

This document describes the Application Programming Interface (API) for the supporting libraries used throughout the book *Computer Science: An Interdisciplinary Approach* [↗](#) by Robert Sedgewick and Kevin Wayne. The corresponding code is part of a package called `stdlib`.

BinaryIn

<code>BinaryIn()</code>	constructs a binary input stream from standard input
<code>BinaryIn(InputStream in)</code>	constructs a binary input stream from an input stream
<code>BinaryIn(Socket socket)</code>	constructs a binary input stream from a socket
<code>BinaryIn(URL url)</code>	constructs a binary input stream from an URL
<code>BinaryIn(String name)</code>	constructs a binary input stream from a file or an URL with the given name
<code>boolean exists()</code>	returns <code>true</code> if this binary input stream exists, and <code>false</code> otherwise
<code>boolean isEmpty()</code>	returns <code>true</code> if this binary input stream is empty, and <code>false</code> otherwise
<code>boolean readBoolean()</code>	reads and returns the next bit of data from this binary input stream as a boolean
<code>char readChar()</code>	reads and returns the next 8 bits of data from this binary input stream as a char
<code>char readChar(int r)</code>	reads and returns the next <code>r</code> bits of data from this binary input stream as an <code>r</code> -bit char
<code>String readString()</code>	reads and returns the remaining bytes of data from this binary input stream as a string
<code>short readShort()</code>	reads and returns the next 16 bits of data from this binary input stream as a short
<code>int readInt()</code>	reads and returns the next 32 bits of data from this binary input stream as an int
<code>int readInt(int r)</code>	reads and returns the next <code>r</code> bits of data from this binary input stream as an <code>r</code> -bit int
<code>long readLong()</code>	reads and returns the next 64 bits of data from this binary input stream as a long
<code>double readDouble()</code>	reads and returns the next 64 bits of data from this binary input stream as a double
<code>float readFloat()</code>	reads and returns the next 32 bits of data from this binary input stream as a float
<code>byte readByte()</code>	reads and returns the next 8 bits of data from this binary input stream as a byte

BinaryOut

<code>BinaryOut()</code>	constructs a binary output stream from standard output
<code>BinaryOut(OutputStream in)</code>	constructs a binary output stream from an output stream
<code>BinaryOut(String name)</code>	constructs a binary output stream from a file with the given name
<code>BinaryOut(Socket socket)</code>	constructs a binary output stream from a socket
<code>void flush()</code>	flushes this binary output stream, padding 0s if number of bits written is not a multiple of 8
<code>void close()</code>	flushes and closes this binary output stream
<code>void write(boolean x)</code>	writes the bit to this binary output stream
<code>void write(byte x)</code>	writes the 8-bit byte to this binary output stream
<code>void write(int x)</code>	writes the 32-bit int to this binary output stream
<code>void write(int x, int r)</code>	writes the <code>r</code> -bit int to this binary output stream
<code>void write(double x)</code>	writes the 64-bit double to this binary output stream
<code>void write(long x)</code>	writes the 64-bit long to this binary output stream
<code>void write(float x)</code>	writes the 32-bit float to this binary output stream
<code>void write(short x)</code>	writes the 16-bit short to this binary output stream
<code>void write(char x)</code>	writes the 8-bit char to this binary output stream
<code>void write(char x, int r)</code>	writes the <code>r</code> -bit char to this binary output stream

<code>void write(String x)</code>	writes the string of 8-bit characters to this binary output stream
<code>void write(String x, int r)</code>	writes the string of <code>r</code> -bit characters to this binary output stream

BinaryStdIn

<code>static void close()</code>	closes standard input and releases any associated resources
<code>static boolean isEmpty()</code>	returns <code>true</code> if standard input is empty, and <code>false</code> otherwise
<code>static boolean readBoolean()</code>	reads and returns the next bit of data from standard input as a boolean
<code>static char readChar()</code>	reads and returns the next 8 bits of data from standard input as a char
<code>static char readChar(int r)</code>	reads and returns the next <code>r</code> bits of data from standard input as an <code>r</code> -bit char
<code>static String readString()</code>	reads and returns the remaining bytes of data from standard input as a string
<code>static short readShort()</code>	reads and returns the next 16 bits of data from standard input as a short
<code>static int readInt()</code>	reads and returns the next 32 bits of data from standard input as an int
<code>static int readInt(int r)</code>	reads and returns the next <code>r</code> bits of data from standard input as an <code>r</code> -bit int
<code>static long readLong()</code>	reads and returns the next 64 bits of data from standard input as a long
<code>static double readDouble()</code>	reads and returns the next 64 bits of data from standard input as a double
<code>static float readFloat()</code>	reads and returns the next 32 bits of data from standard input as a float
<code>static byte readByte()</code>	reads and returns the next 8 bits of data from standard input as a byte

