National Taiwan Normal University
CSIE Computer Programming I

Instructor: Po-Wen Chi
Due Date: 2022.12.06 PM 11:59

Assignment 4

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**. MS Word is not allowed.
- Do not forget your Makefile. For convenience, each assignment needs only one Makefile.
- Please provide a README.

4.1 Derivatives of Polynomials (20 pts)

Given a polynomial $f(x)$ and a number a, please implement the a function to get $f^{(n)}(a)$, where

$$f^{(n)}(x) = \frac{d}{dx} f^{(n-1)}(x)$$

```
// Input Arguments:
// n: n-th derivatives
// a: the input value
// size: the size (item numbers) of coefficients[] and powers[]
// coefficients[], powers[]: polynomials
double cal_n_derivatives( int32_t n, double a, int32_t size,
    int32_t coefficients[], int32_t powers[] );
```

For example, given

```
int32_t coefficients[] = {12,34,56};
int32_t powers[] = {5,1,0};
```

it implies $12x^5 + 34x + 56$.

You should also prepare a header file called myfunc.h. Our TAs will prepare hw0401.c which includes myfunc.h and uses this function. Do not forget to make hw0401.c to hw0401 in your Makefile. You do not need to consider the case that the case that these two arrays do not match. All powers are greater than or equal to zero. However, the order of the power array is not guaranteed.

4.2 Minesweeper (20 pts)

Minesweeper is a logic puzzle video game genre generally played on personal computers and I think that I do not need to teach you how to play this game right? Figure 4.1 is an example of the game screen.

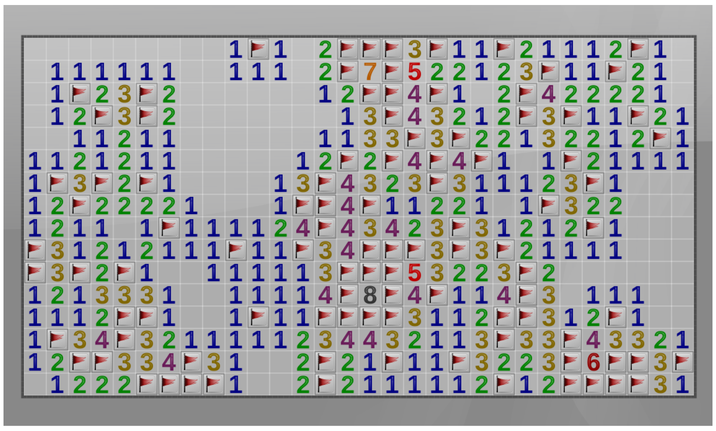


FIGURE 4.1: KMines, a free and open-source variant of Minesweeper.

A player selects a cell to open it. If a player opens a mined cell, the game ends. Otherwise, the opened cell displays either a number, indicating the number of mines diagonally and/or adjacent to it, or a blank tile (or ”0”), and all adjacent non-mined cells will automatically be opened. Now, given a game board, please find cells that can open most cells automatically. For your simplicity, the board size is static and is 16×30.

```
// board: game board. If the cell is a mine, the value is 1;
    otherwise, 0.
// row[480] and col[480]: Output answers (row[0],col[0]), (row
    [1],col[1]), ...
// Note that the row and col should start from the top left
    corner and start from 0.
// The output order should be ascending, row first than col.
// Except the wanted points, other values in these two arrays
    should be -1.

```

```c
int find_good_cells( const int32_t board[16][30], int32_t row
    [480], int32_t col[480] );
```

You should also prepare a header file called mine.h. Our TAs will prepare hw0402.c which includes myfunc.h and uses this function. Do not forget to make hw0402.c to hw0402 in your Makefile.

4.3 Rubik's Cube (20 pts)

You all know what it is, right? Or you can see figure 4.2 and I believe that you know how to play it.

