

De La Salle University - Manila Term 2 AY. 2023-2024

In partial fulfillment of the course Introduction to Computer Organization

IEEE-754 Decimal-32 Floating Point Converter

Documentation

Submitted by:
Gonzaga, Eros Aneelouv
Hallar, Francine Marie
Hong, Letty
Lim, Justin Nathaniel
Milan, John Lloyd

Submitted to: Sir Roger Uy

Introduction

This repository contains a Decimal-32 Floating Point converter implemented in JavaScript. The converter includes functions of converting the input to binary and hexadecimal output and an option to choose a preferred rounding method, such as truncate, round up, round down, and round to nearest ties to even. It also includes a graphical user interface for interacting with the converter, such as a dropdown round method option, convert and clear button, and an export to text file button.

Project Specifications

IEEE-754 Decimal-32 Floating Point Converter including infinity and NaN special cases. It accepts a decimal and base-10 value that can handle more than 7 digits, and a rounding method option. The output handles binary and hexadecimal output with an option to output in a text file.

Implementation

In order to implement the Decimal-32 Floating Point converter, the floating point number first needs to be parsed and converted into values that an algorithm can work with. The goal of this process is to turn the two inputs (a fixed point base **b** and an exponent **e**, in the format **b** * 10^e) into a sign bit, 7 normalized digits (from **b**) and the corresponding normalized exponent. Shown below is a rough outline of the conversion process:

Given inputs **b**, a string, and **e**, an integer,

- 1. Check if **b** is "nan". If it is, display the decimal-32 value for NaN and exit. (This special case has to be checked now since the next line checks to see if **b** follows the format for a floating point number)
- 2. Check to make sure the input **b** is in the correct fixed point format. Fixed point data can start with a +, -, or no sign. Afterwards, any number of digits can be present (including zero, as .23 is still considered a valid floating point number for conversion), followed by an optional decimal point and a required sequence of digits of length 1 or more. This is accomplished using a regular expression.
- 3. Trim leading and trailing whitespace from **b**.
- 4. If **b** starts with -, set the sign bit to 1. Otherwise, set it to 0. Remove all instances of + or from the start of **b**.
- 5. Remove the decimal point from **b**. This operation is effectively the same as moving the decimal point to the right of **b**, increasing the value of **b**. To maintain equality, decrease **e** by the number of digits to the right of the decimal point.
- 6. Remove all leading and trailing zeros from **b**. Removing trailing zeros decreases **b**, so increase **e** by the number of trailing zeros removed.
- 7. If **b** is less than 7 digits, pad the value with zeros from the left so it is seven digits, then end the routine by returning the sign bit, **b** as the normalized digits, and **e** as the exponent.

- 8. Otherwise, let **whole** be the first 7 digits of **b**, and **frac** be the remaining digits. Prepend the digits **frac** with "0.", so that the digits represent a fraction between 0 and 1. Convert **frac** to a float and **whole** to an integer, then adjust the value of **whole** based on the rounding mode.
 - a. Truncate: Do nothing.
 - b. Round down: Add 1 to **whole** if the sign bit is 1.
 - c. Round up: Add 1 to **whole** if the sign bit is 0.
 - d. Nearest ties-to-even: If **frac** is more than 0.5, or **frac** is 0.5 and **whole** is odd, add 1 to **whole**
- 9. Remove all trailing zeros from **whole**. Add the number of zeros removed to **e**.
- 10. End the routine by returning the sign bit, **whole** as the normalized digits, and **e** as the exponent.

Let **digits** be the value of the 7 normalized digits returned by this routine.

To convert the values to decimal-32, the remaining special cases were handled. If \mathbf{e} is over 90, a check was made to see if shifting **digits** to the left would allow \mathbf{e} to equal 90 or less. For example, consider the case where **digits** = 0000001 and \mathbf{e} = 91. This would not normally be representable in decimal-32, but by adjusting **digits** to be 0000010 and \mathbf{e} to 90, it becomes possible to represent this number. These cases are handled in this manner, if possible.

If **e** is still too large, the value for infinity is returned.

If e is instead less than -101, return the decimal-32 value for 0.0 instead.

From here, we can assume that **digits** and **e** represent a normal case. Each part of the decimal-32 output is computed individually. First, to assist in the calculation, \mathbf{e}' is set as equal to $\mathbf{e} + 101$. Then, if the first digit of **digits** is 8 or 9, the combination field is set as "11" + the first two bits of \mathbf{e}' + the last bit of the first digit of **digits**. Otherwise, it is set as the first two bits of \mathbf{e}' + the last three bits of the first digit of **digits**.

