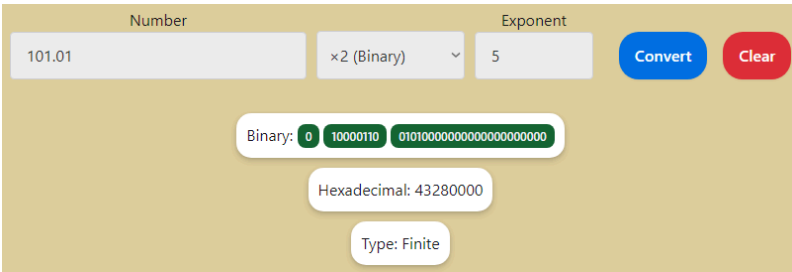
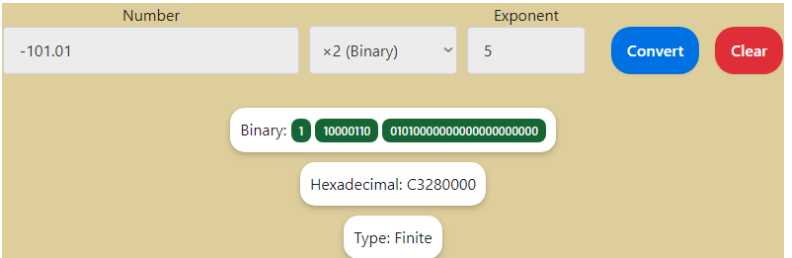
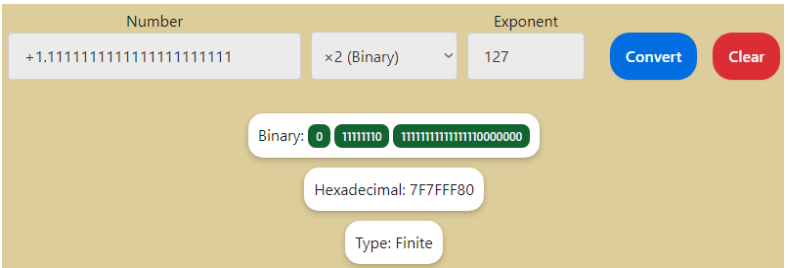


**IEEE-754 Binary-32 Floating-Point Converter**  
**Program Output Screenshots of Test Cases**

BASE-2 (BINARY)	
<b>[Positive Normal]</b>  Number: 101.01 Exponent: 5  Expected Output: Sign Bit: 0 Exponent: 10000110 Mantissa: 010 1000 0000 0000 0000 0000 Hexadecimal: 43280000 Type: Finite	
<b>[Negative Normal]</b>  Number: -101.01 Exponent: 5  Expected Output: Sign Bit: 1 Exponent: 10000110 Mantissa: 010 1000 0000 0000 0000 0000 Hexadecimal: C3280000 Type: Finite	
<b>[Positive Largest Normal]</b>  Number: +1.11111111111111111111 Exponent: 127  Expected Output: Sign Bit: 0 Exponent: 11111110 Mantissa: 111 1111 1111 1111 1000 0000 Hexadecimal: 7F7FFF80 Type: Finite	

**[Negative  
Largest-Magnitude Normal]**

Number:  
-1.11111111111111111111111111111111  
Exponent: 127

Expected Output:  
Sign Bit: 1  
Exponent: 11111110  
Mantissa: 111 1111 1111 1111  
1000 0000  
Hexadecimal: FF7FFF80  
Type: Finite

The screenshot shows a floating-point converter interface. At the top, there are two input fields: 'Number' with the value '-1.11111111111111111111111111111111' and 'Exponent' with the value '127'. Between them is a dropdown menu set to 'x2 (Binary)'. To the right are two buttons: 'Convert' (blue) and 'Clear' (red). Below the inputs, the results are displayed in a light beige box. 'Binary:' shows a sign bit '1' followed by two groups of bits: '11111110' and '1111111111110000000'. 'Hexadecimal:' shows 'FF7FFF80'. 'Type:' shows 'Finite'.

