```
class Math {
  static Array bit_masks;

  /** Initializes the library. */
  function void init() {
    let bit_masks = Array.new(16);
    let bit_masks[0]  = 1;
    let bit_masks[1]  = 2;
    let bit_masks[2]  = 4;
    let bit_masks[3]  = 8;
    let bit_masks[4]  = 16;
    let bit_masks[5]  = 32;
    let bit_masks[6]  = 64;
    let bit_masks[7]  = 128;
    let bit_masks[8]  = 256;
    let bit_masks[9]  = 512;
    let bit_masks[10] = 1024;
    let bit_masks[11] = 2048;
    let bit_masks[12] = 4096;
    let bit_masks[13] = 8192;
    let bit_masks[14] = 16384;
    let bit_masks[15] = 16384 + 16384;

    // create sqrt mask lut
    return;
  }

  /** Returns the absolute value of x. */
  function int abs(int x) {
    if (x < 0) {
      let x = -x;
    }

    return x;
  }

  /** Returns the product of x and y. */
  function int multiply(int x, int y) {
    var int sum, shifted_x, j;

    if (x < 0) {
      return -Math.multiply(-x, y);
    }

    if (y < 0) {
      return -Math.multiply(x, -y);
    }

    let j = 0;
    let sum = 0;
```

```
    let sum = 0;
    let shifted_x = x;

    while (j < 16) {
      if (bit_masks[j] & y) {
        let sum = sum + shifted_x;
      }

      let shifted_x = shifted_x + shifted_x;

      let j = j + 1;
    }

    return sum;
  }

  /** Returns the integer part of x/y. */
  function int divide(int x, int y) {
    var int q;

    if (y = 0) {
      do Sys.error(3); // Math.divide: Division by zero
    }

    if (x < 0) {
      return -Math.divide(-x, y);
    }

    if (y < 0) {
      return -Math.divide(x, -y);
    }

    if (y > x) {
      return 0;
    }

    let q = Math.divide(x, y + y);

    if ((x - (2 * q * y)) < y) {
      return q + q;
    } else {
      return q + q + 1;
    }
  }

  /** Returns the integer part of the square root of x. */
  function int sqrt(int x) {
    var int test, test_square;

    if (x < 0) {
```

```
      do Sys.error(4); // Math.sqrt: Cannot compute square root of a negative
number
    }

    let test = 1;

    while (test < 181) {
      let test_square = test * test;

      if (test_square > x) {
        return test - 1;
      }

      let test = test + 1;
    }

    // start comparison at 128
    // use 8 masks

    // square x
    // mask on first bit of tester
    // if tester > x
    //    unmask it
    // else
    //    leave it
    // end

    // return tester

    return 181;
  }

  /** Returns the greater number. */
  function int max(int a, int b) {
    if (a > b) {
      return a;
    }

    return b;
  }

  /** Returns the smaller number. */
  function int min(int a, int b) {
    if (a < b) {
      return a;
    }

    return b;
  }
}
```

```
149
```