The exponent continuation field is assigned to the remaining bits of **e**'. The last six **digits** are split into two groups of three and converted into Densely Packed Binary Coded Decimal through the following routine:

- 1. Copy the last bit of each digit to the third, sixth, and last bits of the answer, respectively.
- 2. If none of the digits are greater than 7, bit 7 is set to 0. Otherwise, set it to 1.
- 3. Set the remaining bits as follows:
 - a. Let **bc** be the second and third bits of the first digit.
 - b. Let **fg** be the second and third bits of the second digit.
 - c. Let jk be the second and third bits of the third digit.

d. Assign the remaining bits according to this table:

Is this digit 8 or 9?		Assign the following bits:			
First digit	Second Digit	Third Digit	Bits 1 and 2	Bits 4 and 5	Bits 8 and 9
No	No	No	bc	fg	jk
No	No	Yes	bc	fg	00
No	Yes	No	bc	jk	01
No	Yes	Yes	bc	10	11
Yes	No	No	jk	fg	10
Yes	No	Yes	fg	01	11
Yes	Yes	No	bc	00	11
Yes	Yes	Yes	00	11	11

The converted digits are then placed in the coefficient continuation field. Now that all the parts of the converted decimal-32 number are complete, the website is able to show the results of the calculation on the site.

Development Process and Considerations

Some technical limitations were encountered during the development process that influenced the design of the final conversion algorithm. One large issue encountered was the conversion process for the floating point input, since built-in algorithms for parsing strings representing floating point numbers converted them into IEEE 754 binary floats. As a result, parsing the user's input had to be done manually.

Another issue to be addressed was the enforcement of only having leading zeros for the value of **digits**. While values like 7 * 10^4 and 700 * 10^2 are the same value, converting these directly into decimal-32 results in different answers. To enforce a single canonical conversion value, the version of **digits** with no trailing zeros was preferred (if possible). Due to the potential of trailing zeros occurring both before and after rounding, they had to be removed multiple times in the input parsing procedure.

In some cases, it is impossible to use the version of **digits** with no leading zeros, such as in the case where e is over the maximum possible value representable by decimal-32. $7 * 10^92$ has a value of e that is too high to be represented, so $700 * 10^90$ must be used instead. The representation with the minimum number of trailing zeros is used.

Test Cases

Case Description	Input	Output	Expected Output	Pass/Fail
Normal positive less than 7	4.75 x 10^0	Binary Output: 0 01000 100011 0000000000 1001110101 Hexadecimal Output: 22300275	Binary Output: 0 01000 100011 0000000000 1001110101 Hexadecimal Output: 22300275	PASS
	Decimal-32 to Floating Point	DECIMAL-32 FLOATIN	G POINT CONVERTER	₹
	Rounding Method: Truncate	v	Binary Output	
	Enter a decimal number:	x 10^ 20	Hexadecimal Output	
	CONVERT CLEAR			EXPORT TO TEXT FILE
	Binary outpu	esults for 4.7	75 x 10^0 to de 9011 00000000000	
	123.45 x 10^2	Binary Output: 0 01000 100101 0000010010 0111000101	Binary Output: 0 01000 100101 0000010010 0111000101	PASS
		Hexadecimal Output:	Hexadecimal Output:	

		225049C5	225049c5		
	Decimal-32 FLOATING POINT CONVERTER Decimal-32 to Floating Point				
	Rounding Method: Truncate Enter a decimal number: 123.45 CONVERT CLEAR	x 10 [^] 2	Binary Output 0 01000 100101 0000010010 0111000101 Hexadecimal Output 225049c5	EXPORT TO TEXT FILE	
	Binary outp	results for 12	3.45 x 10^2 to 0101 000001001 9c5		
Normal negative less than 7	-9875521 x 10^0	Binary Output: 1 11011 100101 1001111101 1010100001 Hexadecimal Output: EE59F6A1	Binary Output: 1 11011 100101 1001111101 1010100001 Hexadecimal Output: ee9f6a1	PASS	
	Decimal-32 FLOATING POINT CONVERTER Decimal-32 to Floating Point				
	Rounding Method: Truncate Enter a decimal number: -9875521 CONVERT CLEAR	x 10 ^A 0	Binary Output 1 11011 100101 1001111101 1010100001 Hexadecimal Output ee59f6a1	EXPORT TO TEXT FILE	

