 13. Another key to the problem is to note that the output string is 4 characters long. Since each time through the loop, a 0 or 1 is printed, the loop must be executed 4 times, and \( K \) must be 8.

3. We could convert all numbers into base 10, add, and then convert back to base 16. This approach would be somewhat tedious, since the numbers are large (the sum is 983,330_{10}). Adding directly in hex is not too difficult:

\[
\begin{align*}
C + 6 &= 18_{10} = 12 \\
\text{carry} + 3 + E &= 18_{10} = 12 \\
\text{carry} + B + 5 &= 17_{10} = 11 \\
\text{carry} + 2 + D &= 16_{10} = 10 \\
\text{carry} + A + 4 &= 15_{10} = F
\end{align*}
\]

4. An easy way to convert between octal and hexadecimal is to go through the binary representation. The strategy is to write \( 3241164_{8} \) in binary:

\[
011 \ 010 \ 100 \ 001 \ 001 \ 110 \ 100
\]

Now, regroup the bits into groups of 4 starting at the right:

\[
1101 \ 0100 \ 0010 \ 0111 \ 0100
\]

Finally, convert each group into a hex digit:

\[
D \ 4 \ 2 \ 7 \ 4.
\]

5. There isn’t any clever solution to this problem. Just “plug and chug”:

\[
\begin{align*}
10 \cdot 16^3 + 4 \cdot 16^2 + 2 \cdot 16^1 + 14 \cdot 16^0 \\
&= 10 \cdot 4096 + 4 \cdot 256 + 2 \cdot 16 + 14 \cdot 1 \\
&= 40,960 + 1,024 + 32 + 14 \\
&= 42,030
\end{align*}
\]
1. What Does This Program Do? (BASIC)
   A program you've been given prints "ACSL" every so often. The only
   PRINT statement in the program appears at line 25:
   
   \[
   \text{25 IF } X \geq 2 * Y \text{ AND } (Y=1-X) \text{ THEN PRINT "ACSL"}
   \]
   
   If you know that \(X\) is an integer, what is the largest value that \(Y\) could
   have when "ACSL" is printed?

2. Recursive Functions
   Evaluate \(f(16)\) given
   \[
   f(x) = \begin{cases} 
   f(x-4) + 1 & \text{if } x \geq 10 \\
   f(x-3) - 3 & \text{if } 2 < x < 10 \\
   x^2 - 1 & \text{otherwise}
   \end{cases}
   \]

3. Recursive Functions
   Evaluate \(f(6)\) given
   \[
   f(x) = \begin{cases} 
   f(x-2) - 2 f((x-1) \% 3) + 1 & \text{when } x \text{ is even and positive} \\
   x + 1 & \text{when } x \text{ is odd} \\
   0 & \text{otherwise}
   \end{cases}
   \]
   The \% is the mod function: the remainder when \(x-1\) is divided by 3.
   For example, \(5\%3=2\) and \(10\%3=1\).

4. Boolean Algebra
   List all ordered triples that satisfy the following expression:
   \[
   \overline{A + B} \oplus \overline{AC}
   \]

5. Boolean Algebra
   Simplify the following expression as much as possible:
   \[
   \overline{AB} + \overline{AB}
   \]
1. The string is printed when two conditions hold: \( x - 5 > 2y \) and \( y = 1 - x \). If you try small values for \( x \), starting at, say 0, you'd find that \( x = 3 \) is the smallest value of \( x \) for which both conditions are true. At this point, \( y = -2 \). As \( x \) increases, both conditions are true, but the value of \( y \) continues to decrease. You can see this graphically by plotting the equations \( x - 5 > 2y \) and \( y = 1 - x \): 

![Graph showing the intersection of two lines](image)

2. The evaluation is as follows:

\[
\begin{align*}
f(16) &= f(12) + 1 \\
&= (f(8) + 1) + 1 = f(8) + 2 \\
&= (f(5) - 3) + 2 = f(5) - 1 \\
&= (f(2) - 3) - 1 = f(2) - 4 \\
&= (2^2 - 1) - 4 = 3 - 4 = -1
\end{align*}
\]