**[Positive Smallest Normal]**

Number: +1.0  
Exponent: -126

Expected Output:  
Sign Bit: 0  
Exponent: 00000001  
Mantissa: 000 0000 0000  
0000 0000 0000  
Hexadecimal: 00800000  
Type: Finite

The screenshot shows a floating-point converter interface. At the top, there are two input fields: 'Number' with the value '1.0' and 'Exponent' with the value '-126'. Between them is a dropdown menu set to 'x2 (Binary)'. To the right are two buttons: 'Convert' (blue) and 'Clear' (red). Below the inputs, the results are displayed in a light beige box. 'Binary:' shows a sign bit '0' followed by two groups of bits: '00000001' and '000000000000000000000000'. 'Hexadecimal:' shows '00800000'. 'Type:' shows 'Finite'.

**[Negative Smallest Normal]**

Number: -1.0  
Exponent: -126

Expected Output:  
Sign Bit: 1  
Exponent: 00000001  
Mantissa: 000 0000 0000  
0000 0000 0000  
Hexadecimal: 80800000  
Type: Finite

The screenshot shows a floating-point converter interface. At the top, there are two input fields: 'Number' with the value '-1.0' and 'Exponent' with the value '-126'. Between them is a dropdown menu set to 'x2 (Binary)'. To the right are two buttons: 'Convert' (blue) and 'Clear' (red). Below the inputs, the results are displayed in a light beige box. 'Binary:' shows a sign bit '1' followed by two groups of bits: '00000001' and '000000000000000000000000'. 'Hexadecimal:' shows '80800000'. 'Type:' shows 'Finite'.

### [Positive Denormalized]

Number: +1111.00000111  
Exponent: -135

Expected Output:  
Sign Bit: 0  
Exponent: 00000000  
Mantissa: 000 0011 1100  
0001 1100 0000  
Hexadecimal: 0003C1C0  
Type: Denormalized

The screenshot shows a web-based floating-point converter. The 'Number' input field contains '+1111.00000111' and the 'Exponent' input field contains '-135'. A dropdown menu is set to 'x2 (Binary)'. There are 'Convert' and 'Clear' buttons. Below the inputs, the 'Binary' field shows '0 00000000 000001110000011000000'. The 'Hexadecimal' field shows '0003C1C0'. The 'Type' field shows 'Denormalized'.

### [Negative Denormalized]

Number: -1.1110  
Exponent: -130

Expected Output:  
Sign Bit: 1  
Exponent: 00000000  
Mantissa: 000 1111 0000  
0000 0000 0000  
Hexadecimal: 800F0000  
Type: Denormalized

The screenshot shows the same floating-point converter. The 'Number' input field contains '-1.1110' and the 'Exponent' input field contains '-130'. The dropdown menu is set to 'x2 (Binary)'. The 'Binary' field shows '1 00000000 0001110000000000000000'. The 'Hexadecimal' field shows '800F0000'. The 'Type' field shows 'Denormalized'.

### [Positive Infinity]

Number: +1.111111  
Exponent: 444

Expected Output:  
Sign Bit: 0  
Exponent: 11111111  
Mantissa: 000 0000 0000  
0000 0000 0000  
Hexadecimal: 7F800000  
Type: Positive Infinity

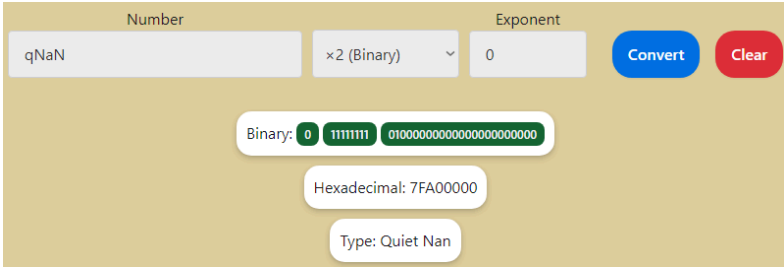
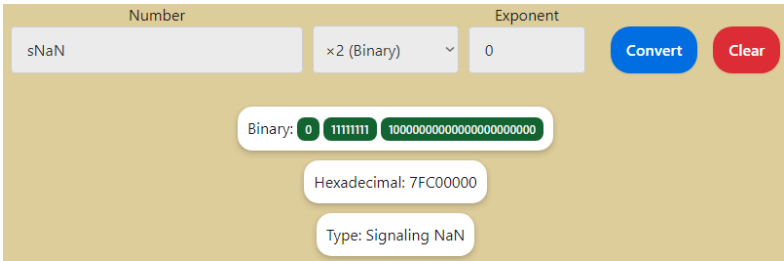
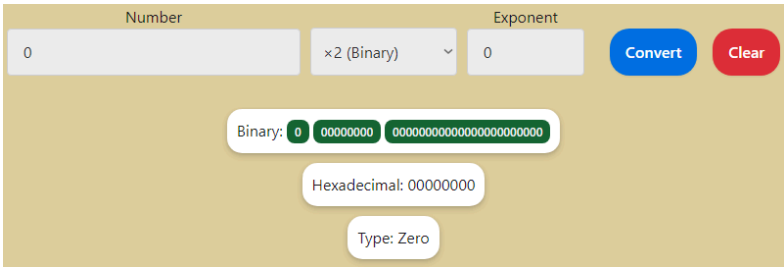
The screenshot shows the same floating-point converter. The 'Number' input field contains '1.111111' and the 'Exponent' input field contains '444'. The dropdown menu is set to 'x2 (Binary)'. The 'Binary' field shows '0 11111111 000000000000000000000000'. The 'Hexadecimal' field shows '7F800000'. The 'Type' field shows 'Positive Infinity'.

