C Programming I
2020 Fall
Homework 06

Instructor: Po-Wen Chi

Due: 2021.01.10 PM 11:59

Policies:

- Zero tolerance for late submission.
- 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.
- For the writing assignment, I only accept pdf. MS. doc/docx format is not acceptable. Moreover, please use Chinese instead of English.
- Do not forget your Makefile. For your convenience, each assignment needs only one Makefile.
- The executable programs should be hw0601, hw0602 ....

1 Integer Editor (20 pts)

Give a 64-bit integer, please develop a program to make the user edit the integer by giving a specific byte position and its new value. Note that you must use pointer in this problem.

```
$ ./hw0601
Please input an integer: 1
The integer: 1
(1) 0x01 (2) 0x00 (3) 0x00 (4) 0x00 (5) 0x00 (6) 0x00 (7) 0x00 (8) 0x00
Please enter the position (1-8, 0: End): 1
Please enter the new value (0-255): 255
The integer: 255
(1) 0xFF (2) 0x00 (3) 0x00 (4) 0x00 (5) 0x00 (6) 0x00 (7) 0x00 (8) 0x00
Please enter the position (1-8, 0: End): 0
```

From the example, you can see that this is **little-endian**.

2 Redo and Undo (20pts)

I believe that you all have used Redo and Undo functions in your text editor. Now, let's implement a program to simulate their behaviors. Suppose your editor can memorize 10 actions. Each action is represented by a positive integer. The simulator is an infinite loop and works as follows:

- When a user inputs a positive integer, add this integer to your action buffer and it will be the newest action. That is, there will no action that you can redo. If the buffer is full, kick off the oldest one.
- When a user inputs -1, it means the user wants to undo an action. Note that the action that is undone is still in your memory buffer since you may redo the action again.
- When a user inputs -2, it means the user wants to redo an action. There will be no new coming action. If nothing can be redone, just skip this command.
- When a user input 0, terminate the program and dump all actions in the buffer according to their coming order. Note that the output action number may be less than 10. 0, -1, -2 are not actions and should not be printed.

```
$ ./hw0602
input: 3
input: 2
input: 1
input: 4
input: 5
input: 0
Result: 3 2 1 4 5
```

Other examples:

1. input: 3 2 1 4 5 6 7 100 97 110 32 0,
   output: 2 1 4 5 6 7 100 97 110 32
2. input: 3 2 1 -1 0,
   output: 3 2
3. input: 1 2 3 4 5 -1 -1 7 0,
   output: 1 2 3 7
4. input: 1 2 3 4 5 -1 -2 7 0,
   output: 1 2 3 4 5 7

You need to implement the function in another C code and prepare a header file.

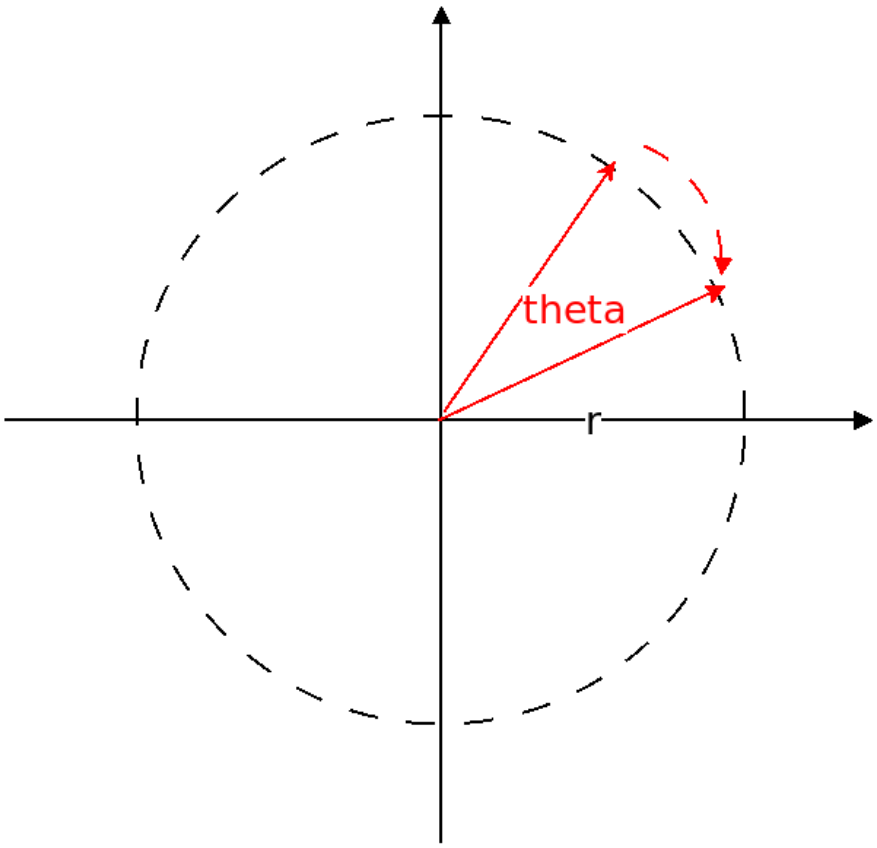

Figure 1: Rotation.

