National Taiwan Normal University
CSIE Computer Programming I

Instructor: Po-Wen Chi
Due Date: 2023.12.19 PM 11:59

Assignment 5

Policies:

- Zero tolerance for late submission.
- 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** or **TEXT**. MS Word is not allowed.
- Do not forget your Makefile. For convenience, each assignment needs only one Makefile.
- Please provide a README file. The README file should have at least the following information:
  - Your student ID and your name.
  - How to build your code.
  - How to execute your built programs.
  - Anything that you want to notify our TAs.
- **DO NOT BE A COPYCAT!!** You will get ZERO if you are caught.

5.1 Statistics (20 pts)

Please develop a program for simple data analysis. Given a series of **int32_t** data, please derive **mean**, **variance**, and **standard deviation**.

```
int32_t statistics( int32_t *pData, int32_t size, double *pMean
    , double *pVariance, double *pStd );
```

You need to implement the following functions with a header file called **mystatistics.h**. Our TAs will prepare hw0501.c which includes mystatistics.h and uses these functions. **Do not forget to make hw0501.c to hw0501 in your Makefile.**

5.2 Gaussian Elimination (20 pts)

Given a matrix $\mathbf{A}$ and a vector $\mathbf{y}$, please find a vector $\mathbf{x}$ so that $\mathbf{Ax} = \mathbf{y}$. If it is unfamiliar to you, do not worry. I will give you an example as follows. Given

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} 6 \\ 17 \\ 34 \end{bmatrix},$$

You can get $\mathbf{x}$ is

$$\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \text{ because } \mathbf{Ax} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ 17 \\ 34 \end{bmatrix}.$$

Actually, this problem is equal to solve the following problem.

$$\begin{cases} x_1 + x_2 + x_3 = 6 \\ x_1 + 2x_2 + 4x_3 = 17 \\ x_1 + 3x_2 + 9x_3 = 34 \end{cases}$$

This time, I want you to develop a function to solve this problem.

```c
// I promise that A is an nxn matrix (2d array).
// However, you need to check if the inputs are valid.
// You should malloc for x.
// Return -1 if the inputs are invalid.
// Return 0 if there is no solution.
// Return 1 if there is only one solution.
// Return 2 if there are more than one solutions.
int32_t gaussian_elimination( int32_t n, int32_t *pA, int32_t *
    py, int32_t **px );
```

You need to implement the following functions with a header file called **myge.h**. Our TAs will prepare hw0502.c which includes myge.h and uses these functions. **Do not forget to make hw0502.c to hw0502 in your Makefile.** For your simplicity, I promise that all answers are 32-bits integers.

5.3 Sphere (20 pts)

A sphere is a geometrical object that is a three-dimensional analogue to a two-dimensional circle. Formally, a sphere is the set of points that are all at the same distance r from a given point in three-dimensional space. A spherical cap is a portion of a sphere cut off by a plane. Fig. 5.1 is an example. Now given a sphere where its center is at $(0, 0, 0)$ and a plane $ax + by + cz = d$, please develop a function to derive the area of the base of the cap.

