Untitled-1

```
/*
 * floatFloat2Int - Return bit-level equivalent of expression (int) f
 *   for floating point argument f.
 *   Argument is passed as unsigned int, but
 *   it is to be interpreted as the bit-level representation of a
 *   single-precision floating point value.
 *   Anything out of range (including NaN and infinity) should return
 *   0x80000000u.
 *   Legal ops: Any integer/unsigned operations incl. ||, &&. also if, while
 *   Max ops: 30
 *   Rating: 4
 */
int floatFloat2Int(unsigned uf) {

  // extract the sign and exp from uf
  unsigned sign = (uf >> 31);
  unsigned exp = (~(0x1 << 31) & uf) >> 23;

  // 0xF = 1111
  // since the mantissa is 23 bits long, we need 5 0xF and a 0x7 (0111)
  // Mantissa mask => 0x7FFFFF;
  unsigned frac = (uf & 0x7FFFFF);

  int E = exp - 127;

  // if the exponent is less that one, we are working with a denomalized
  // float (fractional value), this automatically rounds to 0 so we
  // just return 0 because we are converting to int (losing information)
  if (E < 0)  return 0;

  // inf & NaN base case
  if (exp == 0xFF || E >= 31)  return 0x80000000u;
  else {
    // When converting a 32-bit float to an integer using bitwise operations, we need to
extract the integer value of the float by multiplying the mantissa by 2 raised to the
power of the exponent. However, the mantissa only contains 23 bits, so if the exponent is
larger than 23, we need to shift the mantissa left by the difference between the exponent
and 23 to get the full integer value.

    // In the IEEE 754 binary representation of the float, the leading bit of the
mantissa field is always assumed to be 1. Therefore, we can add this bit back to the
mantissa by OR-ing it with the value 1 shifted to the left by 23 bits, which sets the
24th bit of the mantissa to 1.
    frac = frac | (1 << 23);
    if (E <= 23)  frac >>= (23 - E);
    else  frac <<= (E - 23);
  }

  // return the correct value according to the sign bit
  // because the mantissa does not
  if (sign) return -frac;
  else  return frac;
}
```