Programming Assignment # 1

Quadruple Precision Data Type

1. Objectives

Your goal is implement the 128-bit long floating point data representation scheme, and its associated multiplication and addition operation functions with double precision inputs as well as type cast operation to the double precision data type.

2. Details

This 128 bit quadruple precision is designed not only for applications requiring results in higher than double precision, but also, as a primary function, to allow the computation of double precision results more reliably and accurately by minimising overflow and round-off errors in intermediate calculations and scratch variables: as William Kahan, primary architect of the original IEEE-754 floating point standard noted, "For now the 10-byte Extended format is a tolerable compromise between the value of extra-precise arithmetic and the price of implementing it to run fast; very soon two more bytes of precision will become tolerable, and ultimately a 16-byte format... That kind of gradual evolution towards wider precision was already in view when IEEE Standard 754 for Floating-Point Arithmetic was framed."\(^1\)

Unfortunately, the current GCC compiler on the x86 architecture does not provide the quadruple precision data type. Therefore, you should implement your own.

A quadruple precision variable is 128-bit wide. In C, a quadruple precision variable is represented with a structure holding a character array of 16 as follows:

```c
struct quadruple {
    char data[16];
};
typedef struct quadruple quadruple;
```

The internal structure of the quadruple type consists of 1 bit for sign, 15 bits for exponent and 112 bits for significand, from MSB to LSB. They are organized in the array, `data`, in the little-endian form.

There are two arithmetic operation functions associated with this data type.

```c
/* multiplication of two double variables */
quadruple mult(double op1, double op2);

/* addition of two double variables */
quadruple add(double op1, double op2);
```

The result quadruple value can be converted back to the double precision data type with the following function.

---

\(^1\) From Wikipedia
These functions should be written in a separate C file and the prototype of the function must be provided in header file "quad.h".

We need a method to output the value stored in a quadruple type variable. The following function transforms the value of a quadruple type variable to a corresponding character string. The memory space for the result array will be allocated by the function.

```c
double qtod(quadruple src);
```

In addition to these functions, you should write a simple test program that can show the correctness of your implementation. The test program gets two double precision values, then conducts both arithmetic operations with the input values, and prints out the results.

The test program should be compiled easily by using the "make" command. Naturally, you submission must include a Makefile script.

Finally, you will write a brief report that explains your implementation.

### 3. Logistics

A. Make sure that you have included your name and ID in the header comment of your code.

B. All relevant files must be zipped into a file. The zipped file name must be "studentid.c" (e.g. 2013310123.zip)

C. The report should be in the PDF format and included in the zip file. The document file name should be "studentid.pdf" (e.g. 2012310123.pdf)

D. Submit your zipped file via i-Campus.

E. The due data is April 25th. Only the assignments submitted before the deadline will receive the full credit. 25% of the credit will be deducted for every single day delay.