C Programming II
2022 Spring
Homework 02

Instructor: Po-Wen Chi

Due: 2022.3.22 PM 11:59

Policies:

- **Zero tolerance** for late submission.
- **Plagiarism is not allowed.** Both source and copycat will be **zero**.
- You need to prepare a README file about how to make and run your program. Moreover, you need to provide your name and your student ID in the README file.
  - Your Name and Your ID.
  - The functional description for each code.
  - Anything special.
- Please pack all your submissions in one zip file. **RAR is not allowed!!**
- For convenience, your executable programs must be named following the rule hwXXYY, where the red part is the homework number and the blue part is the problem number. For example, hw0102 is the executable program for homework #1 problem 2.
- I only accept **PDF**. MS Word is not allowed.
- Do not forget your Makefile. For convenience, each assignment needs only one Makefile.

1 Wildcard Matching (20 pts)

Undoubtedly, C standard string library provides some string matching functions, like **strcmp, strstr**. However, sometimes we want a string matching function that supports **patterns** instead of exactly words. Now I introduce a pattern called **wildcard**. First, you need to learn two symbols:

- **?**: Matches any single character.

- *: Matches any sequence of characters (including the empty sequence).

For example, given a pattern **a?e**:

- **ae** does not match the pattern.
- **ace** matches the pattern.
- **ache** does not match the pattern.

For example, given a pattern **a*e**:

- **ae** matches the pattern.
- **aaae** matches the pattern.
- **baaae** does not match the pattern.
- **ace** matches the pattern.
- **ache** matches the pattern.
- **apple** matches the pattern.

Now, given a pattern and an input string, please find all the words in the given string that match the pattern. You can assume all words are separated by a single space. For your convenience, all test strings are composed of English lowercase alphabets only. The string is less than 2048 bytes.

```
int mymatch(char ***pppList, const char *pStr, const char *pPattern);
```

If there is any invalid input, return -1; otherwise, return the number of matching words. You need to prepare **mymatch.h**, which can be the same file with the last problem, and TA will prepare **hw0201.c**. Of course, Makefile is your own business. Do not forget to make **hw0201.c** in your Makefile.

2 IEEE 754 (20 pts)

You all know what IEEE 754 is, right? The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point arithmetic established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). If you forget what it is, I hope figure 1 can arouse your memory.

Now I want you to develop a program for the user to input a double floating-point number and display the number as **sign, exponent, fraction**.

```
$ ./hw0202
Please enter a floating-point number (double precision): 1.75
Sign: 0
Exponent: 01111111111
Fraction: 1100000000000000000000000000000000000000000000000000
1.75 = (-1)^0 * (1 + 2^-1 + 2^-2) * 2^(1023-1023)
```

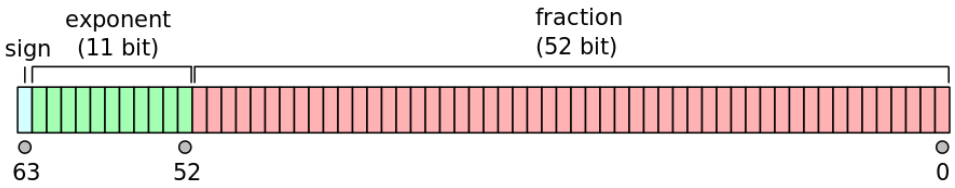

Figure 1: Double-precision floating-point format.

3 Puella Magi Madoka Magica (20 pts)

Many people like Object-Oriented Programming (OOP) language, like C++. If you do not know what OOP is, you can take Prof. Chiang and Prof. Lin courses in the future.

TA Peng is very bored and he wants to make a game about "Puella Magi Madoka Magica", which you can see figure 2 for the reference. I strongly recommend you to read some related information before you do this assignment. He wants you to develop structures for characters.

Figure 2: Puella Magi Madoka Magica.

There are three types of characters, which all inherit a base structure called **Entity**:

1. Entity

- hp（生命值）: default 100

2. Shoujo（少女）
   - kimoji（心情）: default 100
   - despair: hp is set to zero
   - do_wish: print wish
   - is_despair: check if kimoji is $\leq$ -100
3. Mahoushoujo（魔法少女）
   - atk（攻擊力）: default 100
   - hp: Shoujo's hp $\times$ 3
   - do_wish：print the wish defined in Shoujo and print "But nothing is good". Then, kimoji -= 10.
   - despair：print "Watashii de, hondo baga" and turn itself to Mojo.
   - skill：Special skill for different girls.
4. Majo（魔女）
   - atk（攻擊力）: default 30
   - hp: Shoujo's hp $\times$ 50
   - despair: do nothing
   - kekkai（結界）：The target Shoujo'kimoji - 100

There are four different Mahoushoujos, who are listed here:

1. Madoka（小圓）: print "Madoka become god, end." and terminate the process.
2. Homura（暁美焰）: if her HP is $<$ 50, print "This round is hopeless, Homura go to next round." and terminate the process.
3. Sayaka（阿爽）: her hp + 30 and kimoji - 30
4. Kyoko （京子）: If the target is Sayaka, both are dead.

