1. What Does This Program Do? (BASIC)

When the following program is run, how many different values does variable A take on? The initial values of both A and C is 15.

```
for B = 10 to 15 step 2
    if A < B then A = B else C = A - 2
    if B < C then C = B else A = 2*B - C
next B
```

2. Computer Number Systems

Convert 1994 from hex to octal.

3. Computer Number Systems

What is the minimum number of bits that are needed to represent the numbers 0 through 4, 194, 303 (1024 * 1024 * 4 - 1) in binary? For example, 4 bits are needed to represent the numbers 0 through 12 in binary.

4. Recursive Functions

Evaluate $f(10, 6)$ where

$$f(x, y) = \begin{cases} 
  f(x-1, y) & \text{if } x > 7 \\
  f(y-2, x) + 4 & \text{if } 1 \leq x \leq 7 \\
  -10 & \text{otherwise}
\end{cases}$$

5. Recursive Functions

Suppose that $f(3) = 16$ where

$$f(x) = \begin{cases} 
  f(x-2) + 3f(x-1) - 5 & \text{if } x > 0 \\
  k & \text{if } x \leq 0
\end{cases}$$

Find all valid integer values of $k$. 
1. The values that \( A \) takes on are (in order) 15, 7, 12, 11, and 14. The following table gives the values of \( A \) and \( C \) after the if statement each time through the loop:

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>15</td>
<td>13</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>13</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>13</td>
<td>14</td>
<td>14</td>
</tr>
</tbody>
</table>

2. An easy way to convert between hexadecimal and octal is to go through the binary representation. The strategy is to write \( 1994_{16} \) in binary:

\[
0001 1001 1001 0100
\]

regroup the bits into groups of 3, starting at the right:

\[
0 001 100 110 010 100
\]

and finally, convert each group into an octal digit:

\[
14624
\]

3. In general, \( n \) bits specify \( 2^n \) different values. The numbers 0 through 4,194,303 contain 4,194,304 different values, and

\[
4,194,304 = 1024 \cdot 1024 \cdot 4 = 2^{10} \cdot 2^{10} \cdot 2^2 = 2^{22},
\]

so 22 bits are needed.

4. The evaluation is as follows:

\[
\begin{align*}
f(10, 6) &= f(9, 6) = f(8, 6) = f(7, 6) \\
f(7, 6) &= f(4, 7) + 4 \\
f(4, 7) &= f(5, 4) + 4 \\
f(5, 4) &= f(2, 5) + 4 \\
f(2, 5) &= f(3, 2) + 4 \\
f(3, 2) &= f(0, 3) + 4 \\
f(0, 3) &= -10
\end{align*}
\]

Back-substituting is straightforward.

5. Let’s evaluate \( f(3) \) as a function of \( k \):

\[
\begin{align*}
f(3) &= f(1) + 3f(2) - 5 \\
f(2) &= f(0) + 3f(1) - 5 = k + 3f(1) - 5 \\
f(1) &= f(-1) + 3f(0) - 5 = k + 3k - 5 = 4k - 5 \\
f(2) &= \ldots = k + 3(4k - 5) - 5 = 13k - 20 \\
f(3) &= \ldots = (4k - 5) + 3(13k - 20) - 5 = 43k - 70
\end{align*}
\]

We can now solve for \( k \), since we know that

\[
f(3) = 43k - 70 = 16.
\]
1. What Does This Program Do? (BASIC)
   After the following program is run, what is the value of c$?

```basic
  x$ = "abcdefghijklmnoprstuvwxyz"
  for x = 1 to 10
    a$ = mid$(x$, 20-2*x+3, 3)
    b$ = b$ + left$(right$(a$,3),2)
  next x
  c$ = mid$(b$, 10, 2)
```

2. Boolean Algebra
   Suppose that A and C were guaranteed to have the same value. That is, either both are true or both are false. With that assumption, simplify the following expression as much as possible.

   \[
   (A + BC)(A(BC))
   \]

3. Boolean Algebra
   List all ordered triples that make the following expression false.

   \[
   (A + B)C \oplus BC \oplus A + C
   \]