### [Negative Infinity]

Number: -1.1101010  
Exponent: 444

Expected Output:  
Sign Bit: 1  
Exponent: 11111111  
Mantissa: 000 0000 0000

The screenshot shows the same floating-point converter. The 'Number' input field contains '-1.1101010' and the 'Exponent' input field contains '444'. The dropdown menu is set to 'x2 (Binary)'. The 'Binary' field shows '1 11111111 000000000000000000000000'. The 'Hexadecimal' field shows 'FF800000'. The 'Type' field shows 'Negative Infinity'.

0000 0000 0000 Hexadecimal: FF800000 Type: Negative Infinity	
<b>[qNaN]</b>  Number: qNaN Exponent: 0  Expected Output: Sign Bit: 0 Exponent: 11111111 Mantissa: 010 0000 0000 0000 0000 0000 Hexadecimal: 7FA00000 Type: Quiet NaN	
<b>[sNaN]</b>  Number: sNaN Exponent: 0  Expected Output: Sign Bit: 1 Exponent: 11111111 Mantissa: 100 0000 0000 0000 0000 0000 Hexadecimal: 7FC00000 Type: Signaling NaN	
<b>[Zero]</b>  Number: 0 Exponent: 0  Expected Output: Sign Bit: 0 Exponent: 00000000 Mantissa: 000 0000 0000 0000 0000 0000 Hexadecimal: 00000000 Type: Zero	
<b>BASE-10 (DECIMAL)</b>	

### [Positive Normal]

Number: +65.0

Exponent: 3

Expected Output:

Sign Bit: 0

Exponent: 10001110

Mantissa: 111 1101 1110 1000  
0000 0000

Hexadecimal: 477DE800

Type: Finite

The screenshot shows a web-based floating-point converter. At the top, there are two input fields: 'Number' with the value '65.0' and 'Exponent' with the value '3'. Between them is a dropdown menu set to 'x10 (Decimal)'. To the right are two buttons: 'Convert' (blue) and 'Clear' (red). Below these inputs, the results are displayed in a light beige box. 'Binary:' is followed by a green bar containing '0', '10001110', and '1111101110100000000000'. 'Hexadecimal:' is followed by a white box containing '477DE800'. 'Type:' is followed by a white box containing 'Finite'.

### [Negative Normal]

Input:  $-65.0 \times 10^3$

Expected Output:

Sign Bit: 1

Exponent: 10001110

Mantissa: 111 1101 1110 1000  
0000 0000

Hexadecimal: C77DE800

Type: Finite

The screenshot shows the same web-based floating-point converter. The 'Number' field now contains '-65.0' and the 'Exponent' field contains '3'. The 'Convert' button is blue and the 'Clear' button is red. The results box shows 'Binary:' with a green bar containing '1', '10001110', and '1111101110100000000000'. 'Hexadecimal:' shows 'C77DE800' and 'Type:' shows 'Finite'.

### [Positive Largest Normal]

Number: +3.4

Exponent: 38

Expected Output:

Sign Bit: 0

Exponent: 10000000

Mantissa: 100 0000 0000  
0000 0000 0000

Hexadecimal: 40400000

Type: Finite

The screenshot shows the floating-point converter with 'Number' set to '+3.4' and 'Exponent' set to '38'. The 'Convert' button is blue and the 'Clear' button is red. The results box shows 'Binary:' with a green bar containing '0', '10000000', and '1000000000000000000000'. 'Hexadecimal:' shows '40400000' and 'Type:' shows 'Finite'.

