```
class Output {
  // Character map for printing on the left of a screen word
  static Array charMaps;
  static Array cursor, col_masks;

  /** Initializes the screen and locates the cursor at the screen's top-left.
*/
  function void init() {
    do Output.initMap();

    let cursor = Array.new(2);
    let cursor[0] = 0;
    let cursor[1] = 0;

    let col_masks = Array.new(8);
    let col_masks[0] = 1;
    let col_masks[1] = 2;
    let col_masks[2] = 4;
    let col_masks[3] = 8;
    let col_masks[4] = 16;
    let col_masks[5] = 32;
    let col_masks[6] = 64;
    let col_masks[7] = 128;

    return;
  }

  // Initalizes the character map array
  function void initMap() {
    var int i;

    let charMaps = Array.new(127);

    // black square (used for non printable characters)
    do Output.create(0,63,63,63,63,63,63,63,63,63,0,0);

    // Assigns the bitmap for each character in the charachter set.
    do Output.create(32,0,0,0,0,0,0,0,0,0,0,0);          //
    do Output.create(33,12,30,30,30,12,12,0,12,12,0,0);  // !
    do Output.create(34,54,54,20,0,0,0,0,0,0,0,0);       // "
    do Output.create(35,0,18,18,63,18,18,63,18,18,0,0);  // #
    do Output.create(36,12,30,51,3,30,48,51,30,12,12,0); // $
    do Output.create(37,0,0,35,51,24,12,6,51,49,0,0);    // %
    do Output.create(38,12,30,30,12,54,27,27,27,54,0,0); // &
    do Output.create(39,12,12,6,0,0,0,0,0,0,0,0);        // '
    do Output.create(40,24,12,6,6,6,6,6,12,24,0,0);      // (
    do Output.create(41,6,12,24,24,24,24,24,12,6,0,0);   // )
    do Output.create(42,0,0,0,51,30,63,30,51,0,0,0);     // *
    do Output.create(43,0,0,0,12,12,63,12,12,0,0,0);     // +
    do Output.create(44,0,0,0,0,0,0,0,12,12,6,0);        //
    do Output.create(44,0,0,0,0,0,0,0,12,12,6,0);        // ,
    do Output.create(45,0,0,0,0,0,63,0,0,0,0,0);         // -
    do Output.create(46,0,0,0,0,0,0,0,12,12,0,0);        // .
    do Output.create(47,0,0,32,48,24,12,6,3,1,0,0);      // /

    do Output.create(48,12,30,51,51,51,51,51,30,12,0,0); // 0
    do Output.create(49,12,14,15,12,12,12,12,12,63,0,0); // 1
    do Output.create(50,30,51,48,24,12,6,3,51,63,0,0);   // 2
    do Output.create(51,30,51,48,48,28,48,48,51,30,0,0); // 3
    do Output.create(52,16,24,28,26,25,63,24,24,60,0,0); // 4
    do Output.create(53,63,3,3,31,48,48,48,51,30,0,0);   // 5
    do Output.create(54,28,6,3,3,31,51,51,51,30,0,0);    // 6
    do Output.create(55,63,49,48,48,24,12,12,12,12,0,0); // 7
    do Output.create(56,30,51,51,51,30,51,51,51,30,0,0); // 8
    do Output.create(57,30,51,51,51,62,48,48,24,14,0,0); // 9

    do Output.create(58,0,0,12,12,0,0,12,12,0,0,0);      // :
    do Output.create(59,0,0,12,12,0,0,12,12,6,0,0);      // ;
    do Output.create(60,0,0,24,12,6,3,6,12,24,0,0);      // <
    do Output.create(61,0,0,0,63,0,0,63,0,0,0,0);        // =
    do Output.create(62,0,0,3,6,12,24,12,6,3,0,0);       // >
    do Output.create(64,30,51,51,59,59,59,27,3,30,0,0);  // @
    do Output.create(63,30,51,51,24,12,12,0,12,12,0,0);  // ?