Exporting to text file: Conversion results for -9875521 x 10⁰ to decimal32 Binary output: 1 11011 100101 1001111101 1010100001 Hexadecimal output: ee59f6a1 -10.87546 x 10[^] Binary Output: Binary Output: **PASS** -20 1 01001 01001 001100 001100 0001101011 0001101011 1011000110 1011000110 Hexadecimal Hexadecimal Output: Output: A4C1AEC6 a4c1aec6 **DECIMAL-32 FLOATING POINT CONVERTER** Binary Output 1 01001 001100 0001101011 1011000110 Rounding Method: Truncate Enter a decimal number: Hexadecimal Output **x 10^** -20 a4c1aec6 CONVERT CLEAR EXPORT TO TEXT FILE Exporting to text file: Conversion results for -10.87546 x 10^-20 to decimal32 Binary output: 1 01001 001100 0001101011 1011000110 Hexadecimal output: a4c1aec6 Binary Output: Binary Output: Normal **PASS** 10.8754678 x 10^20 positive 0 0 more than 7 01001 01001 round by 110110 110110 truncation 0001101011 0001101011 1011000111 1011000111

Hexadecimal

Hexadecimal

	Output: 2741AEC6	Output: 2741aec6	
	DECIMAL-32 FLOATII	ng Point Convertei	R
Decimal-32 to Floating Point			
		Binary Output	
Rounding Method: Truncate	~	0 01001 110100 0001101011 1011000110	
Enter a decimal number:		Hexadecimal Output	
10.8754678	x 10^ 20	2741aec6	
	1		
CONVERT CLEAR			EXPORT TO TEXT FILE
xporting to text	file:		
		754678 x 10^20 t	o decimal32
Conversion re	sults for 10.87	754678 x 10^20 t 90 0001101011 10	
Conversion re Binary output	sults for 10.87	00 0001101011 10	
Conversion re Binary output	esults for 10.87 :: 0 01001 11016	00 0001101011 10	
Conversion re Binary output Hexadecimal c	esults for 10.87 :: 0 01001 11016	00 0001101011 10	
Conversion re Binary output Hexadecimal c	esults for 10.87 : 0 01001 11016 output: 2741aec6 Binary Output:	Binary Output:	11000110
Conversion re Binary output Hexadecimal o	Binary Output:	Binary Output: 0 11011	11000110
Binary output Hexadecimal c	Binary Output: 0 11011 11111	Binary Output: 0 11011 11111	11000110
Conversion re Binary output Hexadecimal o	Binary Output: 0 11011 1101111101	Binary Output: 0 11011 11111 1101111101	11000110
Conversion re Binary output Hexadecimal o	Binary Output: 0 11011 11111	Binary Output: 0 11011 11111	11000110
Conversion re Binary output Hexadecimal o	Binary Output: 0 11011 1101111101 1101010100	Binary Output: 0 11011 11111 1101111101 1101010100	11000110
Conversion re Binary output Hexadecimal o	Binary Output: 0 11011 1101111101	Binary Output: 0 11011 11111 1101111101	11000110

		DECIMAL-32 FLOATIN	G POINT CONVERTER	:
	Fixed Point Base 10 to Decimal-32 Floating Po	pint		
			Binary Output	
	Rounding Method: Truncate	~	0 11011 101000 11011111101 1101010100	
	Enter a decimal number:		Hexadecimal Output	
	9877654500	x 10^ 0	6e8df754	
	CONVERT CLEAR			EXPORT TO TEXT FILE
	Exporting to text f	ĩle:		
	Conversion re Binary output	sults for 98776	54500 x 10^0 to 0 1101111101 110	
Normal positive more than 7 round up	10.8754678 x 10^20	Binary Output: 0 01001 110100 0001101011 1011000111 Hexadecimal Output: 2741AEC7	Binary Output: 0 01001 110100 0001101011 1011000111 Hexadecimal Output: 2741aec7	PASS
	Decimal-32 to Floating Point	DECIMAL-32 FLOATIN	AG POINT CONVERTER	2
			Binary Output	
	Rounding Method: Round Up	~	0 01001 110100 0001101011 1011000111	
	Enter a decimal number:			
	10.8754678	x 10^ 20	Hexadecimal Output	
	CONVERT CLEAR		E/M10CL/	EXPORT TO TEXT FILE