3 Rotation (20 pts)

Given a 2-d coordinate and θ, please rotate this point with θ clockwise where the circle center is O, as shown in figure 1.

Note that you need to implement the following function. The coordinate after rotation should be returned in the input arguments.

```
void rotate( double *x, double *y, double theta );
```

The behavior of your program should be as follows.

```
$ ./hw0603
Please enter a point: 0 1
Please enter theta (0-360): 90
1 0
```

Again, precision is not the concern in this class.

You need to implement the function in another C code and prepare a header file.

4 Projection of a Point onto a Plane (20 pts)

In mathematics, a plane is a flat, two-dimensional surface that extends infinitely far. Generally, we use the following linear equation as a normal form if a plane:

$$ax + by + cz + d = 0,$$

where x, y, z are used to describe a position in a three-dimensions.

Now give you a plane and a point, please develop a function to project the point to the plane and get the projection point.

```
void project( double *x, double *y, double *z, int32_t a, int32_t b, int32_t c
    , int32_t d );
```

The behavior of your program should be as follows.

```
$ ./hw0604
Please enter the plane: 0 0 2 0
The plane is 2z=0
Please enter the point: 1.5 2.7 -3.2
The projection is (1.5,2.7,0)
```

Again, precision is not the concern in this class.

You need to implement the function in another C code and prepare a header file.

If you do not know how to do projection. maybe you can reference the following link or ask our smart TAs.

`https://stackoverflow.com/questions/9605556/how-to-project-a-point-onto-a-plane-in-3d`

5 Poker Again (20 pts)

Let's play poker!! In some poker games, the player sorts his hand card by the card number instead of the card suit. The card encoding is as follows:

- 1-13: ♠ Ace to King.
- 14-26: ♡ Ace to King.
- 27-39: ♢ Ace to King.
- 40-52: ♣ Ace to King.

You should implement a shuffle function and output four hand-cards for four players. Then you should sort their cards in the ascending order according to the card value, not suit. That is, you must implement the following functions:

```
void print_card( const uint8_t player[13] );
int32_t sort_card( uint8_t player[13], int32_t ( * compare)( int32_t a,
    int32_t b ) );
```

There are three sorting ways defined as follows.

1. According to the card value in the descending order.
2. According to the card value in the descending order, but "2" is the biggest number.

3. According to the card suit, ♠ → ♡ → ♢ → ♣. Each suit should be sorted in the descending order according to the card value.

That is, you need to develop three different comparison functions. Note that these function should be placed in another file different with the main function. For your simplicity, the TA will prepare a header file called **test.h** and you need to include it in your source code. **test.h** will be like the following code:

```
uint8_t player1[13] = {...};
uint8_t player2[13] = {...};
uint8_t player3[13] = {...};
uint8_t player4[13] = {...};
```

You need to include it in your code. The main function should be as follows:

```
printf( "Before:\n" );
print_card( player1 );
print_card( player2 );
print_card( player3 );
print_card( player4 );
printf( "Type 01:\n" );
sort_card( player1, func01 );
sort_card( player2, func01 );
sort_card( player3, func01 );
sort_card( player4, func01 );
print_card( player1 );
print_card( player2 );
print_card( player3 );
print_card( player4 );
printf( "Type 02:\n" );
sort_card( player1, func02 );
sort_card( player2, func02 );
sort_card( player3, func02 );
sort_card( player4, func02 );
print_card( player1 );
print_card( player2 );
print_card( player3 );
print_card( player4 );
printf( "Type 03:\n" );
sort_card( player1, func03 );
sort_card( player2, func03 );
sort_card( player3, func03 );
sort_card( player4, func03 );
print_card( player1 );
print_card( player2 );
print_card( player3 );
print_card( player4 );
```

Of course, if TA gives you an unreasonable hand of cards, you need to print a warning message.

You need to implement the function in another C code and prepare a header file.

6 Bonus: What is the difference (10 pts)

What is the difference of the following two codes?

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

int main()
{
    int32_t a[9][9] = {0};

    for( size_t i = 0 ; i < 9 ; i++ )
    {
        for( size_t j = 0 ; j < 9 ; j++ )
        {
            a[i][j] = ( i + 1 ) * ( j + 1 );
        }
    }

    for( size_t i = 0 ; i < 9 ; i++ )
    {
        for( size_t j = 0 ; j < 9 ; j++ )
        {
            printf( "%02d ", a[i][j] );
        }

        printf( "\n" );
    }

    return 0;
}
```

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

int main()
{
    int32_t * a[9] = {0};

    for( size_t i = 0 ; i < 9 ; i++ )
    {
        a[i] = malloc( 9 * sizeof( int32_t ) );

        for( size_t j = 0 ; j < 9 ; j++ )
        {
            a[i][j] = ( i + 1 ) * ( j + 1 );
        }
    }

    for( size_t i = 0 ; i < 9 ; i++ )
    {
        for( size_t j = 0 ; j < 9 ; j++ )
```

```
        {
            printf( "%02d ", a[i][j] );
        }

        printf( "\n" );
    }

    return 0;
}
```

Please write down your answer and do some experiments to show your description. Again, you need to write down your answer in **Chinese** except some foreign students.

Merry Christmas