It sounds very complex, right? Do not worry! Actually, our kind TA prepares **madoka.h** for you. **You are not allowed to change this!** The only thing that you need to do is to write **madoka.c** for implementing these functions. TA Peng will generate a **hw0203.c** for you. **Do not forget to build hw0203.c in your Makefile**.

Note: This assignment is not difficult. I want you to use this chance to **think** why TA Peng writes code like this and do you have any better way to implement the similar things?

Note2: This assignment may make you well-prepared for your final project.

Note3: Frankly speaking, there are lots of missing information. Please contact TA Peng.

4 Mixed Fraction Arithmetic (20 pts)

A mixed fraction is a traditional denotation of the sum of an integer and a proper fraction. There are some examples:

$$1\frac{5}{7}, -3\frac{2}{9}\dots$$

This time, I want you to develop a mixed fraction calculator. You need to make user input an equation and then output the answer. To input a mixed fraction, we follow the Latex style. For example, we can use **1\frac{2}{3}** to represent $1\frac{2}{3}$. You need to support **addition +, subtraction -, multiplication *, division /**. Of course, you must follow arithmetic operation precedence. For simplicity, you do not need to consider parentheses and all input numbers are **int8_t**. Your answer must follows the following rules. Given $a\frac{b}{c}$,

- $|c| > |b|$.
- $\frac{b}{c}$ must be the reduced form.
- b, c must be a positive integer.
- 0 is presented as 0.

There is an example. If a user wants to get the result of the following equation:

$$\frac{1}{2} + 1\frac{5}{6} \times 2\frac{3}{10} = 4\frac{43}{60}$$

```
$ ./hw0204
Q: \frac{1}{2}+1\frac{5}{6}*2\frac{3}{10}
A: 4\frac{43}{60}
```

You **MUST** design a structure for the mixed fraction, design related functions in some header file, and implement these functions in a C file other than your main function. If the user input is invalid, just print a warning message and terminate the program. You need to implement at least these functions and undoubtedly, you can design more functions as your need. For your convenience, the input string's length will be less than 4096.

```
typedef struct _sMixedNumber {
}sMixedNumber;

void mixed_add( sMixedNumber *pNumber, const sMixedNumber r1, const
    sMixedNumber r2);
// pNumber = r1 + r2
void mixed_sub( sMixedNumber *pNumber, const sMixedNumber r1, const
    sMixedNumber r2);
// pNumber = r1 - r2
void mixed_mul( sMixedNumber *pNumber, const sMixedNumber r1, const
    sMixedNumber r2);
// pNumber = r1 * r2
void mixed_div( sMixedNumber *pNumber, const sMixedNumber r1, const
    sMixedNumber r2);
// pNumber = r1 / r2
```

5 Vector (20 pts)

You all know what a vector is, right? A vector is a quantity or phenomenon that has two independent properties: magnitude and direction. A vector is usually described in terms of their components in a coordinate system. I think you also know that there are multiple coordinate systems, like Cartesian coordinates and polar coordinates. For your reference, you can see figure 3.