Exporting to text file: Conversion results for 10.8754678 x 10^20 to decimal32 Binary output: 0 01001 110100 0001101011 1011000111 Hexadecimal output: 2741aec7 9877654500 x Binary Output: Binary Output: **PASS** 10^0 0 11011 11011 111111 111111 1101111101 1101111101 1101010101 1101010101 Hexadecimal Hexadecimal Output: Output: 6E8DF755 6e8df755 **DECIMAL-32 FLOATING POINT CONVERTER** 0 11011 101000 1101111101 1101010101 Rounding Method: Round Up Enter a decimal number: Hexadecimal Output x 10^ o 9877654500 6e8df755 CONVERT CLEAR EXPORT TO TEXT FILE Exporting to text file: Conversion results for 9877654500 x 10^0 to decimal32 Binary output: 0 11011 101000 1101111101 1101010101 Hexadecimal output: 6e8df755 Normal 10.8754678 x Binary Output: Binary Output: **PASS** positive 10^20 0 0 01001 01001 more than 7 round down 110110 110110 0001101011 0001101011 1011000111 1011000111 Hexadecimal Hexadecimal Output: Output: 2761AEC6 2761aec6

Decimal-32 to Floating Point	DECIMAL-32 FLOATIN	IG POINT CONVERTER	2
Rounding Method: Round Down Enter a decimal number: 10.8754678 CONVERT CLEAR	x 10^ 20	Binary Output 0 01001 110100 0001101011 1011000110 Hexadecimal Output 2741aec6	EXPORT TO TEXT FILE
Binary output	sults for 10.87	54678 x 10^20 to 0 0001101011 10:	
9877654500 x 10^0	Binary Output: 0 11011 111111 1101111101 110101000 Hexadecimal Output: 6E8DF754	Binary Output: 0 11011 111111 1101111101 1101010100 Hexadecimal Output: 6e8df754	PASS
Fixed Point Base 10 to Decimal-32 Floating Point	DECIMAL-32 FLOATII	ng Point Converter	
Rounding Method: Round Down Enter a decimal number: 9877654500 CONVERT CLEAR	x 10 [^] 0	Binary Output 0 11011 101000 1101111101 1101010100 Hexadecimal Output 6e8df/54	EXPORT TO TEXT FILE

Exporting to text file: Conversion results for 9877654500 x 10^o0 to decimal32 Binary output: 0 11011 101000 1101111101 1101010100 Hexadecimal output: 6e8df754 Normal 10.8754678 x Binary Output: **PASS** Binary Output: positive 10^20 0 0 01001 01001 more than 7 110100 110100 round to nearest ties 0001101011 0001101011 1011000111 1011000111 even Hexadecimal Hexadecimal Output: Output: 2741AEC7 2741aec7 **DECIMAL-32 FLOATING POINT CONVERTER** 0 01001 110100 0001101011 1011000111 Rounding Method: Round to Nearest Ties Even ~ Enter a decimal number: Hexadecimal Output **x 10^** 20 10.8754678 2741aec7 CONVERT CLEAR EXPORT TO TEXT FILE Exporting to text file: Conversion results for 10.8754678 x 10^20 to decimal32 Binary output: 0 01001 110100 0001101011 1011000111 Hexadecimal output: 2741aec7 **PASS** 9877654500 x Binary Output: Binary Output: 10^0 0 0 11011 11011 111111 111111 1101111101 1101111101 1101010100 1101010100 Hexadecimal Hexadecimal Output: Output:

		6FFDF754	6ffdf754	
		DECIMAL-32 FLOATII	ng point converter	
	Rounding Method: Round to Nearest Ties E Enter a decimal number: 9877654500 CONVERT CLEAR	x 10 [^] 0	Binary Output 0 11011 101000 1101111101 1101010100 Hexadecimal Output 6e8df754	EXPORT TO TEXT FILE
	Exporting to text f	ĭle:		
	Binary outpu	esults for 9877 t: 0 11011 1016 output: 6e8df7	000 1101111101	
Normal negative more than 7 round by truncation	-23.3486129 x 10^24	Binary Output: 1 01010 111000 0110110100 0001101101 Hexadecimal Output: AB86D06D	Binary Output: 1 01010 111000 0110110100 0001101101 Hexadecimal Output: ab86d06d	PASS
		Decimal-32 FLOATin	G POINT CONVERTER	
	Rounding Method: Truncate Enter a decimal number: -23.3486129 CONVERT CLEAR	x 10^ 24	Binary Output 1 01010 111000 0110110100 0001101101 Hexadecimal Output ab86606d	EXPORT TO TEXT FILE