BinaryStdOut

<code>static void flush()</code>	flushes standard output, padding 0s if number of bits written is not a multiple of 8
<code>static void close()</code>	flushes and closes standard output
<code>static void write(boolean x)</code>	writes the bit to standard output
<code>static void write(byte x)</code>	writes the 8-bit byte to standard output
<code>static void write(int x)</code>	writes the 32-bit int to standard output
<code>static void write(int x, int r)</code>	writes the <code>r</code> -bit int to standard output
<code>static void write(double x)</code>	writes the 64-bit double to standard output
<code>static void write(long x)</code>	writes the 64-bit long to standard output
<code>static void write(float x)</code>	writes the 32-bit float to standard output
<code>static void write(short x)</code>	writes the 16-bit short to standard output
<code>static void write(char x)</code>	writes the 8-bit char to standard output
<code>static void write(char x, int r)</code>	writes the <code>r</code> -bit char to standard output
<code>static void write(String x)</code>	writes the string of 8-bit characters to standard output
<code>static void write(String x, int r)</code>	writes the string of <code>r</code> -bit characters to standard output

In

<code>In()</code>	constructs an input stream from standard input
<code>In(Socket socket)</code>	constructs an input stream from a socket
<code>In(URL url)</code>	constructs an input stream from an URL
<code>In(String name)</code>	constructs an input stream from a file with the given name

<code>In(Scanner scanner)</code>	constructs an input stream from a scanner
<code>boolean exists()</code>	returns true if this input stream exists, and false otherwise
<code>boolean isEmpty()</code>	returns true if this input stream is empty, and false otherwise
<code>boolean hasNextLine()</code>	returns true if this input stream has a next line, and false otherwise
<code>boolean hasNextChar()</code>	returns true if this input stream has a next char, and false otherwise
<code>String readLine()</code>	reads and returns the next line from this input stream
<code>char readChar()</code>	reads and returns the next char from this input stream
<code>String readAll()</code>	reads and returns the remainder of this input stream as a string
<code>String readString()</code>	reads and returns the next token from this input stream as a string
<code>int readInt()</code>	reads and returns the next int from this input stream
<code>double readDouble()</code>	reads and returns the next double from this input stream
<code>float readFloat()</code>	reads and returns the next float from this input stream
<code>long readLong()</code>	reads and returns the next long from this input stream
<code>short readShort()</code>	reads and returns the next short from this input stream
<code>byte readByte()</code>	reads and returns the next byte from this input stream
<code>boolean readBoolean()</code>	reads and returns the next boolean from this input stream
<code>String[] readAllStrings()</code>	reads and returns all the remaining tokens from this input stream as an array of strings
<code>String[] readAllLines()</code>	reads and returns all the remaining lines from this input stream as an array of strings
<code>int[] readAllInts()</code>	reads and returns all the remaining tokens from this input stream as an array of ints
<code>long[] readAllLongs()</code>	reads and returns all the remaining tokens from this input stream as an array of longs
<code>double[] readAllDoubles()</code>	reads and returns all the remaining tokens from this input stream as an array of doubles
<code>void close()</code>	closes this input stream

