To simulate the quantization of real numbers into other real but a quantized one, we use the routine **quantize.m**. It receives as input two parameters: a real number x and the total number of bits, B (including the sign bit), for its quantized representation. The routine then returns a real number that corresponds to the quantized value of x. This value is determined as follows. With B total bits, the largest integers that can be represented are

$$\pm 2^{(B-1)} - 1$$

If x exceeds these extreme values then its quantized representation is taken as either one of them, depending on the sign of x. If, on the other hand, x falls within the interval

$$x \in \left(-2^{(B-1)} - 1, 2^{(B-1)} - 1\right)$$

then the routine determines how many bits are needed to represent the integer part of x, and the remaining bits are used to represent the fractional part of x.

```
function y = quantize(x,B)

% The function rounds x into a binary fixed point
% representation with B bits, including the sign bit.
% If overflow or underflow occurs, the largest possible
% fixed-point representation is returned.
% The largest numbers that can be represented are
% +- 2^(B-1)-1
% If x is less than these extreme values, the routine finds
% the number of bits that are needed to represent the integer
% part of x and uses the remaining bits for its fractional part.

y = 0;
Fix_x = fix(x); % Integer part of x
frac = x - Fix; % Fractional part of x

if abs(x) >= (2^(B-1)-1)
   if x > 0
      y = 2^(B-1)-1;    % overflow
   else
      y = -(2^(B-1)-1); % underflow
   end;
else
   for i=0:B-1,
      if abs(x) < 2^i; % i bits are needed to represent the integer part of x
         M = B-i-1;  % M bits are used for its fractional part
         y = (2^(-M)*round(frac/2^(-M))) + fix(x);
         break
      end
   end
end
```