### [Negative Largest-Magnitude Normal]

Number: -3.4

Exponent: 38

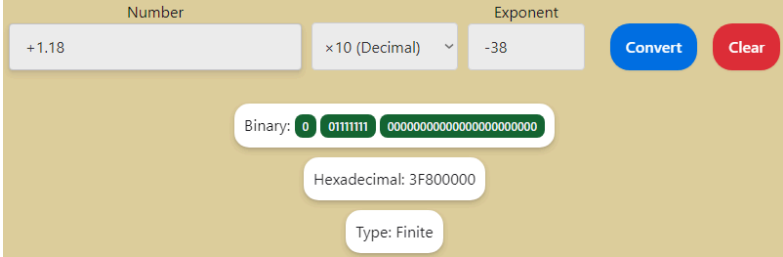
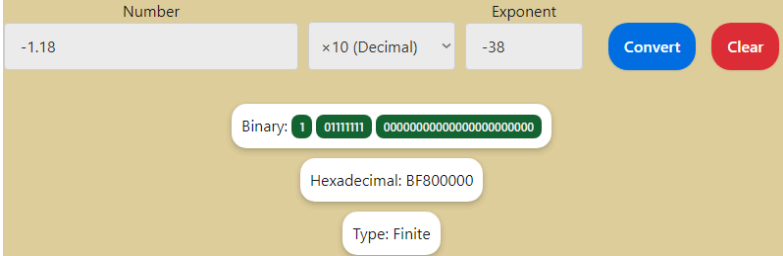
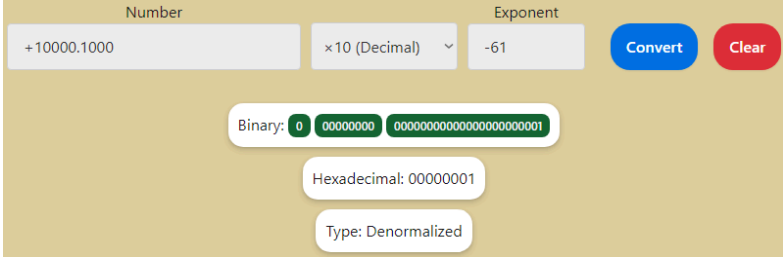
Expected Output:

Sign Bit: 1

Exponent: 10000000

Mantissa: 100 0000 0000

The screenshot shows the floating-point converter with 'Number' set to '-3.4' and 'Exponent' set to '38'. The 'Convert' button is blue and the 'Clear' button is red. The results box shows 'Binary:' with a green bar containing '1', '10000000', and '1000000000000000000000'. 'Hexadecimal:' shows 'C0400000' and 'Type:' shows 'Finite'.

0000 0000 0000 Hexadecimal: C0400000 Type: Finite	
<p><b>[Positive Smallest Normal]</b></p> <p>Number: +1.18  Exponent: -38</p> <p>Expected Output:  Sign Bit: 0  Exponent: 01111111  Mantissa: 000 0000 0000  0000 0000 0000  Hexadecimal: 3F800000  Type: Finite</p>	
<p><b>[Negative Smallest-Magnitude Normal]</b></p> <p>Number: -1.18  Exponent: -38</p> <p>Expected Output:  Sign Bit: 1  Exponent: 01111111  Mantissa: 000 0000 0000  0000 0000 0000  Hexadecimal: BF800000  Type: Finite</p>	
<p><b>[Positive Denormalized]</b></p> <p>Number: +10000.1000  Exponent: -61</p> <p>Expected Output:  Sign Bit: 0  Exponent: 00000000  Mantissa: 000 0000 0000  0000 0000 0001  Hexadecimal: 00000001  Type: Denormalized</p>	

### [Negative Denormalized]

Number: -126.2943

Exponent: -58

Expected Output:

Sign Bit: 1

Exponent: 00000000

Mantissa: 000 0000 0000

0000 0000 1000

Hexadecimal: 80000008

Type: Denormalized

The screenshot shows a web-based floating-point converter. At the top, there are two input fields: 'Number' with the value '-126.2943' and 'Exponent' with the value '-58'. A dropdown menu between them is set to '×10 (Decimal)'. To the right are two buttons: 'Convert' (blue) and 'Clear' (red). Below these inputs, the results are displayed in a light beige box. The 'Binary' field shows a sign bit of '1' followed by two groups of zeros: '00000000' and '000000000000000000000000'. The 'Hexadecimal' field shows '80000008'. The 'Type' field shows 'Denormalized'.