Out

<code>Out()</code>	constructs an output stream from standard output
<code>Out(Socket socket)</code>	constructs an output stream from a socket
<code>Out(String name)</code>	constructs an output stream from a file with the given name
<code>void close()</code>	closes this output stream
<code>void println()</code>	prints a newline to this output stream
<code>void println(Object x)</code>	prints an object and a newline to this output stream
<code>void println(boolean x)</code>	prints a boolean and a newline to this output stream
<code>void println(char x)</code>	prints a char and a newline to this output stream
<code>void println(double x)</code>	prints a double and a newline to this output stream
<code>void println(float x)</code>	prints a float and a newline to this output stream
<code>void println(int x)</code>	prints an int and a newline to this output stream
<code>void println(long x)</code>	prints a long and a newline to this output stream
<code>void println(byte x)</code>	prints a byte and a newline to this output stream
<code>void print()</code>	flushes this output stream
<code>void print(Object x)</code>	prints an object to this output stream
<code>void print(boolean x)</code>	prints a boolean to this output stream
<code>void print(char x)</code>	prints a char to this output stream

<code>void print(double x)</code>	prints a double to this output stream
<code>void print(float x)</code>	prints a float to this output stream
<code>void print(int x)</code>	prints an int to this output stream
<code>void print(long x)</code>	prints a long to this output stream
<code>void print(byte x)</code>	prints a byte to this output stream
<code>void printf(String fmt, Object... args)</code>	prints <code>args</code> to this output stream using format string <code>fmt</code>
<code>void printf(Locale loc, String fmt, Object... args)</code>	prints <code>args</code> to this output stream using locale <code>loc</code> and format string <code>fmt</code>

Picture

<code>Picture(int w, int h)</code>	constructs a picture of width <code>w</code> and height <code>h</code>
<code>Picture(Picture pic)</code>	constructs a picture that is a deep copy of <code>pic</code>
<code>Picture(String name)</code>	constructs a picture from an image with the given name
<code>JLabel getLabel()</code>	returns the current label of this picture for use in some other GUI
<code>void setOriginUpperLeft()</code>	sets the origin to be the upper left pixel (default)
<code>void setOriginLowerLeft()</code>	sets the origin to be the lower left pixel
<code>void show()</code>	displays this picture on the screen
<code>int height()</code>	returns the height of this picture
<code>int width()</code>	returns the width of this picture
<code>RGBColor get(int col, int row)</code>	returns the color of pixel (<code>col</code> , <code>row</code>) as a <code>RGBColor</code> object
<code>void set(int col, int row, RGBColor c)</code>	sets the color of the pixel (<code>col</code> , <code>row</code>) to the given value (a <code>RGBColor</code> object)
<code>void save(String name)</code>	saves this picture to a file with the given name
<code>void actionPerformed(ActionEvent e)</code>	opens a save dialog box when the user selects “Save As” from the menu

RGBColor

<code>RGBColor(int r, int g, int b)</code>	constructs a color given its red, green, and blue components
<code>public int getRed()</code>	returns the red component of the color
<code>public int getGreen()</code>	returns the green component of the color
<code>public int getBlue()</code>	returns the blue component of the color
<code>public double luminance()</code>	returns the luminance of the color
<code>public RGBColor toGray()</code>	returns the grayscale equivalent of the color
<code>public boolean isCompatible(RGBColor other)</code>	returns <code>true</code> if the color is compatible with <code>other</code> , and <code>false</code> otherwise
<code>public String toString()</code>	returns a string representation of the color

StdArrayIO

<code>static double[] readDouble1D()</code>	reads an integer <code>n</code> from standard input, and then reads <code>n</code> doubles also from standard input and returns them as a 1D array of size <code>n</code>
<code>static void print(double[] a)</code>	prints the size and elements of the 1D array <code>a</code> to standard output

<code>static double[] [] readDouble2D()</code>	reads integers m and n from standard input, and then reads mn doubles also from standard input and returns them as a 2D array of size $m \times n$
<code>static void print(double[] [] a)</code>	prints the size and elements of the 2D array a to standard output
<code>static int[] readInt1D()</code>	reads an integer n from standard input, and then reads n ints also from standard input and returns them as a 1D array of size n
<code>static void print(int[] a)</code>	prints the size and elements of the 1D array a to standard output
<code>static int[] [] readInt2D()</code>	reads integers m and n from standard input, and then reads mn ints also from standard input and returns them as a 2D array of size $m \times n$
<code>static void print(int[] [] a)</code>	prints the size and elements of the 2D array a to standard output
<code>static boolean[] readBoolean1D()</code>	reads an integer n from standard input, and then reads n booleans also from standard input and returns them as a 1D array of size n
<code>static void print(boolean[] a)</code>	prints the size and elements of the 1D array a to standard output
<code>static boolean[] [] readBoolean2D()</code>	reads integers m and n from standard input, and then reads mn booleans also from standard input and returns them as a 2D array of size $m \times n$
<code>static void print(boolean[] [] a)</code>	prints the size and elements of the 2D array a to standard output