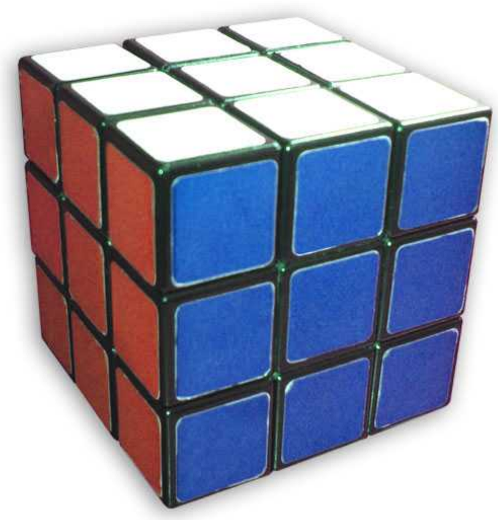


FIGURE 4.2: Rubik's Cube.

This time, I want you to develop a Rubik's Cube simulator. The color scheme is as figure 4.3. The game start from the red color, while the up side color is white. The user can input [0-6] to control the cube:

- 0: Exit the program.
- 1: Turn the first row from left to right.
- 2: Turn the second row from left to right.
- 3: Turn the third row from left to right.
- 4: Turn the first column from up to down.
- 5: Turn the second column from up to down.

- 6: Turn the third column from up to down.

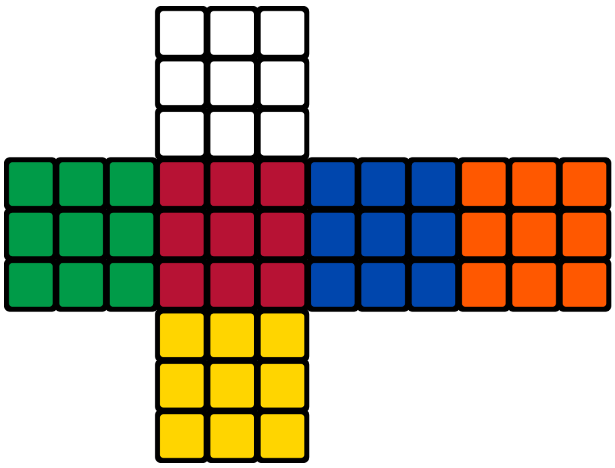


FIGURE 4.3: Rubik's Cube Color Scheme.

You can see figure 4.4 for example. Figure 4.5 is the program operation example. For any invalid input, you should make the user input again.

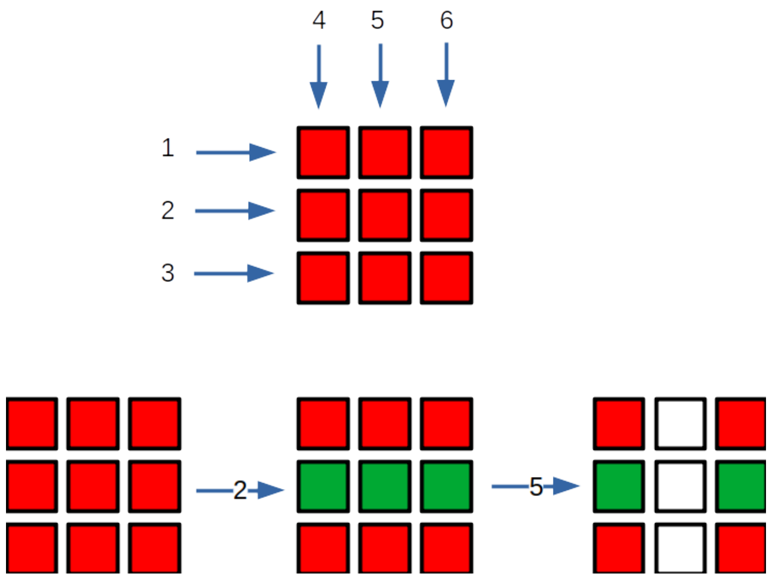
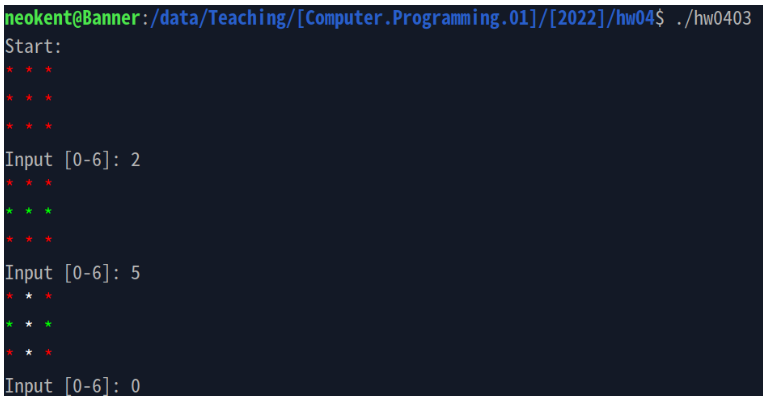