    do Output.create(65,12,30,51,51,63,51,51,51,51,0,0); // A
    do Output.create(66,31,51,51,51,31,51,51,51,31,0,0); // B
    do Output.create(67,28,54,35,3,3,3,35,54,28,0,0);    // C
    do Output.create(68,15,27,51,51,51,51,51,27,15,0,0); // D
    do Output.create(69,63,51,35,11,15,11,35,51,63,0,0); // E
    do Output.create(70,63,51,35,11,15,11,3,3,3,0,0);    // F
    do Output.create(71,28,54,35,3,59,51,51,54,44,0,0);  // G
    do Output.create(72,51,51,51,51,63,51,51,51,51,0,0); // H
    do Output.create(73,30,12,12,12,12,12,12,12,30,0,0); // I
    do Output.create(74,60,24,24,24,24,24,27,27,14,0,0); // J
    do Output.create(75,51,51,51,27,15,27,51,51,51,0,0); // K
    do Output.create(76,3,3,3,3,3,3,35,51,63,0,0);       // L
    do Output.create(77,33,51,63,63,51,51,51,51,51,0,0); // M
    do Output.create(78,51,51,55,55,63,59,59,51,51,0,0); // N
    do Output.create(79,30,51,51,51,51,51,51,51,30,0,0); // O
    do Output.create(80,31,51,51,51,31,3,3,3,3,0,0);     // P
    do Output.create(81,30,51,51,51,51,51,63,59,30,48,0);// Q
    do Output.create(82,31,51,51,51,31,27,51,51,51,0,0); // R
    do Output.create(83,30,51,51,6,28,48,51,51,30,0,0);  // S
    do Output.create(84,63,63,45,12,12,12,12,12,30,0,0); // T
    do Output.create(85,51,51,51,51,51,51,51,51,30,0,0); // U
    do Output.create(86,51,51,51,51,51,30,30,12,12,0,0); // V
    do Output.create(87,51,51,51,51,51,63,63,63,18,0,0); // W
    do Output.create(88,51,51,30,30,12,30,30,51,51,0,0); // X
    do Output.create(89,51,51,51,51,30,12,12,12,30,0,0); // Y
    do Output.create(90,63,51,49,24,12,6,35,51,63,0,0);  // Z

    do Output.create(91,30,6,6,6,6,6,6,6,30,0,0);          // [
    do Output.create(92,0,0,1,3,6,12,24,48,32,0,0);        // \
    do Output.create(93,30,24,24,24,24,24,24,24,30,0,0);   // ]
    do Output.create(94,8,28,54,0,0,0,0,0,0,0,0);          // ^
    do Output.create(95,0,0,0,0,0,0,0,0,0,63,0);           // _
    do Output.create(96,6,12,24,0,0,0,0,0,0,0,0);          // `

    do Output.create(97,0,0,0,14,24,30,27,27,54,0,0);      // a
    do Output.create(98,3,3,3,15,27,51,51,51,30,0,0);      // b
    do Output.create(99,0,0,0,30,51,3,3,51,30,0,0);        // c
    do Output.create(100,48,48,48,60,54,51,51,51,30,0,0);  // d
    do Output.create(101,0,0,0,30,51,63,3,51,30,0,0);      // e
    do Output.create(102,28,54,38,6,15,6,6,6,15,0,0);      // f
    do Output.create(103,0,0,30,51,51,51,62,48,51,30,0);   // g
    do Output.create(104,3,3,3,27,55,51,51,51,51,0,0);     // h
    do Output.create(105,12,12,0,14,12,12,12,12,30,0,0);   // i
    do Output.create(106,48,48,0,56,48,48,48,48,51,30,0);  // j
    do Output.create(107,3,3,3,51,27,15,15,27,51,0,0);     // k
    do Output.create(108,14,12,12,12,12,12,12,12,30,0,0);  // l
    do Output.create(109,0,0,0,29,63,43,43,43,43,0,0);     // m
    do Output.create(110,0,0,0,29,51,51,51,51,51,0,0);     // n
    do Output.create(111,0,0,0,30,51,51,51,51,30,0,0);     // o
    do Output.create(112,0,0,0,30,51,51,51,31,3,3,0);      // p
    do Output.create(113,0,0,0,30,51,51,51,62,48,48,0);    // q
    do Output.create(114,0,0,0,29,55,51,3,3,7,0,0);        // r
    do Output.create(115,0,0,0,30,51,6,24,51,30,0,0);      // s
    do Output.create(116,4,6,6,15,6,6,6,54,28,0,0);        // t
    do Output.create(117,0,0,0,27,27,27,27,27,54,0,0);     // u
    do Output.create(118,0,0,0,51,51,51,51,30,12,0,0);     // v
    do Output.create(119,0,0,0,51,51,51,63,63,18,0,0);     // w
    do Output.create(120,0,0,0,51,30,12,12,30,51,0,0);     // x
    do Output.create(121,0,0,0,51,51,51,62,48,24,15,0);    // y
    do Output.create(122,0,0,0,63,27,12,6,51,63,0,0);      // z

    do Output.create(123,56,12,12,12,7,12,12,12,56,0,0);   // {
    do Output.create(124,12,12,12,12,12,12,12,12,12,0,0);  // |
    do Output.create(125,7,12,12,12,56,12,12,12,7,0,0);    // }
    do Output.create(126,38,45,25,0,0,0,0,0,0,0,0);        // ~

    return;
  }

  // Creates a character map array of the given char index with the given
values.