3. The first part of the evaluation is as follows:

\[
\begin{align*}
f(6) &= f(4) - 2f(2) + 1 \\
f(2) &= f(0) - 2f(1) + 1 \\
&= 0 - 2 \cdot 2 + 1 = -3 \\
f(4) &= f(2) - 2f(0) + 1 \\
&= -3 - 2 \cdot 0 + 1 = -2
\end{align*}
\]

We can now complete the evaluation of \( f(6) \).

4. Let's take advantage of the observation that \( x \oplus y \) is equivalent to \( x \oplus y \) to simplify the problem to \( A + B \oplus AC \). This is true when \( A + B \) is different than \( AC \). There are two cases to consider: 

Case 1: \( \text{rhs} = 1 \) ⇒ the lhs must also be 1, since \( A = 1 \) from the rhs. This is not a solution. 

Case 2a: \( \text{rhs} = 0 \) \( A = 0 \) and \( C = 1 \) ⇒ \( B = 1 \). 

Case 2b: \( \text{rhs} = 0 \) \( A = 1 \) and \( C = 1 \) ⇒ no solutions. 

Case 2c: \( \text{rhs} = 0 \) \( A = 1 \) and \( C = 0 \) ⇒ \( B = 1 \). 

\[
\begin{align*}
(0,1,0) & \quad \text{and} \quad (1,0,0) \\
(0,1,1) & \quad \text{and} \quad (1,1,0)
\end{align*}
\]

5. After applying DeMorgan's Law, the expression becomes \( AB \cdot \overline{AB} \). Apply DeMorgan's Law to the right term to get \( AB \cdot (A + B) \). This simplifies to \( ABA + ABB \), which simplifies to \( AB + 0 = AB \). 

\[
AB
\]
1. What Does This Program Do? (Pascal)
   What is printed when the following code is executed?
   ```pascal
   x := 1; y := 9;
   while x < y do
     begin
     y := y + 2;
     ct := 0;
     repeat
     x := x + 1;
     ct := ct + 1;
     until (ct = 5) or (x = y);
     end;
   writeln ('x = ', x, ' and y = ', y);
   ```
   \[ x = \square \text{ and } y = \square \]

2. Bit String Flicking
   Evaluate the following expression:
   \[ \neg (\text{LCIRC-2 01100}) \land (\text{RCIRC-3 (LSHIFT-2 01110)}) \]

3. Bit String Flicking
   Find all values of \(x\), a 5-bit long string, that makes the following equation true:
   \[ (\text{LSHIFT-3 } x) \oplus (\text{RCIRC-2 } x) = \neg x \]

4. Digital Electronics
   Find a Boolean expression that represents the following circuit, and simplify the expression as much as possible.
   ![Digital Electronics Circuit](image)

5. Digital Electronics
   Find all inputs that make the following circuit true.
   ![Digital Electronics Circuit](image)
1. The while loop executes as long as \( x \) is less than \( y \). Each time through the loop, \( y \) is incremented by 2. So, the first time through the loop, \( x = 1 \) and \( y = 11 \) just before the until loop. The until loop increments \( x \) until it equals \( y \), but never more than 5 times. The until loop is called 3 times, the first time with \( x = 1 \) and \( y = 11 \); the second time with \( x = 6 \), \( y = 13 \); and the third time with \( x = 11 \), \( y = 15 \). The third time, the until loop exits because \( x = y \) (the other times it exits because \( ct = 5 \)), and this causes the while loop to also terminate.

\[
x = \boxed{15} \quad \text{and} \quad y = \boxed{15}
\]

2. The evaluation is as follows:
\[
(\text{NOT (LCIRC-2 01110)}) \text{ AND (RCIRC-3 (LSHIFT-2 01110)})
\]
\[
= (\text{NOT 10001}) \text{ AND (RCIRC-3 11000)}
\]
\[
= 01110 \text{ AND 00011}
\]
\[
= 00010
\]

3. Express \( x \) as abdc are simplify:
\[
(\text{LSHIFT-3 abcd}) \text{ XOR (RCIRC-2 abcd)} = (\text{NOT abcd})
\]
\[
de000 \text{ XOR deabc} = (\text{NOT abcd})
\]
\[
\text{NOT(de000 XOR deabc)} = (\text{NOT (NOT abcd)})
\]
\[
de000 \text{ EQ deabc} = abcd
\]
Now, examine each bit: From \( d \) EQ \( d = a \), we know that \( a = 1 \).
Similarly, \( b = 1 \), and the equation is \( de000 \text{ EQ de1c} = 11cde \).
From \( 0 \) EQ \( 1 = c \), we have \( c = 0 \). From \( 0 \) EQ \( 1 = d \), we have \( d = 0 \). And finally, from \( 0 \) EQ \( c = e \), we have \( e = 1 \).

4. The circuit is represented by the Boolean expression
\[
AB \oplus BC
\]
This simplifies as follows:
\[
(\overline{AB})(\overline{BC}) + (\overline{AB})BC
\]
\[
(AB)(B + C) + (A + B)BC
\]
\[
AB\overline{B} + ABC + \overline{ABC} + \overline{B}BC
\]
\[
\overline{AB}C + \overline{ABC}
\]
\[
B(\overline{A}C + \overline{AC})
\]
\[
B(\overline{A} \oplus C)
\]
The operands of the AND and the XOR can be commuted.

5. The circuit is represented by the Boolean expression
\[
A(A + C) + (C + B) \oplus B
\]
The following truth table shows the 3 triples that make the expression true:

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
<th>( A + C )</th>
<th>Let ( x = A(A + C) )</th>
<th>( B + C )</th>
<th>Let ( y = B \oplus (B + C) )</th>
<th>( x \oplus y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\((0, 0, 0), (0, 1, 0), \) and \((0, 1, 1)\)
1. What Does This Program Do? (Pascal)
When the following program is run, what are the final values of \(x\) and \(y\) after the third call to \(\text{senior}\) returns?

```pascal
program tricky;
procedure senior (a: integer; var b: integer);
begin
  a := a + b;
  b := b + a;
end;
procedure main;
  var x, y: integer;
  begin
    x := 1; y := 1;
    senior(x, y);
    senior(y, x);
    senior(x, y);
  end;
begin main end.
```

2. What Does This Program Do?
Evaluate the following postfix expression:

\[ 8 2 3 * + 2 4 - / \]

3. Prefix/Infix/Postfix Notation
Convert the following prefix expression into postfix.

\[ / + * 2 \uparrow A 3 6 + 4 B \]

4. Data Structures
What is the maximum depth of a binary tree containing 6 nodes, and at least 1 node contains 2 children?

5. Data Structures
Insert the letters B I L L C L I N T O N into an initially empty binary search tree, starting with the B and ending with the final N. How many internal nodes have 2 children?
1. There are three tricky parts to this question: First, in procedure `senior`, the values assigned to `a` and `b` are not the same. Second, the second argument to `senior` is passed by `var`. Third, the parameters to the second call to `senior` are swapped compared to the first and third calls.

\[ x = 5 \]
\[ y = 11 \]

2. The expression converts into infix as follows (a subexpression is boxed after it has been converted and evaluated):
\[ 8 [0 + -2] / \Rightarrow [14 -2] / \Rightarrow -7 \]

3. The easiest approach to this problem is to convert the postfix into infix, and then the infix into prefix. The infix version is:
\[ \frac{2A^3 + 6}{4 + B} \]

4. The node with 2 children should be as low as possible:

```
        o
       /|
      / |\n     o  o
```

5. The resulting tree looks like:

```
      o
     /|
    / |\n   B  I  
  /\      \n C  L    O
 /\    /\   /\   /\   /\   /\
I  L  N  T
```