FIGURE 4.4: Rubik's Cube Movement Example.

4.4 Chinese Checkers (20 pts)

Chinese checkers is a strategy board game of German origin that can be played by two, three, four, or six people, playing individually or with part-



```
neokent@Banner:/data/Teaching/[Computer.Programming.01]/[2022]/hw04$ ./hw0403
Start:
* * *

* * *

* * *

Input [0-6]: 2
* * *

* * *

* * *

Input [0-6]: 5
* * *

* * *

* * *

Input [0-6]: 0
```

FIGURE 4.5: hw0403 Example.

ners. If you do not know what it is, you can see Figure 4.6 for reference and I believe that you know how to play this game[1].

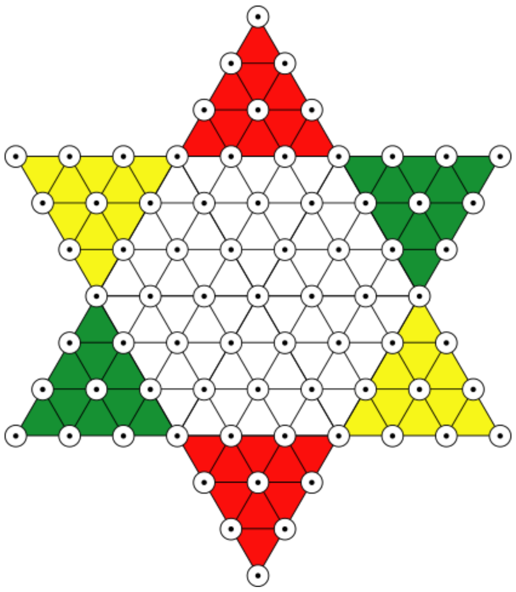
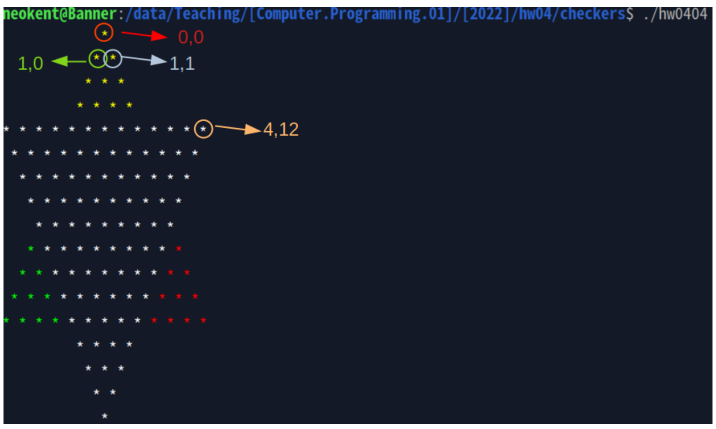


FIGURE 4.6: Chinese Checkers.