  function void create(int index, int a, int b, int c, int d, int e, int f,
    int g, int h, int i, int j, int k) {
    var Array map;

    let map = Array.new(11);
    let charMaps[index] = map;

    let map[0] = a;
    let map[1] = b;
    let map[2] = c;
    let map[3] = d;
    let map[4] = e;
    let map[5] = f;
    let map[6] = g;
    let map[7] = h;
    let map[8] = i;
    let map[9] = j;
    let map[10] = k;

    return;
  }

  // Returns the character map (array of size 11) for the given character
  // If an invalid character is given, returns the character map of a black square.
  function Array getMap(char c) {
    if ((c < 32) | (c > 126)) {
      let c = 0;
    }

    return charMaps[c];
  }

  /** Moves the cursor to the j�th column of the i�th row,
   *  and erases the character that was there. */
  function void moveCursor(int i, int j) {
    if (((i < 0) | (i > 22)) | ((j < 0) | (j > 63))) {
      do Sys.error(20); // Output.moveCursor: Illegal cursor location
    }

    let cursor[0] = i;
    let cursor[1] = j;

    do Output.print_char(32);

    return;
  }

  /** Prints c at the cursor location and advances the cursor one
   *  column forward. */
  function void printChar(char c) {
    do Output.print_char(c);
    do Output.advance_cursor();

    return;
    return;
  }

  /** Prints s starting at the cursor location, and advances the
   *  cursor appropriately. */
  function void printString(String s) {
    var int i;

    let i = 0;

    while (i < s.length()) {
      do Output.printChar(s.charAt(i));

      let i = i + 1;
    }

    return;
  }

  /** Prints i starting at the cursor location, and advances the
   *  cursor appropriately. */
  function void printInt(int i) {
    if (i < 0) {
      do Output.printChar(45);
      let i = -i;
    }

    do Output.printString(Output.int2String(i));

    return;
  }

  /** Advances the cursor to the beginning of the next line. */
  function void println() {
    let cursor[1] = 0;

    if (cursor[0] = 22) {
      let cursor[0] = 0;
    } else {
      let cursor[0] = cursor[0] + 1;
    }

    return;
  }

  /** Moves the cursor one column back. */
  function void backSpace() {
    if (cursor[1] = 0) {
      let cursor[1] = 63;

      if (cursor[0] = 0) {
```

```
      if (cursor[0] = 0) {
        let cursor[0] = 22;
      } else {
        let cursor[0] = cursor[0] - 1;
      }
    } else {
      let cursor[1] = cursor[1] - 1;
    }

    do Output.print_char(32);

    return;
  }

  // private
  function void advance_cursor() {
    if (cursor[1] = 63) {
      let cursor[1] = 0;

      if (cursor[0] = 22) {
        let cursor[0] = 0;
      } else {
        let cursor[0] = cursor[0] + 1;
      }
    } else {
      let cursor[1] = cursor[1] + 1;
    }

    return;
  }

  function void print_char(char c) {
    var Array char_map;
    var int row, col, row_bits;
    var int row_offset, col_offset;

    let row_offset = cursor[0] * 11;
    let col_offset = cursor[1] * 8;

    let char_map = Output.getMap(c);
    let col = 0;

    while (row < 11) {
      let col = 0;
      let row_bits = char_map[row];

      while (col < 8) {
        do Screen.setColor(col_masks[col] & row_bits);
        do Screen.drawPixel(col_offset + col, row_offset + row);

        let col = col + 1;
      }

      let row = row + 1;
    }

    return;
  }

  function String int2String(int n) {
    var int lastDigit;
    var int c;
    var String s;

    let lastDigit = n - ((n / 10) * 10); // n % 10

    let c = lastDigit + 48;

    if (n < 10) {
      let s = String.new(6);
      return s.appendChar(c);
    } else {
      let s = Output.int2String(n / 10);
      return s.appendChar(c);
    }
  }
}
```