Problem 1. (16 points):

Condider the following assembly code for a C `for` loop:

```
loop:
        pushl %ebp
        movl  %esp,%ebp
        pushl %esi
        pushl %ebx
        movl  8(%ebp),%esi
        xorl  %ecx,%ecx
        subl  %ebx,%ebx
        movl  $1,%edx
.L1     leal  (%edx,%ebx,1),%eax
        movl  %edx,%ebx
        movl  %eax,%edx
        incl  %ecx
        cmpl  %esi,%ecx
        jl .L1
        movl  %ebx,%eax
        popl  %ebx
        popl  %esi
        leave
        ret
```

Based on the assembly code above, fill in the blanks below in its corresponding C source code. (Note: besides constants, you may only use the symbolic variables `i`, `n`, `x`, `y` and `result` in your expressions below — *do not use register names.*) Each correct response is worth 2 points.

```
int loop(int n)
{
   int i = _______;   int result = _______;   int x = _______;

   do {
      int y = ____________________;

      ____________________;

      ____________________;

      ____________________;

   }  while ( _______________ );

   return result;
}
```

Problem 2. (16 points):

The following procedure takes a single-precision floating point number in IEEE format and prints out information about what category of number it is. Fill in the missing code so that it performs this classification correctly. Each correct entry is worth 3 points.

```
void classify_float(float f)
{
   /* Unsigned value  u has same bit pattern as  f */
   unsigned u = *(unsigned *) &f;
   /* Split  u into the different parts */
   int sign = (u >> 31) & 0x1;      // The sign bit

   int exp  = (u >> 23) & 0xFF;      // The exponent field

   int frac = (u & 0x7FFFFF);      // The fraction field

   /* The remaining expressions can be written in terms of the
   values of  sign,  exp, and  frac */

   if (_____________________)
     printf("Plus or minus zero\");

   else if (_____________________)
     printf("Nonzero, denormalized\");

   else if (_____________________)
     printf("Plus or minus infinity\");

   else if (_____________________)
     printf("NaN\");

   else if (_____________________)
     printf("Greater than -1.0 and less than 1.0\");

   else if (_____________________)
     printf("Less than or equal to -1.0\");
   else
     printf("Greater than or equal to 1.0\");
}
```

Answers:

Prob 1 Taken from the textbook page 159.

```
int loop(int n)
{
   int i = 0;
   int result = 0;
   int x = 1;

   do {
      int y = result + x;

      result = x;

      x = y;

      i++;

   }  while ( i < n );

   return result;
 (from p159... a fib function)
```

Prob 2: The answers are given in order of the slots in the program.

```
(u >> 23) & 0xFF
 u & 0x7FFFFF
(exp == 0 && frac == 0)
(exp == 0)
(exp == 0xFF && frac == 0)
(exp == 0xFF)
(exp <= 126)
(sign == 1)
```