Now I want you to develop this game for three human players. The gameboard should be as Figure 4.7. The player order is **yellow** → **green** → **red**. The gameboard should be as Figure 4.7.

```
$ ./hw0404
(board ... omitted here)
Yellow Player Turn:
  From: 3,0
  To: 4,4
```

[1]You can google the rule on Wiki.



```
neokent@Banner:/data/Teaching/[Computer.Programming.01]/[2022]/hw04/checkers$ ./hw0404
            *
           * *
          * * *
         * * * *
* * * * * * * * * * * * *
 * * * * * * * * * * * *
  * * * * * * * * * * *
   * * * * * * * * * *
    * * * * * * * * *
   * * * * * * * * * *
  * * * * * * * * * * *
 * * * * * * * * * * * *
* * * * * * * * * * * * *
         * * * *
          * * *
           * *
            *
```

FIGURE 4.7: Chinese Checkers Game Board.

```
(board ... omitted here)
Green Player Turn:
  From:
...
```

The game ends when a player moves all pieces to the other side and you should print **"The Winner is Yellow"** for example. Note that if there is any invalid input, you should make the player restart its turn again.

4.5 A Game but not Pokemon (20 pts)

SubaRya, CSIE major at NTNU , likes to play video game with his schoolmates after accomplishing several wearing tasks from Computer Programming I. One day, a good idea crossed SubaRya's mind while he was playing Genshin Impact (原神), a well-known ARPG game, after doing Computer Programming I homework 1. He wanted to create a game by himself which is like Genshin Impact as well as the fouth question in homework 1, Pokemon.

At first, there is a 5×5 map. Adventurer is initially located at the top-left corner of the map, and Boss is initially located at the bottom-right corner of it, as shown in Figure 4.8. However, for your simplicity, Adventurer can only move either **down or right** at any point one step. And I promise that the size of map will exactly be 5×5.

Secondly, there is a device called Artifact (聖遺物) on every grid of the map which can be equipped on Adventurer to improve his or her ability. However, one Adventurer can only equip at most FIVE Artifacts. To put it differently, Adventurer can **choose the better Artifact and fall away one** if he or she has been already equipped five Artifacts. Adventurer can also **choose not to pick up the one** on the grid, though.

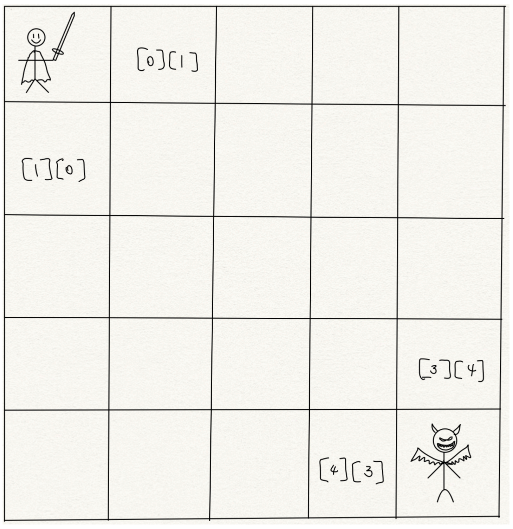


FIGURE 4.8: Game MAP.

Moreover, I promise that there are **exactly two arbitrary and different Attributes** on every Artifact. And then please see the Table 4.1. For example, if you get an Artifact which IDs are 2 and 5, the point of this Artifact you get will be...

$$100 + 50 = 150$$

Thirdly, there are **exactly seven types** of the Artifact in this world. Consequently, Adventurer can get some buff if he equip 2-Piece Set (**二件套**) or 4-Piece Set (**四件套**). Note that 2-Piece Set or 4-Piece Set means that Adventurer equips 2 or 4 Artifacts with the same type. Please see Table 4.2. For example, if you have five Artifacts, which {type, ID1, ID2} are {7,1,2},{7,1,5},{2,5,6},{3,7,2},{6,6,1} respectively, the point you get will be...

$$\{7, 1, 2\} = 125 + 100 = 225$$
$$\{7, 1, 5\} = 125 + 50 = 175$$
$$\{2, 5, 6\} = 50 + 25 = 75$$
$$\{3, 7, 2\} = 10 + 100 = 110$$
$$\{6, 6, 1\} = 25 + 125 = 150$$
$$2 - Piece\, Set\, of\, 7 = 100$$

$$total\, point = (point\, of\, every\, Artifact) + (point\, of\, 2 - Piece\, of\, Artifact)$$

Table 4.1: ID List.