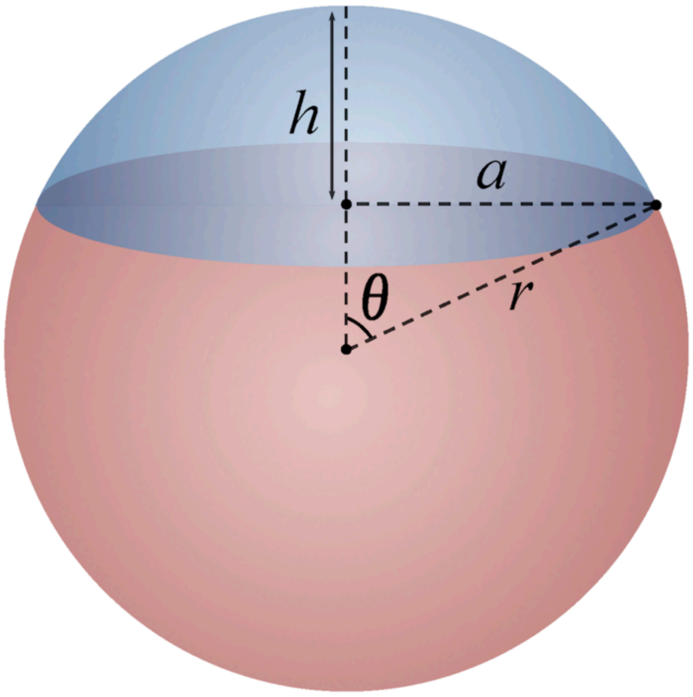

FIGURE 5.1: A spherical cap example.

```c
// Return -1 if the inputs are invalid.
// Return 0 if no such a plane.
// Otherwise, Return 1.
int32_t get_cap_area( double r, double a, double b, double c,
    double d, double *pArea );
```

You need to implement the following functions with a header file called **mysphere.h**. Our TAs will prepare hw0503.c which includes mysphere.h and uses these functions. **Do not forget to make hw0503.c to hw0503 in your Makefile.** If the plane is a tangent plane, the area will be set to zero.

5.4 TLV (20 pts)

TLV (type-length-value or tag-length-value) is an encoding scheme used for informational elements. TLV is an element unit. The type and length are fixed in size, and the value field is of variable size. The decoder can process value according to the type and the length. This structure make it possible to assemble many TLVs in one stream.

In this problem, type is one byte[1] and length is two bytes[2]. All types

[1]uint8_t.
[2]uint16_t.

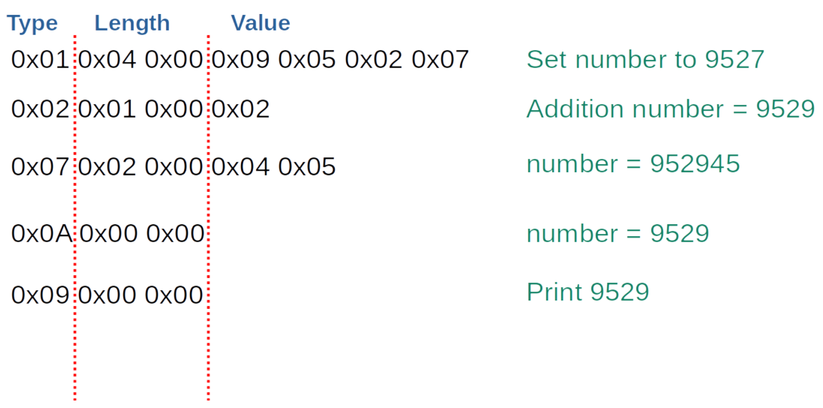

FIGURE 5.2: Step-by-Step processing.

are listed as Table 5.1.

Let's see an example. Given a byte array as follows.

```
uint8_t array[] = {0x01,0x04,0x00,0x09,0x05,0x02,0x07,0x02,
                   0x01,0x00,0x02,0x07,0x02,0x00,0x04,0x05,
                   0x0A,0x00,0x00,0x09,0x00,0x00 };
```

You should print 9529 on your screen. The step-by-step interpretation is shown in Fig. 5.2.

Please develop a function that given a series of uint8_t bytes, please print its output.

```
// Return -1 if the inputs are invalid.
// Otherwise, Return 0.
int32_t run( uint8_t *pByteArray, int32_t size );
```

You need to implement the following functions with a header file called **mytlv.h**. Our TAs will prepare hw0504.c which includes mytlv.h and uses these functions. **Do not forget to make hw0504.c to hw0504 in your Makefile.**

5.5 TAS Editor (20 pts)

TAS (Tool-assisted speedrun)link generally defined as a speedrun or playthrough composed of precise inputs recorded with tools such as video game emulators. e.g. Super Mario Bros and Pokemon. It is commonly employed to achieve records in various categories like any%link, 100%link, glitch end runlink and arbitrary code executionlink. TAS can do the console actions that would be impossible for a human to perform in reality.

