C5 Solutions

1. Convert the following base 10 numbers into 8-bit 2’s complement notation
   0, -1, -12

   To Compute 0

   0 = 00000000

   To Compute –1

   Step 1. Convert 1 to binary
   00000001

   Step 2. Flip the bits
   11111110

   Step3. Add 1
   11111111

   Therefore **–1 = 11111111**

   To Compute –12

   Step 1. Convert 12 to binary
   00001100

   Step 2. Flip the bits
   11110011

   Step3. Add 1
   11110100

   Therefore **–12 = 11110100**

2. Perform each of the following additions assuming that the bit strings represent values in 2’s complement notation. Identify the cases in which the answer is incorrect because of overflow.

$$\begin{array}{lr} & 1111 \\ + & 1111 \\ \hline & 11110 \end{array}$$

Answer = 11110
Overflow = 0
$\therefore$ Answer is correct

$$\begin{array}{lr} & 01111 \\ + & 10001 \\ \hline & 100000 \end{array}$$

Answer = 00000
Overflow = 1
$\therefore$ Answer is incorrect

$$\begin{array}{lr} & 01110 \\ + & 01010 \\ \hline & 11000 \end{array}$$

Answer = 11000
Overflow = 0
$\therefore$ Answer is correct

3. Write an algorithm to convert a negative decimal number into a binary number in 2’s complement form. Assume that the number ranges from +127 to -128

   1. If the number is less than 0
      a. Multiply by –1
      b. Flip the bits by ‘number XOR 0xff’
      c. Add 1 to the result

   2. Convert the number into binary

   Hint: You already know how to convert a positive decimal number into binary notation. Think about determining sign and inverting bit positions.

4. Implement your algorithm in Ada95. Turn in an electronic copy of your code listing and a hard copy of your code.

```
GNAT 3.13p  (20000509) Copyright 1992-2000 Free Software Foundation, Inc.

Compiling: c:/docume~2/jk/desktop/16070/codeso~1/decimal_to_binary.adb (source file time stamp: 2003-09-17 11:09:18)

with Ada.Text_Io;
use Ada.Text_Io;

with Ada.Integer_Text_Io;
use Ada.Integer_Text_Io;

procedure Decimal_To_Binary is

   -- bit-wise operations are only defined for modular types
   type byte is mod 256;

   Number_To_Convert : integer;
   Place_Holder: Byte;

   Binary_Number : String (1..8);
   Count : Integer :=8;


begin
   -- set the string to all zeroes
   Binary_Number :="00000000";

   -- get the number to be converted
   Put("Please enter an integer :");
   Get(Number_To_Convert);

   -- check if the number is negative. If it is,
   -- convert it into positive
   if Number_To_Convert < 0 then

      Number_To_Convert := -1 * Number_To_Convert;

      -- convert to modular type
      Place_Holder := Byte'Val(Integer'Pos(Number_To_Convert));

      -- flip the bits
      Place_Holder := Place_Holder xor 2#11111111#;
      -- add 1
      Place_Holder := Place_Holder + 2#1#;
      -- reconvert to integer
      Number_To_Convert := Integer'Val(Byte'Pos(Place_Holder));

   end if;

   -- decimal to binary conversion
   -- fill in the bit pattern from left to right
   loop
      exit when Count = 0;
      -- if the remainder is non-zero, the bit is set to 1
      -- else the bit is 0
      if (Number_To_Convert mod 2) = 1 then
         Binary_Number(Count) :='1';
      else
         Binary_Number(Count) :='0';
      end if;

      Count := Count -1;
      Number_To_Convert := Number_To_Convert/2;

   end loop;

   Put(Binary_Number);

end Decimal_To_Binary;



67 lines: No errors
```