1.) IEEE information- the IEEE specifications are quite detailed over the course of 70 pages, so in summary:

a.) IEEE floating points come in 7 different formats:

Name	Common name	Base	Digits	E min	E max	Notes	Decimal digits	Decimal E max
binary16	Half precision	2	10+1	−14	+15	storage, not basic	3.31	4.51
binary32	Single precision	2	23+1	−126	+127		7.22	38.23
binary64	Double precision	2	52+1	−1022	+1023		15.95	307.95
binary128	Quadruple precision	2	112+1	−16382	+16383		34.02	4931.77
decimal32		10	7	−95	+96	storage, not basic	7	96
decimal64		10	16	−383	+384		16	384
decimal128		10	34	−6143	+6144		34	6144

b.) There are 8 operations that can be performed on them:

i.) Arithmetic operations (add, subtract, multiply, divide, square root, fused multiply–add, remainder, etc.)

ii.) Conversions (between formats, to and from strings, etc.)

iii.) Scaling and (for decimal) quantizing.

iv.) Copying and manipulating the sign (abs, negate, etc.)

v.) Comparisons and total ordering.

vi.) Classification and testing for NaNs, etc.

vii.) Testing and setting flags.

viii.) Miscellaneous operations.

c.) There are 5 possible exceptions

i.) Invalid operation (e.g., square root of a negative number) **(returns qNaN by default).**

ii.) Division by zero (an operation on finite operands gives an exact infinite result, e.g., 1/0 or log(0)) **(returns ±infinity by default).**
iii.) Overflow (a result is too large to be represented correctly) **(returns ±infinity by default (for round-to-nearest mode)).**
iv.) Underflow (a result is very small (outside the normal range) and is inexact) **(returns a denormalized value by default).**
v.) Inexact **(returns correctly rounded result by default).**

d.) Notes:

i.) **Conversions to and from a decimal character** format are required for all formats. Conversion to an external character sequence must be such that conversion back using round to even will recover the original number

ii.) The original binary value will be preserved by **converting to decimal and back again** using:

1.) 5 decimal digits for binary16
2.) 9 decimal digits for binary32
3.) 17 decimal digits for binary64
4.) 36 decimal digits for binary128

iii.) When using a decimal floating point format the decimal representation will be preserved using:

2.) How to Solve Quadratics!

3.) Spikes

```
/*
Programming Standard
- subversion
- Unit Testing: Cunit
- Automation using make
- Documentation standard (Things to look at: doxygen, IEEE (wiki))
*/

/*
PROGRAM NEEDS
needs to work on most Linux systems
given an "a" "b" "c" in float pofloat need to return "x1" "x2"
8 digits of accuracy (8 digits need to be correct, it's impossible) and precision

x = [b +-  sqrt(b^2 - 4ac)] / 2a

gcc compiler (which one? more current but doesn't need to be newest)
```

```
R2
run in shell script and return the values
*/


/*      Quadratic Equation */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

float QuadraticPlus(float, float, float);
float QuadraticMinus(float, float, float);

float main()
{
        float a, b, c;
        float x1, x2;

        a = 1.1;
        b = 4.8;
        c = 3.1;

        //Calculate them bad boys
        x1 = QuadraticPlus(a, b, c);
        x2 = QuadraticMinus(a, b, c);

        //Now to output to terminal
        printf("The quadratic result for a = %f, b = %f, c = %f is:\nx1 = %.8f\nx2 = %.8f\n\n", a, b, c, x1, x2);

        //Grab a beer, job well done
        return 0;
}

//---------------------------------
// Calculate the addition
// part of the quadratic formula
//---------------------------------
float QuadraticPlus(float a, float b, float c)
{
        float x1;
```

```
	if((pow(b, 2) - 4. * a * c) > 0)
	{
		x1 = (-b + sqrt(pow(b, 2) - 4. * a * c)) / (2. * a);
	}
	else
	{	printf("The result is a non-real number\n");	}

	return x1;
}

//---------------------------------
// Calculate the subtraction
// part of the quadratic formula
//---------------------------------
float QuadraticMinus(float a, float b, float c)
{
	float x2;
	if((pow(b, 2) - 4. * a * c) > 0)
	{
		x2 = (-b - sqrt(pow(b, 2) - 4. * a * c)) / (2. * a);
	}
	else
	{	printf("The result is a non-real number\n");	}

	return x2;
}
```