### [Positive Infinity]

Number:

123456789.87654321

Exponent: 143

Expected Output:

Sign Bit: 0

Exponent: 11111111

Mantissa: 000 0000 0000

0000 0000 0000

Hexadecimal: 7F800000

Type: Positive Infinity

The screenshot shows the same floating-point converter interface. The 'Number' field contains '123456789.87654321' and the 'Exponent' field contains '143'. The 'Convert' button is active. The results box shows the 'Binary' field with a sign bit of '0', followed by '11111111' and a group of zeros. The 'Hexadecimal' field shows '7F800000'. The 'Type' field shows 'Positive Infinity'.

### [Negative Infinity]

Number:

-112233445566778899

Exponent: 256

Expected Output:

Sign Bit: 1

Exponent: 11111111

Mantissa: 000 0000 0000

0000 0000 0000

Hexadecimal: FF800000

Type: Positive Infinity

The screenshot shows the same floating-point converter interface. The 'Number' field contains '-112233445566778899' and the 'Exponent' field contains '256'. The 'Convert' button is active. The results box shows the 'Binary' field with a sign bit of '1', followed by '11111111' and a group of zeros. The 'Hexadecimal' field shows 'FF800000'. The 'Type' field shows 'Negative Infinity'.

### [qNaN]

Number: qNaN

Exponent: 0

Expected Output:

Sign Bit: 0

Exponent: 11111111

Mantissa: 010 0000 0000  
0000 0000 0000

Hexadecimal: 7FA00000

Type: Quiet NaN

The screenshot shows the 'Number' field with 'qNaN' and the 'Exponent' field with '0'. The 'Convert' button is highlighted. Below the input fields, the 'Binary' field shows '0 11111111 010000000000000000000000', the 'Hexadecimal' field shows '7FA00000', and the 'Type' field shows 'Type: Quiet Nan'.

### [sNaN]

Number: sNaN

Exponent: 0

Expected Output:

Sign Bit: 1

Exponent: 11111111

Mantissa: 100 0000 0000  
0000 0000 0000

Hexadecimal: 7FC00000

Type: Signaling NaN

The screenshot shows the 'Number' field with 'sNaN' and the 'Exponent' field with '0'. The 'Convert' button is highlighted. Below the input fields, the 'Binary' field shows '0 11111111 100000000000000000000000', the 'Hexadecimal' field shows '7FC00000', and the 'Type' field shows 'Type: Signaling NaN'.

### [Zero]

Number: 0.0

Exponent: 0

Expected Output:

Sign Bit: 0

Exponent: 00000000

Mantissa: 000 0000 0000  
0000 0000 0000

Hexadecimal: 00000000

Type: Zero

The screenshot shows the 'Number' field with '0.0' and the 'Exponent' field with '0'. The 'Convert' button is highlighted. Below the input fields, the 'Binary' field shows '0 00000000 000000000000000000000000', the 'Hexadecimal' field shows '00000000', and the 'Type' field shows 'Type: Zero'.

## ERROR-CHECKING

### [Null Input]

Input: (null)

Expected Output: ERROR:  
Null Input

The screenshot shows the 'Number' field with 'Binary or Decimal or sNaN or qNaN' and the 'Exponent' field with 'Exponent'. The 'Convert' button is highlighted. Below the input fields, a red error message box displays 'ERROR: Null input'.



**[Invalid Input]**

Input: +1.0000.001

Expected Output: ERROR: Not a valid input

Number		Exponent	
+1.0000.001	× 2 (Binary) ▾	0	
			<div>Convert</div> <div>Clear</div>
ERROR: Not a valid input			

**[Invalid Binary Input]**

Input: 123

Expected Output: ERROR: Not a valid binary input

Number		Exponent	
123	× 2 (Binary) ▾	69	
			<div>Convert</div> <div>Clear</div>
ERROR: Not a valid binary input			

**[Invalid Decimal Input]**

Input: A2F4

Expected Output: ERROR: Not a valid decimal input

Number		Exponent	
A2F4	× 10 (Decimal) ▾	69	
			<div>Convert</div> <div>Clear</div>
ERROR: Not a valid decimal input			