Your TA was very love to play DS and Wii which are Nintendo video game consoles in his childhood. One of his favorite series game is Super Mario BrosFig5.3. He repeatedly completing the game 100%, even attempting speedrun on it. Of course, he sometimes watches something cool video

Table 5.1: Type Table

Type	Description
1	number = value. The length implies the value's decimal digits. EX: len=3 and data={0x01,0x02,0x03}, the value is 123. For your simplicity, I promise that all bytes for this type is from 0 to 9.
2	number += value. The value decoding is the same with type 1.
3	number *= value. The value decoding is the same with type 1.
4	number = number / 2. Note the length should be set to zero for this type. Only keep the quotient.
5	number = number / 10. Note the length should be set to zero for this type. Only keep the quotient.
6	number = value || number. The value decoding is the same with type 1. Ex: number = 123 and value = 456, the number will be 456123.
7	number = number || value. The value decoding is the same with type 1. Ex: number = 123 and value = 456, the number will be 123456.
8	Initial the number to zero. Note the length should be set to zero for this type. Only keep the quotient.
9	Print the current number. Note the length should be set to zero for this type. Only keep the quotient.
10	Invalidate the previous TLV. You should process the cancel TLV from the last to the beginning. For Example, given Cancel-Cancel-Cancel, the last Cancel will invalidate its previous Cancel and only the first Cancel works. Note the length should be set to zero for this type. Only keep the quotient.
Others	Just skip the value.

about TAS speedrun. And he is also curious about the principles behind it. He finds out that TAS control based on reading the contents inside files.

FIGURE 5.3: Super Mario Bros game

So this time, I want you to implement some functions about TAS edit action. For your convenient, it is not necessary to test TAS on DS or Wii platform which are require high performance. And your program is required to generate a file named with the extension fm2link for testing on the NESlink platform, with the classic game being Super Mario Bros.

There's the description of these function:

```
void button_set_frame(uint8_t **src, size_t *size, const
    uint8_t button, const uint64_t start_frame, const uint64_t
    end_frame);
/*
     Parameters:
         src: Button value array pointer for generate fm2 file.
         size: src size for pointer.
         button: Button uint8_t value to set, not replace.
         start_frame: Represent start frame to set button.
         end_frame: Represent end frame to set button.

     Note:
         If the frame range is overflow, please increase the
    size of src automatically

     Examples:
         Set 80 value (L, U button) from frame 3 to frame 480.

         $ button_set_frame(src, size, 80, 3, 480);
```

```

        Button set list: R, L, D, U, T, S, B, A.
        That is uint8_t range value convert to. For example:

              RLDUTSBA
        80 -> 01010000 -> set L, U

        Orginal src:
            {0, 8, 130, 0, 130, 130, 255}
            size of array: 7

        After passing the function of src will be:
            {0, 8, 130, 80, 210, 210, 255, 80 ..... 80}
            size of array: 481
*/

void button_unset_frame(uint8_t *src, const size_t size, const
    uint8_t button, const uint64_t start_frame, const uint64_t
    end_frame);
/*
    Parameters:
        src: Button value array for generate fm2 file.
        size: src size for pointer.
        button: Button uint8_t value to unset.
        start_frame: Represent start frame to unset button.
        end_frame: Represent end frame to unset button.

    Examples:
        Unset 2 value (B button) from frame 1 to frame 100.

        $ button_unset_frame(src, size, 2, 1, 100);

             RLDUTSBA
        2 -> 00000010 -> unset B

        Orginal src:
            {0, 8, 130, 0, 130, 130, 255}
            size of array: 7

        After passing the function of src will be:
            {0, 8, 128, 0, 128, 128, 253}
            size of array: 7
*/
```