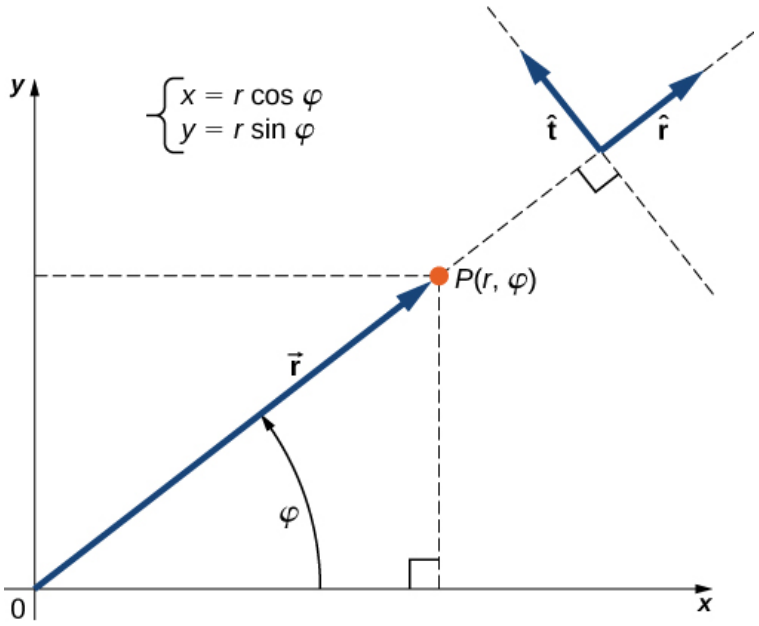

Figure 3: Polar Coordinate vs. Cartesian Coordinate. Source: https://courses.lumenlearning.com/suny-osuniversityphysics/chapter/2-2-coordinate-systems-and-components-of-a-vector/

This time, I want you to implement the following APIs for a vector structure defined by me.

```
typedef struct _sVector
{
  uint8_t type; // 1: Cartesian coordinates; 2: polar coordinates
  union Component
  {
    struct _sCartesian
    {
      double x;
      double y;
    } c;
    struct _sPolar
    {
      double distance;
      double angle;
    } p;
  } data;
}sVector;
```

```
18
19 // Memory allocation. Fill zeros to all memory block.
20 sVector * myvector_init();
21
22 // If error, return -1; otherwise, return 0;
23 int myvector_set( sVector *pVector, uint8_t type, double a, double b );
24
25 // Print the given vector according to the given type.
26 // If Cartesian, print (x,y).
27 // If polar, print (distance, theta-pi). EX: distance=2 and angle=90 degree,
       print (2,0.5-pi).
28 // Theta must be between 0 and 2.
29 // If error, return -1; otherwise, return 0;
30 int myvector_print( const sVector *pVector, uint8_t type );
31
32 // A = B + C
33 // A's type is set to B's type.
34 // If error, return -1; otherwise, return 0;
35 int myvector_add( sVector *pA, const sVector *pB, const sVector *pC );
36
37 // A = B dot C
38 // If error, return -1; otherwise, return 0;
39 int myvector_inner_product( double *pA, const sVector *pB, const sVector *pC )
       ;
40
41 // Get the area of the parallelogram spanned by two vectors.
42 // If error, return -1; otherwise, return 0;
43 int myvector_area( double *pArea, const sVector *pB, const sVector *pC );
44
45 // Given a target point, find the closest point which can be combined from two
       given vectors with integer coefficients.
46 // (*pX,*pY) = m*A + n*B, m and n must be integers.
47 // (*pX,*pY) is the closet point to (*pTx,*pTy)
48 // If error, return -1; otherwise, return 0;
49 int myvector_cvp( double *pX, double *pY, const double *pTx, const double *pTy
       , const sVector *pA, const sVector *pB );
```

Figure 4 is an example of **myvector_area**. For **myvector_cvp**, given (Tx, Ty), find (x, y) which is closet to (Tx, Ty) and

$$(x, y) = m \cdot \overrightarrow{A} + n \cdot \overrightarrow{B},$$

where m, n must be integers.

You need to prepare **myvector.h** and TA will prepare **hw0205.c**. Of course, Makefile is your own business. Do not forget to make **hw0205.c** in your Makefile. If you use math.h, remember to use -lm in your Makefile. For your simplicity, I promise that m, n are 32-bits integers.

6 Bonus: Bit Operation (5 pts)

I want to write a program to display a 32-bit integer in the binary form. So I write the following code. However, this code has some problem.

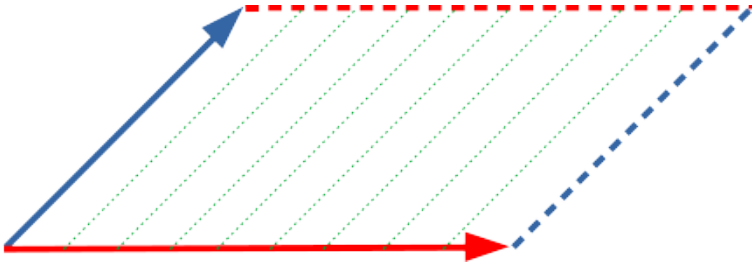

Figure 4: The area of the parallelogram spanned by two vectors.

```c
#include <stdio.h>
#include <stdint.h>

int main()
{
    int32_t number = 0;

    scanf( "%d", & number );

    int32_t bit = 1;
    bit = bit << 31;

    for( int i = 0 ; i < 32 ; i++ )
    {
        if( bit & number )
            printf( "1" );
        else
            printf( "0" );
        bit = bit >> 1;
    }
    return 0;
}
```

Please explain the reason of the problem of this code and show how to fix it.