StdAudio

<code>static void close()</code>	closes standard audio
<code>static void play(double sample)</code>	plays one sample (between -1.0 and +1.0) using standard audio
<code>static void play(double[] samples)</code>	plays an array of samples (between -1.0 and +1.0) using standard audio
<code>static double[] read(String name)</code>	reads and returns samples from a file (.au or .wav format) with the given name
<code>static void save(String name, double[] samples)</code>	saves the samples as a file (.au or .wav format) with the given name
<code>static void play(String name)</code>	plays a file (.au, .mid, or .au format) with the given name in the background using standard audio
<code>static void loop(String name)</code>	loops a file (.au, .mid, or .au format) with the given name in the background using standard audio

StdDraw

<code>static RGBColor BLACK</code>	represents black
<code>static RGBColor BLUE</code>	represents blue
<code>static RGBColor CYAN</code>	represents cyan
<code>static RGBColor DARK_GRAY</code>	represents dark gray
<code>static RGBColor GREEN</code>	represents green
<code>static RGBColor LIGHT_GRAY</code>	represents light gray
<code>static RGBColor MAGENTA</code>	represents magenta
<code>static RGBColor ORANGE</code>	represents orange
<code>static RGBColor PINK</code>	represents pink
<code>static RGBColor RED</code>	represents red

<code>static RGBColor WHITE</code>	represents white
<code>static RGBColor YELLOW</code>	represents yellow
<code>static void setCanvasSize()</code>	sets the width and height of the canvas to 512 pixels
<code>static void setCanvasSize(int w, int h)</code>	sets the width and height of the canvas to <code>w</code> and <code>h</code> pixels
<code>static void setXscale()</code>	sets the <i>x</i> -scale of the canvas to the default <code>([0, 1])</code>
<code>static void setYscale()</code>	sets the <i>y</i> -scale of the canvas to the default <code>([0, 1])</code>
<code>static void setScale()</code>	sets the <i>x</i> - and <i>y</i> -scale of the canvas to the default <code>([0, 1])</code>
<code>static void setXscale(double min, double max)</code>	sets the <i>x</i> -scale of the canvas to <code>[min, max]</code>
<code>static void setYscale(double min, double max)</code>	sets the <i>y</i> -scale of the canvas to <code>[min, max]</code>
<code>static void setScale(double min, double max)</code>	sets the <i>x</i> - and <i>y</i> -scale of the canvas to <code>[min, max]</code>
<code>static void clear()</code>	clears the canvas to the default (white) color
<code>static void clear(GBColor c)</code>	clears the canvas to the color <code>c</code>
<code>static double getPenRadius()</code>	returns the current pen radius of the canvas
<code>static void setPenRadius()</code>	sets the pen radius of the canvas to the default (0.002)
<code>static void setPenRadius(double r)</code>	sets the pen radius of the canvas to <code>r</code>
<code>static RGBColor getPenColor()</code>	returns the current pen color of the canvas
<code>static void setPenColor()</code>	sets the pen color of the canvas to the default (black) color
<code>static void setPenColor(GBColor c)</code>	sets the pen color of the canvas to the color <code>c</code>
<code>static void setPenColor(int r, int g, int b)</code>	sets the pen color of the canvas to the <code>(r, g, b)</code> color
<code>static void getFont()</code>	returns the current font for the canvas
<code>static void setFont()</code>	sets the font for the canvas to the default (sans serif, 16 point) font
<code>static void setFont(Font f)</code>	sets the font for the canvas to the font <code>f</code>
<code>static void drawLine(double x0, double y0, double x1, double y1)</code>	draws a line on the canvas from <code>(x0, y0)</code> to <code>(x1, y1)</code>
<code>static void draw(double x, double y)</code>	draws one pixel on the canvas at <code>(x, y)</code>
<code>static void point(double x, double y)</code>	draws a point on the canvas at <code>(x, y)</code>
<code>static void circle(double x, double y, double r)</code>	draws a circle on the canvas of radius <code>r</code> , centered at <code>(x, y)</code>
<code>static void filledCircle(double x, double y, double r)</code>	draws a filled circle on the canvas of radius <code>r</code> , centered at <code>(x, y)</code>
<code>static void ellipse(double x, double y, double maj, double min)</code>	draws an ellipse on the canvas with semimajor and semiminor axes <code>maj</code> and <code>min</code> , centered at <code>(x, y)</code>
<code>static void filledEllipse(double x, double y, double maj, double min)</code>	draws a filled ellipse on the canvas with semimajor and semiminor axes <code>maj</code> and <code>min</code> , centered at <code>(x, y)</code>
<code>static void arc(double x, double y, double r, double a1, double a2)</code>	draws an arc on the canvas with radius <code>r</code> , centered at <code>(x, y)</code> , from <code>a1</code> to <code>a2</code> (in degrees)
<code>static void square(double x, double y, double l)</code>	draws a square on the canvas of side length <code>l</code> , centered at <code>(x, y)</code>
<code>static void filledSquare(double x, double y, double l)</code>	draws a filled square on the canvas of side length <code>l</code> , centered at <code>(x, y)</code>
<code>static void rectangle(double x, double y, double w, double h)</code>	draws a rectangle on the canvas of width <code>2w</code> and height <code>2h</code> , centered at <code>(x, y)</code>
<code>static void filledRectangle(double x, double y, double w, double h)</code>	draws a filled rectangle on the canvas of width <code>2w</code> and height <code>2h</code> , centered at <code>(x, y)</code>