ID of Attribute	Attribute	Point
1	Critical Damage (暴擊傷害)	125
2	Critical Rate (暴擊率)	100
3	Percent of ATK (攻擊力百分比)	175
4	ATK (攻擊力)	50
5	HP (生命值)	50
6	Percent of DEF (防禦力百分比)	25
7	DEF (防禦力)	10

Table 4.2: Additional Bonus for Set.

Type of Artifact	2-Piece Set Point	4-Piece Set Point
1	10	160
2	20	150
3	35	75
4	50	135
5	60	125
6	75	95
7	100	110

$$= (225 + 175 + 75 + 110 + 150) + (100)$$
$$= 835$$

The 835 point you get is called **"Total Point"**. Note that if you get the point of 4-Piece of Artifact, you **CANNOT** get the point of 2-Piece Set of Artifact with that type. For your simplicity, {type, ID1, ID2} = {0,0,0} is Adventurer, and {type, ID1, ID2} = {8,0,0} is Boss.

At last, Aventurer is so cautious that he or she will gather the information of the Boss before launching punitive expeditions against the Boss. Namely, before Adventurer head over to the point of the Boss (魔王的據點), he or she will figure out how many points do he or she required to defeat the Boss. You need to print the **top three "Total Points"** which Adventurer will get after he or she reach the point of the Boss by any road. After print the **top three "Total Points"**, you also need to print **their details of FIVE Artifacts (Type only)**. And then if **the third highest "Total Points"** you get is not smaller than the point required of the Boss, print "Ready Perfectly!" with green color. If **the third highest "Total Points"** you get is smaller than the point required of the Boss, print "Gonna be OK." with red color.

TA will prepare **basic.h** for you and you need to include it in your hw0405.c. You need to call TA' s function in your code. Of course, content of bossPoint and MapInfo will be different in TA's test file. basic.h will be as follows:

```
#ifndef BASIC_H
#define BASIC_H
#include<stdint.h>
static int32_t bossPoint = 1055;
static int32_t TypeOfAttribute[5][5]={
    {0,3,1,2,6},
    {1,5,1,4,7},
    {4,1,6,1,4},
    {7,6,1,7,2},
    {6,4,4,1,8}
};
static int32_t AttributeID1[5][5]={
    {0,1,1,1,1},
    {7,2,1,2,1},
    {5,1,1,3,5},
    {5,4,3,4,1},
    {1,1,5,4,0}
};
static int32_t AttributeID2[5][5]={
    {0,2,3,4,4},
    {6,1,6,1,3},
    {6,2,7,1,2},
    {3,6,5,1,4},
    {4,7,2,6,0}
};
#endif
```

Hint: If you have no idea how to get all Total Points and then select top three of them, you can try to enumerate all possible path.

```
$ ./hw0405
1st Top Total Point: 1075. Type of five Artifacts: 3 5 4 7 4.
2nd Top Total Point: 1075. Type of five Artifacts: 3 5 4 1 4.
3rd Top Total Point: 1060. Type of five Artifacts: 3 1 4 1 4.
Target Boss required point: 1055
Ready Perfectly!
```

4.6 Bonus: Code (5 pts)

Please explain the following code. Do not just describe what it will print on your screen, but explain why this happens.

```c
int p(int i, int N)
{
    return (i <= N && printf("%d ", i) && printf(" %d\n", N) &&
    !p(i + 1, N-1)) ;
}
```