C-1.7

If we express i as the sum of powers of 2, for example, consider the case $i = 11$. We have

$$i = 11_{10} = 1011_2 = 1 + 2 + 8 = 2^0 + 2^1 + 2^3.$$

We observe that in this case,

$$2^i = 2^{(2^0+2^1+2^3)} = (2^{2^0}) \cdot (2^{2^1}) \cdot (2^{2^3}).$$

Our program maintains two variables, repeatedSquare, which holds the value 2^{2^j} and partialProduct, which holds the value of the above product. To determine whether the binary expansion of i contains a 1 bit at a certain position, we perform the bitwise-or of this number with 1, and then shift right by one position. If the extracted bit is 1, then we augment the partial product by multiplying with the current repeated square value, and in any case, we square the repeated square value.

```
long twoToThe( int i ) {
  long partialProduct = 1;
  long repeatedSquare = 2;

  while( i != 0 ) {
    int bit = (i & 1);
    i = i >> 1;
    if( bit == 1 ) {
      partialProduct *= repeatedSquare;
    }
    repeatedSquare *= repeatedSquare;
  }
  return partialProduct;
}
```