<code>static void polygon(double[] x, double[] y)</code>	draws a polygon on the canvas with vertices whose coordinates are given by <code>x</code> and <code>y</code>
<code>static void filledPolygon(double[] x, double[] y)</code>	draws a filled polygon on the canvas with vertices whose coordinates are given by <code>x</code> and <code>y</code>
<code>static void picture(double x, double y, String name)</code>	draws the image with the given name on the canvas, centered at <code>(x, y)</code>
<code>static void picture(double x, double y, String name, double a)</code>	draws the image with the given name on the canvas, centered at <code>(x, y)</code> , rotated by <code>a</code> (in degrees)
<code>static void picture(double x, double y, String name, double w, double h)</code>	draws the image with the given name on the canvas, centered at <code>(x, y)</code> , rescaled to a <code>w x h</code> bounding box
<code>static void picture(double x, double y, String name, double w, double h, double a)</code>	draws the image with the given name on the canvas, centered at <code>(x, y)</code> , rotated by <code>a</code> (in degrees), rescaled to a <code>w x h</code> bounding box
<code>static void text(double x, double y, String s)</code>	draws the text <code>s</code> on the canvas, centered at <code>(x, y)</code>
<code>static void text(double x, double y, String s, double a)</code>	draws the text <code>s</code> on the canvas, centered at <code>(x, y)</code> , rotated by <code>a</code> (in degrees)
<code>static void textLeft(double x, double y, String s)</code>	draws the text <code>s</code> on the canvas, left-aligned at <code>(x, y)</code>
<code>static void textRight(double x, double y, String s)</code>	draws the text <code>s</code> on the canvas, right-aligned at <code>(x, y)</code>
<code>static void pause(int t)</code>	pauses the canvas for <code>t</code> milliseconds
<code>static void show()</code>	copies offscreen buffer to onscreen buffer on the canvas
<code>static void enableDoubleBuffering()</code>	enables double buffering on the canvas
<code>static void disableDoubleBuffering()</code>	disables double buffering on the canvas
<code>static void save(String name)</code>	saves the drawing on the canvas to a file with the given name
<code>static boolean isMousePressed()</code>	returns <code>true</code> if the mouse is being pressed on the canvas, and <code>false</code> otherwise
<code>static double mouseX()</code>	returns the <i>x</i> -coordinate of the mouse pointer on the canvas
<code>static double mouseY()</code>	returns the <i>y</i> -coordinate of the mouse pointer on the canvas
<code>static boolean hasNextKeyTyped()</code>	returns <code>true</code> if the user has typed a key on the canvas, and <code>false</code> otherwise
<code>static char nextKeyTyped()</code>	returns the next key typed by the user on the canvas
<code>static boolean isKeyPressed(int c)</code>	returns <code>true</code> if the keycode <code>c</code> is being pressed on the canvas, and <code>false</code> otherwise