Exporting to text file: Conversion results for -23.3486129 x 10^24 to decimal32 Binary output: 1 01010 111000 0110110100 0001101101 Hexadecimal output: ab86d06d Binary Output: Binary Output: -0.66728995 x **PASS** 10^-12 01110 01110 010010 010010 1101110010 1101110010 0001111111 0001111111 Hexadecimal Hexadecimal Output: Output: B92DC87F b92dc87f **DECIMAL-32 FLOATING POINT CONVERTER** Fixed Point Base 10 to Decimal-32 Floating Point Binary Output 1 01110 010010 1101110010 0001111111 Rounding Method: Truncate Enter a decimal number: Hexadecimal Output x 10[^] -12 b92dc87f CONVERT CLEAR EXPORT TO TEXT FILE Exporting to text file: Conversion results for -0.66728995 x 10^-12 to decimal32 Binary output: 1 01110 010010 1101110010 0001111111 Hexadecimal output: b92dc87f Normal -23.3486129 x Binary Output: Binary Output: **PASS** negative 10^24 more than 7 01010 01010 round up 111000 111000 0110110100 0110110100 0001101101 0001101101 Hexadecimal Hexadecimal Output: Output: AB86D06D ab86d06d

DECIMAL-32 FLOATING POINT CONVERTER			
Rounding Method: Round Up Enter a decimal number: -23.3486129 CONVERT CLEAR	x 10^ 24 ile:	Binary Output 1 01010 111000 0110110100 0001101101 Hexodecimal Output absoluted	EXPORT TO TEXT FILE
Binary output	: 1 01010 11100	0 0110110100 00	
-0.66728995 x 10^-12	Binary Output: 1 01110 010010 1101110010 0001111111 Hexadecimal Output: B92DC87F	Binary Output: 1 01110 010010 1101110010 0001111111 Hexadecimal Output: b92dc87f	PASS
	DECIMAL-32 FLOATII	ng Point Converter	
Rounding Method: Round Up Enter a decimal number: -0.66728995 CONVERT CLEAR	x 10 [^] -12	Binary Output 1 01110 010010 1101110010 0001111111 Hexodecimal Output b92d:87f	EXPORT TO TEXT FILE
	Enter a decimal number: -23.346129 CONVERT CLEAR CONVERT CLEAR CONVERT CLEAR CONVERT CLEAR -0.66728995 x 10^-12 Fixed Point Base 10 to Decimal-32 Floating Point Rounding Method: Round Up Enter a decimal number: -0.66728995	Exporting to text file: CONVERT CLEAR CONVERT CLEAR CONVERT CLEAR Exporting to text file: CONVERT CLEAR CONVERT CLEAR CONVERT CLEAR Exporting to text file: CONVERT CLEAR CONVERT CLEAR Exporting to text file: CONVERT CLEAR A 10 1110 Hexadecimal output: ab86d06d -0.66728995 x 10^-12 Binary Output: 1 01110 010010 1101110010 010010 1101110010 000111111	Rinery Output State Stat

Exporting to text file: Conversion results for -0.66728995 x 10^-12 to decimal32 Binary output: 1 01110 010010 1101110010 0001111111 Hexadecimal output: b92dc87f Normal -23.3486129 x Binary Output: PASS Binary Output: negative 10^24 more than 7 01010 01010 round down 111000 111000 0110110100 0110110100 0101101100 0101101100 Hexadecimal Hexadecimal Output: Output: AB86D16C ab86d16c **DECIMAL-32 FLOATING POINT CONVERTER** Binary Output 1 01010 111000 0110110100 0101101100 Enter a decimal number: x 10^ 24 -23.3486129 CONVERT CLEAR Exporting to text file: Conversion results for -23.3486129 x 10^24 to decimal32 Binary output: 1 01010 111000 0110110100 0101101100 Hexadecimal output: ab86d16c -0.66728995 x Binary Output: Binary Output: PASS 10^-12 01000 01000 010100 010100 0001100110 0001100110 1110101001 1110101001 Hexadecimal Hexadecimal Output: Output:

