

```
1 class Math {
2   static Array bit_masks;
3
4   /** Initializes the library. */
5   function void init() {
6     let bit_masks = Array.new(16);
7     let bit_masks[0] = 1;
8     let bit_masks[1] = 2;
9     let bit_masks[2] = 4;
10    let bit_masks[3] = 8;
11    let bit_masks[4] = 16;
12    let bit_masks[5] = 32;
13    let bit_masks[6] = 64;
14    let bit_masks[7] = 128;
15    let bit_masks[8] = 256;
16    let bit_masks[9] = 512;
17    let bit_masks[10] = 1024;
18    let bit_masks[11] = 2048;
19    let bit_masks[12] = 4096;
20    let bit_masks[13] = 8192;
21    let bit_masks[14] = 16384;
22    let bit_masks[15] = 16384 + 16384;
23
24    // create sqrt mask lut
25    return;
26  }
27
28  /** Returns the absolute value of x. */
29  function int abs(int x) {
30    if (x < 0) {
31      let x = -x;
32    }
33
34    return x;
35  }
36
37  /** Returns the product of x and y. */
38  function int multiply(int x, int y) {
39    var int sum, shifted_x, j;
40
41    if (x < 0) {
42      return -Math.multiply(-x, y);
43    }
44
45    if (y < 0) {
46      return -Math.multiply(x, -y);
47    }
48
49    let j = 0;
50    let sum = 0;
```

```
50     let sum = 0;
51     let shifted_x = x;
52
53     while (j < 16) {
54         if (bit_masks[j] & y) {
55             let sum = sum + shifted_x;
56         }
57
58         let shifted_x = shifted_x + shifted_x;
59
60         let j = j + 1;
61     }
62
63     return sum;
64 }
65
66 /** Returns the integer part of x/y. */
67 function int divide(int x, int y) {
68     var int q;
69
70     if (y = 0) {
71         do Sys.error(3); // Math.divide: Division by zero
72     }
73
74     if (x < 0) {
75         return -Math.divide(-x, y);
76     }
77
78     if (y < 0) {
79         return -Math.divide(x, -y);
80     }
81
82     if (y > x) {
83         return 0;
84     }
85
86     let q = Math.divide(x, y + y);
87
88     if ((x - (2 * q * y)) < y) {
89         return q + q;
90     } else {
91         return q + q + 1;
92     }
93 }
94
95 /** Returns the integer part of the square root of x. */
96 function int sqrt(int x) {
97     var int test, test_square;
98
99     if (x < 0) {
```

```
100     do Sys.error(4); // Math.sqrt: Cannot compute square root of a negative
number
101     }
102
103     let test = 1;
104
105     while (test < 181) {
106         let test_square = test * test;
107
108         if (test_square > x) {
109             return test - 1;
110         }
111
112         let test = test + 1;
113     }
114
115     // start comparison at 128
116     // use 8 masks
117
118     // square x
119     // mask on first bit of tester
120     // if tester > x
121     //     unmask it
122     // else
123     //     leave it
124     // end
125
126     // return tester
127
128     return 181;
129 }
130
131 /** Returns the greater number. */
132 function int max(int a, int b) {
133     if (a > b) {
134         return a;
135     }
136
137     return b;
138 }
139
140 /** Returns the smaller number. */
141 function int min(int a, int b) {
142     if (a < b) {
143         return a;
144     }
145
146     return b;
147 }
148 }
```

149 |