StdIn

<code>static boolean isEmpty()</code>	returns <code>true</code> if standard input is empty, and <code>false</code> otherwise
<code>static boolean hasNextLine()</code>	returns <code>true</code> if standard input has a next line, and <code>false</code> otherwise
<code>static boolean hasNextChar()</code>	returns <code>true</code> if standard input has a next char, and <code>false</code> otherwise
<code>static String readLine()</code>	reads and returns the next line from standard input
<code>char readChar()</code>	reads and returns the next char from this standard input
<code>static String readAll()</code>	reads and returns the remainder of standard input as a string
<code>static String readString()</code>	reads and returns the next token from standard input as a string

<code>static int readInt()</code>	reads and returns the next int from standard input
<code>static double readDouble()</code>	reads and returns the next double from standard input
<code>static float readFloat()</code>	reads and returns the next float from standard input
<code>static long readLong()</code>	reads and returns the next long from standard input
<code>static short readShort()</code>	reads and returns the next short from standard input
<code>static byte readByte()</code>	reads and returns the next byte from standard input
<code>static boolean readBoolean()</code>	reads and returns the next boolean from standard input
<code>static String[] readAllStrings()</code>	reads and returns all the remaining tokens from standard input as an array of strings
<code>static String[] readAllLines()</code>	reads and returns all the remaining lines from standard input as an array of strings
<code>static int[] readAllInts()</code>	reads and returns all the remaining tokens from standard input as an array of ints
<code>static long[] readAllLongs()</code>	reads and returns all the remaining tokens from standard input as an array of longs
<code>static double[] readAllDoubles()</code>	reads and returns all the remaining tokens from standard input as an array of doubles
<code>static void resync()</code>	reinitializes the scanner underlying standard input

StdOut

<code>static void println()</code>	prints a newline to standard output
<code>static void println(Object x)</code>	prints an object and a newline to standard output
<code>static void println(boolean x)</code>	prints a boolean and a newline to standard output
<code>static void println(char x)</code>	prints a char and a newline to standard output
<code>static void println(double x)</code>	prints a double and a newline to standard output
<code>static void println(float x)</code>	prints a float and a newline to standard output
<code>static void println(int x)</code>	prints an int and a newline to standard output
<code>static void println(long x)</code>	prints a long and a newline to standard output
<code>static void println(short x)</code>	prints a short and a newline to standard output
<code>static void println(byte x)</code>	prints a byte and a newline to standard output
<code>static void print()</code>	flushes standard output
<code>static void print(Object x)</code>	prints an object to standard output
<code>static void print(boolean x)</code>	prints a boolean to standard output
<code>static void print(char x)</code>	prints a char to standard output
<code>static void print(double x)</code>	prints a double to standard output
<code>static void print(float x)</code>	prints a float to standard output
<code>static void print(int x)</code>	prints an int to standard output
<code>static void print(long x)</code>	prints a long to standard output
<code>static void print(short x)</code>	prints a short to standard output
<code>static void print(byte x)</code>	prints a byte to standard output
<code>static void printf(String fmt, Object... args)</code>	prints <code>args</code> to standard output using format string <code>fmt</code>

<code>static void printf(Locale loc, String fmt, Object... args)</code>	prints args to standard output using locale loc and format string fmt
<code>static void resync()</code>	reinitializes the writer underlying standard output