		A1419BA9	a1419ba9	
		Decimal-32 Floatii	ng Point Converter	
		sults for -0.66		
		: 1 01000 01010 output: a1419ba9	0 0001100 110 111	19101001
Normal negative more than 7 round to nearest ties to even	-23.3486129 x 10^24	Binary Output: 1 01010 111000 0110110100 0001101101 Hexadecimal Output: AB86D06D	Binary Output: 1 01010 111000 0110110100 0001101101 Hexadecimal Output: ab86d06d	PASS
		Decimal-32 Floatin	AG POINT CONVERTER	
	Rounding Method: Round to Nearest Ties Even Enter a decimal number: 23.3466129 CONVERT CLEAR	x 10^ 24	Binary Output 1 01010 111000 0110110100 0001101101 Hexadecimal Output abblished	EXPORT TO TEXT FILE
	Rounding Method: Round to Nearest Ties Even Tenter a decimal number:	Hexadecimal Output: AB86D06D Decimal-32 Floating	Hexadecimal Output: ab86d06d DG POINT CONVERTER Binary Output 101010 111000 011011010 0001101101	EXPORT TO TEXT F

Exporting to text file: Conversion results for -23.3486129 x 10^24 to decimal32 Binary output: 1 01010 111000 0110110100 0001101101 Hexadecimal output: ab86d06d -0.66728995 x Binary Output: Binary Output: PASS 10^-12 01000 01000 010100 010100 0001100110 0001100110 1110101001 1110101001 Hexadecimal Hexadecimal Output: Output: A1419BA9 a1419ba9 **DECIMAL-32 FLOATING POINT CONVERTER** Binary Output 1 01000 010100 0001100110 1110101001 Rounding Method: Round to Nearest Ties Even 🗸 Enter a decimal number: Hexadecimal Output x 10[^] -12 -0.66728995 a1419ba9 CONVERT CLEAR EXPORT TO TEXT FILE Exporting to text file: Conversion results for -0.66728995 x 10^-12 to decimal32 Binary output: 1 01000 010100 0001100110 1110101001 Hexadecimal output: a1419ba9 NaN Binary Output: PASS Nan Binary Output: NaN 0 0 11111 000000 11111 000000 nan 0000000000 0000000000 nAn nAN 0000000000 000000000 **NAN** Hexadecimal Hexadecimal Output: Output: 7C000000 7c000000

		DECIMAL-32 FLOATII	ng point converter	
	Freed Point Base 10 to Decimal-32 Floating Point Rounding Method: Truncate Enter a decimal number: nan CONVERT CLEAR	x 10 ^A Enter exponent value	Binary Output 0 11111 000000 000000000 0000000000 Hexadecimal Output 7c000000	EXPORT TO TEXT FILE
	Binary output	sults for Nan x	10^ to decimal3: 0 00000000000 000	
Positive Infinity	1234567 x 10^91	Binary Output: 0 11110 000000 00000000000 0000000000	Binary Output: 0 11110 000000 0000000000 0000000000	PASS
	Fixed Point Base 10 to Decimal-32 Floating Point		ng Point Converter	
	Rounding Method: Round to Nearest Ties E Enter a decimal number: 1234567 CONVERT CLEAR	x 10 [^] 91	Binary Output 0 11110 000000 000000000 0000000000 Hexadecimal Output 78000000	EXPORT TO TEXT FILE

Exporting to text file: Conversion results for 1234567 x 10^91 to decimal32 Binary output: 0 11110 000000 0000000000 0000000000 Hexadecimal output: 78000000 -1234567 x Binary Output: PASS Binary Output: 10^111 11110 11110 000000 000000 000000000 000000000 000000000 000000000 Hexadecimal Hexadecimal Output: Output: F8000000 f8000000 **DECIMAL-32 FLOATING POINT CONVERTER** Binaru Output 1 11110 000000 0000000000 0000000000 Rounding Method: Round to Nearest Ties Even Enter a decimal number: **x 10^** 91 -1234567 CONVERT CLEAR EXPORT TO TEXT FILE Exporting to text file: Conversion results for -1234567 x 10^111 to decimal32 Binary output: 1 11110 000000 000000000 00000000000 Hexadecimal output: f8000000 Negative PASS 1234567 x Binary Output: Binary Output: **Infinity** 10^-102 0 0 01000 01000 100101 100101 000000000 000000000 0000000000 000000000 Hexadecimal Hexadecimal Output: Output: 2500000 2500000

	DECIMAL-32 FLOATING	POINT CONVERTER
red Point Base 10 to Decimal-32 Floating Point		
Rounding Method: Round to Nearest Ties Even ✓		Binary Output 0 01000 100101 000000000 000000000
inter a decimal number:		Hexadecimal Output
1234567 CONVERT CLEAR	x 10 [^] -102	22500000 EXPORT TO TEXT FILE
		EAFORT TO TEAT TILL.
xporting to text file):	
		67 x 10^-102 to decimal32
	0 01000 10010: tput: 22500000	1 0000000000 0000000000