Programming Assignment # 1

Mediocre Precision Floating Point Data Type

1. Objectives

Design and implement the 48-bit floating point data type and its associated multiplication and addition operations. Write a program that gets two mediocre data through keyboard input and prints both multiplication and addition of them in the bit-level representation.

2. Details

32-bit *float* is too coarse for scientific computation. 64-bit *double* is too big. We want a compromise between float and double. Therefore, here we design and implement a new data type, named "mediocre precision floating point data type".

A mediocre precision variable is 48-bit wide. In C, a mediocre precision variable is represented with a structure holding a character array of six as follows:

```
struct mediocre {
      char data[6];
};
typedef struct mediocre mediocre;
```

The internal structure of the mediocre type consists of 1 bit for sign, 9 bit for exponent and 38-bit for significand, from MSB to LSB. They are organized in the array, *data*, in little-endian form.

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

```
/* multiplication of two mediocre variables */
mediocre mult(mediocre op1, mediocre op2);

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

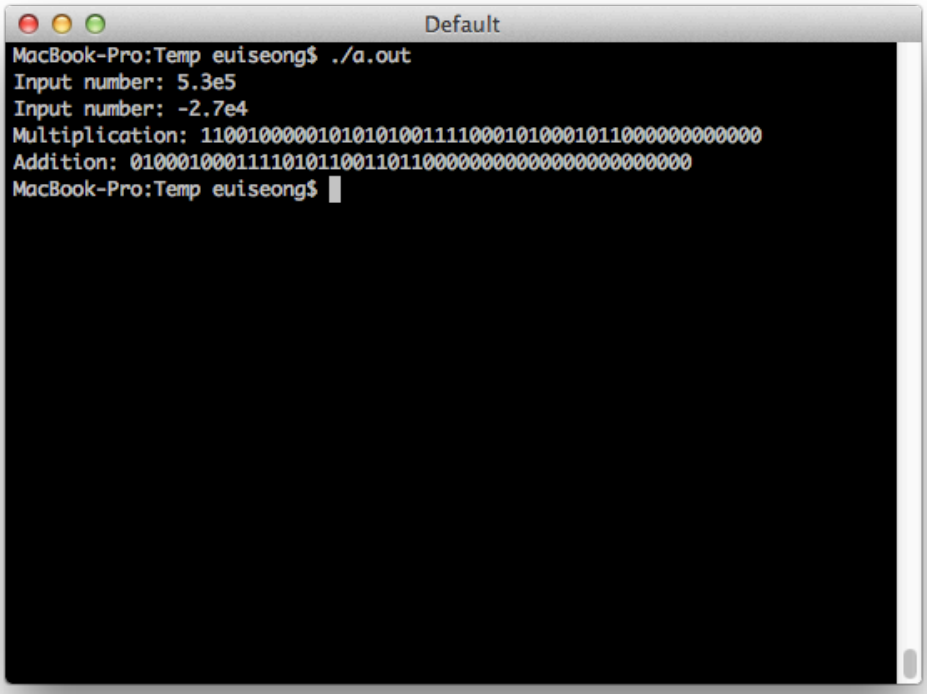
Because your program needs to obtain keyboard input for two mediocre operands and print the result of multiplication and addition operations, you have to write following functions for input/output:

```
/* obtaining a mediocre operand through keyboard */
int key_in(mediocre *op);

/* printing bit-level representation of a mediocre variable */
int print_bits(mediocre op);
```

Restriction: **A.** You can't use *double* or *float* typed variables in your code except the *key_in* function. **B.** Your data must be aligned in little endian so that the LSB must be accommodated in data[0] and the MSB in data[5].

3. Example



4. Hints

A. There are no standard ways to reinterpret "double precision" as "long long int" (casting will change the bit sequence). So, I would recommend to use "memcpy" function, which copies the contents of a memory area to another memory area. An example code is listed below.

```
#include <string.h>
.....
long long int casted;
double input;

printf("Input number: ");
scanf("%lf", &input);
memcpy(&casted, &input, sizeof(double));
```

B. The basic algorithm of the *key_in* function is getting the input as a *double* typed data and casting it to the mediocre type manually. During the procedure, you supposedly round significand of the double-typed data. The easiest round to implement in C is "round toward zero". You are allowed to use "round toward zero" as your rounding method.

5. Logistics

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

B. The source file name should be "studentid.c" (e.g. 2011310123.c)

C. Prepare a separate document in PDF format, which explains the design and implementation of your code. The document file name should be "studentid.pdf" (e.g. 2011310123.pdf)

D. Send a mail to [*homework.skku@gmail.com*] with attaching the two files, source code and documentation. The subject of the mail should be [PA#1] studentID.

E. The submission status will be posted on the course home page at http://csl.skku.edu/CSE2003S12

F. Only the assignments submitted before the deadline will receive the full credit. 25% of the credit will be deducted for every single day delay.