TAS achieves high-precision control by manipulating button events frame by frame in a game. To let a NES emulator to do the TAS, the input is provided in the form of an fm2 text file. Each line in the fm2 file represents a button set, as the TAS input for each frame. There's fm2 file example here:

```
|0|........|||  # Frame 0:  Nothing, value: 0
|0|....T...|||  # Frame 1:  Button T event, value: 8
```

```
|0|........|||  # Frame 2:  Nothing, value: 0
|0|........|||  # Frame 3:  Nothing, value: 0
|0|R.....B.|||  # Frame 4:  Button R, B event, value: 130
|0|R.....B.|||  # Frame 5:  Button R, B event, value: 130
|0|........|||  # Frame 6:  Nothing, value: 0
|0|........|||  # Frame 7:  Nothing, value: 0
|0|R......A|||  # Frame 8:  Button R, A event, value: 129
|0|R.....B.|||  # Frame 9:  Button R, B event, value: 130
|0|..D.....|||  # Frame 10: Button D event, value: 32
|0|..D.....|||  # Frame 11: Button D event, value: 32
|0|R.....BA|||  # Frame 12: Button R, B, A event, value: 131
|0|RLDUTSBA|||  # Frame 13: All buttons event, value: 255
```

The src array to generate fm2 file above is:

```
src: {0, 8, 0, 0, 130, 130, 0, 0, 129, 130, 32, 32, 131, 255}
```

To generate fm2 file, I'll give you decode.c and decode.h to extract your fm2 file. Please link decode file in Makefile. The prototype of function is:

```
void extract_fm2_file(const uint8_t *src, const size_t size);
```

I'll use a NES emulator called FCEUXlink while running your fm2 file. To install it, you can run the following command:

```
$ sudo apt install fceux #Ubuntu, Debian
$ sudo brew install fceux #macOS

# Windows version following this URL:
# https://fceux.com/web/download.html

# Note: Maybe other Linux distribution also has repository that
    able to install FCEUX. You can try to find out yourself.
```

Emulator require an NES ROM to start. You can find your ROM here. And this is how to run it:

```
$ fceux --playmov <your fm2 file> <your NES game ROM>
```

There are some FCEUX shortcuts:

- NES GamepadFig5.4 (keyboard map to gamepad button):
  - D: B
  - F: A
  - Enter: T (Start)
  - S : S (Select)
  - Keypad up: U (Up)
  - Keypad left: L (Left)
  - Keypad down: D (Down)
  - Keypad right: R (Right)

- Show Frame Count: .
- Frame Speed Up: +
- Frame Speed Down: -
- Enter or Quit Fullscreen:
  - Alt + Enter (For Ubuntu or Windows)
  - Option + Enter (For macOS)

FIGURE 5.4: NES Gamepad

You should also prepare a header file called tas.h. TA will prepare hw0505.c which includes tas.h and uses these functions. **Do not forget to make hw0505.c and decode.c to hw0505 in your Makefile.**

5.6 Bonus: What is Wrong? (5 pts)

There is a programming problem: Please write a program to ask the user to input 5 16-bits integers and print these integers in the reverse order. Alice and Bob write their own codes and the execution results are as follows.

```c
// Alice Code
#include <stdio.h>
#include <stdint.h>

int main()
{
    int16_t array[10] = {0};

    for( int32_t i = 0; i < 5 ; i++ )
    {
        scanf( "%d", &array[i] );
    }

    for( int32_t i = 5; i > 0; i-- )
    {
        printf( "%d ", array[i-1] );
    }

    printf( "\n" );

    return 0;
}
```

```
$ ./a.out
1
2
3
4
5
5 4 3 2 1
```

```c
// Bob Codes
#include <stdio.h>
#include <stdint.h>

int main()
{
    int16_t array[10] = {0};

    for( int32_t i = 5; i > 0 ; i-- )
    {
        scanf( "%d", &array[i-1] );
    }

    for( int32_t i = 0; i < 5; i++ )
    {
        printf( "%d ", array[i] );
    }

    printf( "\n" );

    return 0;
}
```

```
$ ./a.out
1
2
3
4
5
5 0 0 0 0
```

What is wrong with Bob's code? Please explain the result. Note that I do not want you to correct their codes but I want you to describe the reason. Besides, why the array size is ten? What will happen if the size is five?