4. Bit String Flicking
   Evaluate the following expression:

   \[
   (\text{LCIRC-2} ((\text{RCIRC-2} 10111) \text{ AND } (\text{LSHIFT-2} 00011)))
   \]

5. Bit String Flicking
   Find all values of x, a 5-bit long string, that make the following equation true:

   \[
   (\text{RSHIFT-1} (\text{LCIRC-3} (\text{NOT} \ x))) = (\text{RCIRC-2} (\text{LSHIFT-3} 11010))
   \]
1. After the loop finishes, the value of b is uvstqropmnkljghfcd.
The variable c is set to the 10th and 11th character of b.

2. Ignoring the assumption, the expression simplifies as follows:

\[
\frac{(A + BC)(\overline{A}BC)}{(A + BC) + (A\overline{BC})} = (A + BC) + (\overline{A}\overline{BC}) \\
= (A \cdot BC) + ABC \\
= B \overline{AC} + AC \\
= B(\overline{A} + C)
\]

When A and C are the same, \( A \oplus C \) is false, and \( \overline{A} \oplus C \) is always true. Thus, the expression becomes simply \( B \).

3. First, consider the expression \( x \oplus y \oplus z \). This is false when an even number of inputs are true. (Take a moment and convince yourself of this fact.) The truth table below shows the values of each argument to the XOR operator in the original expression; the last column counts the number of arguments that are true.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>((A + B)C)</th>
<th>(BC)</th>
<th>(\overline{A} + C)</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\((0,0,0), (0,1,0), and (1,0,1)\)

4. The evaluation is as follows:

\[
(LCIRC-2 ((RCIRC-2 10111) AND (LSHIFT-2 00011))) \\
= (LCIRC-2 (11101 AND 01100)) \\
= (LCIRC-2 01100) \\
= 10001
\]

5. The right side evaluates to 00100. Now, express \( x \) as \( abcde \) and the NOT of each bit as the capitalized letter, and simplify the left side:

\[
(RSHIFT-1 (LCIRC-3 (NOT abcde))) \\
(RSHIFT-1 (LCIRC-3 ABCDE)) \\
(RSHIFT-1 DEABC)
\]

11010 and 11110

Now, by examining each bit in the two sides, we see that \( D=0, E=1, A=0, \) and \( B=0 \). Equivalently, we have \( d=1, e=0, a=1, \) and \( b=1 \). Bit \( c \) can take on any value.
1. What Does This Program Do? (Pascal)
When procedure whome is called, what is the value of variable \( y \) just before the procedure returns?

```pascal
procedure whome;
    var x, y, z: integer;
    begin
        y := 0; x := 10;
        repeat
            x := x-1; y := y+1; z := 8;
            while (z>5) and (x>=z) do
                begin z := z-1; y := y+1 end
        until x=4
    end;
```

![Diagram](image)

<table>
<thead>
<tr>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Box )</td>
</tr>
</tbody>
</table>

2. Regular Expressions and FSAs
If a string generated by the regular expression below contains 6 c’s and 15 a’s, how many b’s are in that string?

\(((ab)ac)^*\)

3. Regular Expressions and FSAs
What is the length of the shortest string containing at least one E that is accepted by the following FSA?

![Diagram](image)

4. Digital Electronics
Suppose that you had only OR, AND, and NOT gates, and that each one costs 5 cents. What is the cost of the cheapest circuit you could build that is true for the exact same inputs as the following circuit?

![Diagram](image)

5. Digital Electronics
Find all inputs that make the following circuit false.

![Diagram](image)
1. This program consists of a while loop within a repeat-until loop. The repeat-until loop is controlled by the variable \( x \), and \( x \) starts at 10 and is decremented once each time through the loop. Thus, the while loop is executed 6 times, with \( x = 9 \) down to 4. When \( x = 9 \) and 8, the body of the while loop does not execute at all.

\[ y = 12 \]