StdRandom

<code>static seed(long s)</code>	sets the seed of the random number generator to s
<code>static long getSeed()</code>	returns the seed of the random number generator
<code>static double uniform()</code>	returns a double chosen uniformly at random from the interval [0, 1)
<code>static int uniform(int n)</code>	returns an integer chosen uniformly at random from the interval [0, n)
<code>static long uniform(long n)</code>	returns a long chosen uniformly at random from the interval [0, n)
<code>static int uniform(int a, int b)</code>	returns an integer chosen uniformly at random from the interval [a, b)
<code>static double uniform(double a, double b)</code>	returns a double chosen uniformly at random from the interval [a, b)
<code>static boolean bernoulli(double p)</code>	returns true with probability p and false with probability 1 - p
<code>static boolean bernoulli()</code>	returns true with probability 0.5 and false with probability 0.5
<code>static double gaussian()</code>	returns a double from the standard Gaussian distribution
<code>static double gaussian(double mu, double sigma)</code>	returns a double from a Gaussian distribution with mean mu and standard deviation sigma
<code>static int geometric(double p)</code>	returns an integer from a geometric distribution with success probability p
<code>static int poisson(double lambda)</code>	returns an integer from a Poisson distribution with mean lambda
<code>static double pareto()</code>	returns a double from the standard Pareto distribution
<code>static double pareto(double alpha)</code>	returns a double from a Pareto distribution with shape parameter alpha
<code>static double cauchy()</code>	returns a double from the Cauchy distribution
<code>static int discrete(double[] probabilities)</code>	returns an integer i with probability probabilities[i]
<code>static int discrete(int[] frequencies)</code>	returns an integer i with probability frequencies[i]
<code>static double exp(double lambda)</code>	returns a double from an exponential distribution with rate lambda
<code>static void shuffle(Object[] a)</code>	shuffles the array a
<code>static void shuffle(double[] a)</code>	shuffles the array a
<code>static void shuffle(int[] a)</code>	shuffles the array a
<code>static void shuffle(Object[] a, int lo, int hi)</code>	shuffles the subarray a[lo, ..., hi)
<code>static void shuffle(double[] a, int lo, int hi)</code>	shuffles the subarray a[lo, ..., hi)
<code>static void shuffle(int[] a, int lo, int hi)</code>	shuffles the subarray a[lo, ..., hi)

<code>static int[] permutation(int n)</code>	returns a uniformly random permutation of n elements
<code>static int[] permutation(int n, int k)</code>	returns a uniformly random permutation of k (out of n) elements

StdStats

<code>static double max(double[] a)</code>	returns the maximum value in the array a
<code>static double max(double[] a, int lo, int hi)</code>	returns the maximum value in the subarray a[lo, ..., hi]
<code>static int max(int[] a)</code>	returns the maximum value in the array a
<code>static double min(double[] a)</code>	returns the minimum value in the array a
<code>static double min(double[] a, int lo, int hi)</code>	returns the minimum value in the subarray a[lo, ..., hi]
<code>static int min(int[] a)</code>	returns the minimum value in the array a
<code>static double mean(double[] a)</code>	returns the average value in the array a
<code>static double mean(double[] a, int lo, int hi)</code>	returns the average value in the subarray a[lo, ..., hi]
<code>static double mean(int[] a)</code>	returns the average value in the array a
<code>static double var(double[] a)</code>	returns the sample variance in the array a
<code>static double var(double[] a, int lo, int hi)</code>	returns the sample variance in the subarray a[lo, ..., hi]
<code>static double var(int[] a)</code>	returns the sample variance in the array a
<code>static double varp(double[] a)</code>	returns the population variance in the array a
<code>static double varp(double[] a, int lo, int hi)</code>	returns the population variance in the subarray a[lo, ..., hi]
<code>static double stddev(double[] a)</code>	returns the sample standard deviation in the array a
<code>static double stddev(double[] a, int lo, int hi)</code>	returns the sample standard deviation in the subarray a[lo, ..., hi]
<code>static double stddev(int[] a)</code>	returns the sample standard deviation in the array a
<code>static double stddevp(double[] a)</code>	returns the population standard deviation in the array a
<code>static double stddevp(double[] a, int lo, int hi)</code>	returns the population standard deviation in the subarray a[lo, ..., hi]
<code>static double sum(double[] a)</code>	returns the sum of all values in the array a
<code>static double sum(double[] a, int lo, int hi)</code>	returns the sum of all values in the subarray a[lo, ..., hi]
<code>static int sum(int[] a)</code>	returns the sum of all values in the array a
<code>static void plotPoints(double[] a)</code>	plots the values in the array a as points
<code>static void plotLines(double[] a)</code>	plots the values in the array a as line end-points
<code>static void plotBars(double[] a)</code>	plots the values in the array a as bars