2. The strings generated by the regular expression have the property that every \( a \) is followed by either a \( b \) or a \( c \). Moreover, every \( b \) and \( c \) is preceded by exactly one \( a \). Thus, if there are a total of 15 \( a \)'s, and 6 \( c \)'s, this means that 6 of the \( a \)'s are preceded by a \( c \), and the other 9 \( a \)'s must be preceded by a \( b \).

\[ 9 \]

3. There are two such strings: \textbf{BEDA} and \textbf{BDEC}.

\[ 4 \]

4. The circuit is represented by the Boolean expression

\[( (AB) + C ) \bar{A} \]

This simplifies as follows:

\[ (AB + C)\bar{A} = AB\bar{A} + C\bar{A} = CA \]

This circuit costs 10 cents to realize: 5 cents for the NOT (of \( A \)) and another 5 cents for the AND (of \( A \) and \( C \)).

\[ 10 \text{ cents} \]

5. The circuit is represented by the rather complicated Boolean expression

\[ A( (AB)(BC) ) C \]

When this is false, the following is true:

\[ A( (AB)(BC) ) C \]

Clearly, \( C = 1 \) and \( A = 1 \). Substitute and simplify as follows:

\[ (1((1B)(\bar{B}))1) = (B)(B) = \bar{u} = 1 \]

Thus, it doesn’t matter what value \( B \) has.
1. What Does This Program Do? (Pascal)
   When the following program is run, what are the final values of \( x \) and \( y \) after the second call to `senior` returns?

   ```pascal
   program foo;
   var x, y: integer;
   procedure senior(var a: integer; b: integer);
   begin
     a := b - a;
     b := b + 2*a;
   end;
   begin
     x := 5; y := 8;
     senior(x, y);
     senior(y, x);
   end.
   ```

   \( x = \boxed{\_\_\_} \)
   \( y = \boxed{\_\_\_} \)

2. Prefix/Infix/Postfix Notation
   Evaluate the following postfix expression when \( A=8, B=6, C=7, \) and \( D=2 \).

   \[ A B + C / A D * * B + \]

3. Prefix/Infix/Postfix Notation
   For this problem, we define two new binary operators. 
   \# is the least common multiple and \% is the greatest common factor.
   For example, \((4 \# 6) \% 8\) has a value of 4, since \(4 \# 6\) is 12 and \(12 \% 8\) is 4.
   Evaluate the following prefix expression.

   \[ # \% # 2 3 # 4 5 # \% 8 12 \% 6 3 \]

4. Data Structures
   Consider the following sequence of operations on an empty stack:

   \[ H O + U S + O + N + + T X + \]

   A plus sign indicates a pop operation, and a letter indicates a push operation of that letter. What would be the next item to be popped?

5. Data Structures
   Insert the letters `S A L T I A K E C I T Y` into an initially empty binary search tree, starting with the `S` and ending with the `Y`. How many internal nodes have 1 child?
1. There are two tricky parts to this question: One, the first argument to `senior` is passed by `var`. Two, the parameters to the second call to `senior` are swapped compared to the first call. After the first call, the value of `x` is changed to 3; after the second call, the value of `y` becomes -5.

   $x = 3$
   $y = -5$

2. The conversion to infix is as follows (an expression is boxed after it has been converted):

   \[
   \frac{A + B}{C} + \frac{A D}{A D} + B + \frac{A + B}{C} A D + B
   \]

   Now, substitute the value of the variables and evaluate:

   \[
   \left( \frac{A + B}{C} \right) A D + B = \frac{8 + 6}{7} \cdot 8 \cdot 2 + 6 = 2 \cdot 8 + 6 = 38.
   \]

3. The evaluation is as follows; an expression is boxed after it has been evaluated:

   $2 \ 3 \ 4 \ 5 \ 8 \ 12 \ 6 \ 3$
   $6 \ 20 \ 4 \ 3$
   $2 \ 12$

4. Remember that a stack is “last-in, first-out.” That is, the most recent item pushed will be the next one removed. Thus, the items popped are O, S, 0, N, U, and X in this order. The stack contains two elements, H at the bottom, and T above that.

5. The three nodes with only 1 child are the two L’s and the K. Here’s what the resulting tree looks like: