Lab11

1. **Homework Problem I**
   a. Design a program to simulate 100 rolls of a six-sided die: Count the number of times each side of die appears and for each count use “*” to print out. The output format will be:

```
1.:****************
2.:*****************
3.:***************
4.:***************
5.:******************
6.:*******************
```

   b. We have some score data, the “-1” represents the end of the input, not the data.

   - Calculate the average score of the data.
   - Count the number of data in each interval. The difference between each interval is 10 point. That is 0~9, 10~19, 20~29, 30~39, 40~49,50~59, 60~69,70~79, 80~89, 90~100
   - Count the number of data above the average and below the average.
   - The output format will be:

```
*****SCORE REPORT*****

   MEAN = ##.###
   ABOVE MEAN = ##
   BELOW MEAN = ##

   0~    9:    *
   10~  19:    *
   20~  29:    ***
   30~  39:    ****
   40~  49:    **
   50~  59:    ****
   60~  69:    ******
   70~  79:    ****
   80~  89:    ****
   90~ 100:    *****
```

2. **Homework Problem II**

   a. Design a program to print out the result of matrix multiplication. Multiply matrix A and B, and then save the result to C. The dimensions of input matrixes should be inputted by the user and the array size of the matrix and its dimension should be the same. Example:

$$\begin{bmatrix} 2 & 1 & 3 & 4 \\ 1 & 2 & 1 & 1 \\ 3 & 1 & 1 & 3 \end{bmatrix} \times \begin{bmatrix} 1 & 3 & 2 \\ 2 & 1 & 1 \\ 1 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 27 & 15 \\ 7 & 11 & 7 \\ 9 & 20 & 12 \end{bmatrix}$$

   b. An interesting method of encoding data is to load a message to be encoded into a two-dimensional array and then interchange rows and interchange columns a number of times. The resulting sequence of characters is the encoded message. In order to decode the message, the sequence of steps used in the encoding process is followed in reverse order. For example, consider the message I HAVE BUT ONE LIFE TO GIVE FOR MY COUNTRY. Let us load the message into a 6×7 array:

I		H	A	V	E	
B	U	T		O	N	E
	L	I	F	E		T
O		G	I	V	E	
F	O	R		M	Y	
C	O	U	N	T	R	Y

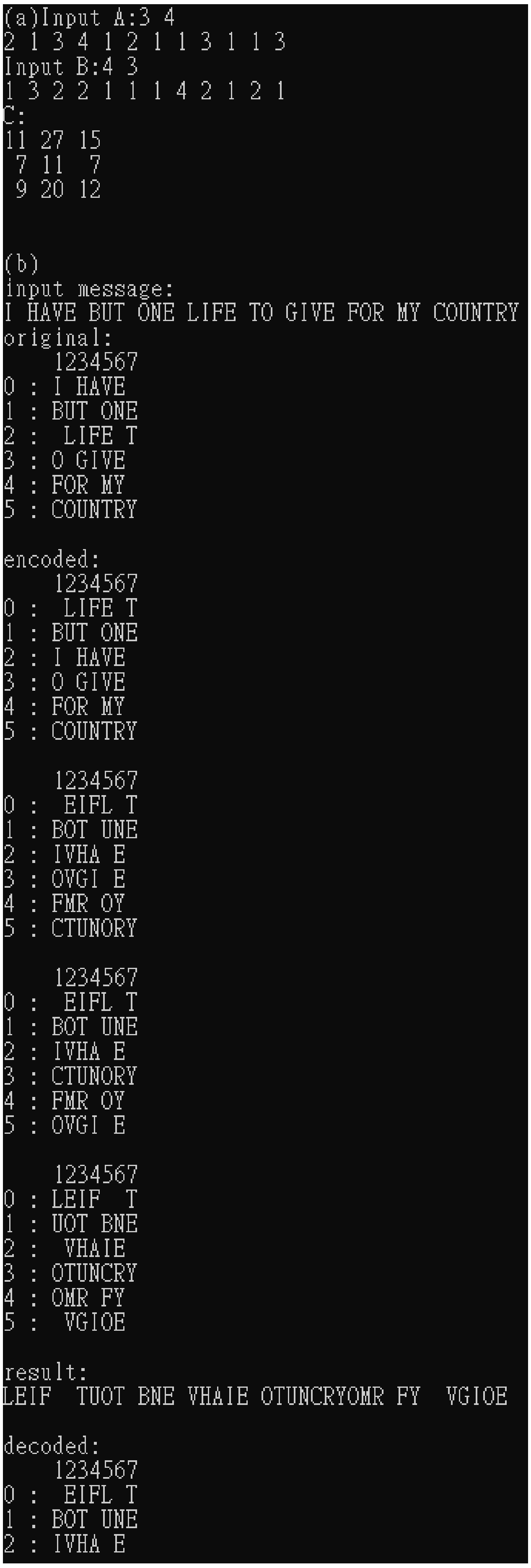
   Now consider the following encoding process:

   I. Interchange rows 1 and 3.
   II. Interchange columns 2 and 5.
   III. Interchange rows 4 and 6.
   IV. Interchange columns 1 and 5.

The resulting string is LEIF TUOT BNE VHAIE OTUNCRYOMR FY
VGIOE. In order to decode the message, the encoding process is reversed; i.e., the encoded message would be loaded into a 6×7 array. And then the following interchanges would be performed:

   I. Columns 1 and 5
   II. Rows 4 and 6
   III. Columns 2 and 5
   IV. Rows 1 and 3

Write a main program to let user input the message, encode and decode messages, using subroutines to perform the column interchange and row interchange operations.

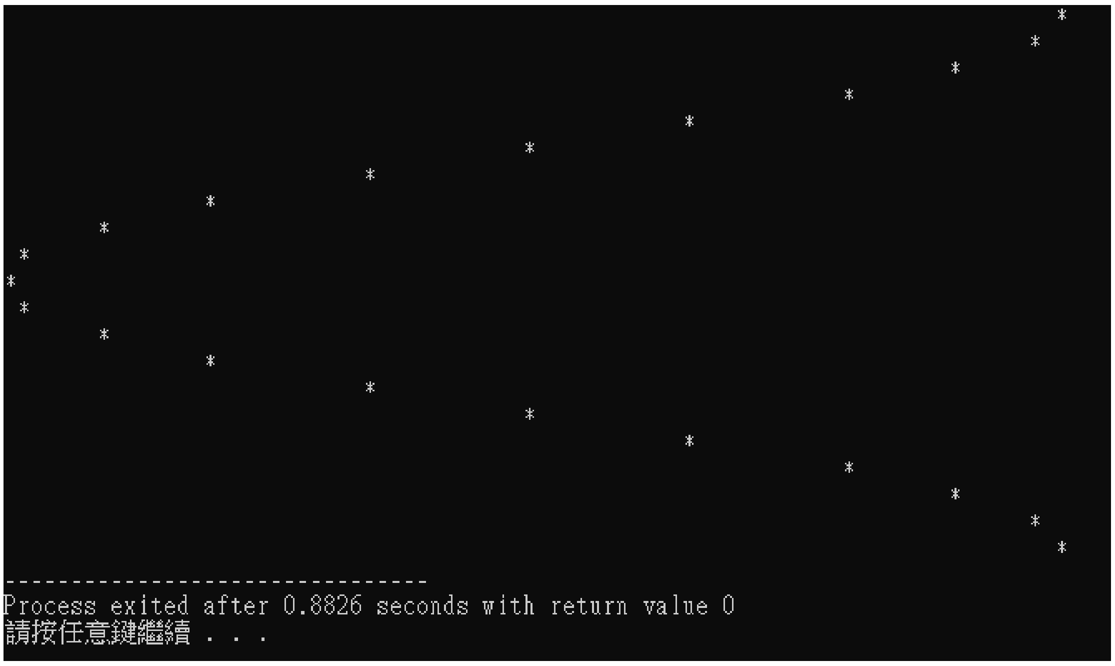


```
(a)Input A:3 4
2 1 3 4 1 2 1 1 3 1 1 3
Input B:4 3
1 3 2 2 1 1 1 4 2 1 2 1
C:
11 27 15
 7 11  7
 9 20 12


(b)
input message:
I HAVE BUT ONE LIFE TO GIVE FOR MY COUNTRY
original:
    1234567
0 : I HAVE
1 : BUT ONE
2 :  LIFE T
3 : O GIVE
4 : FOR MY
5 : COUNTRY

encoded:
    1234567
0 :  LIFE T
1 : BUT ONE
2 : I HAVE
3 : O GIVE
4 : FOR MY
5 : COUNTRY

    1234567
0 :  EIFL T
1 : BOT UNE
2 : IVHA E
3 : OVGI E
4 : FMR OY
5 : CTUNORY

    1234567
0 :  EIFL T
1 : BOT UNE
2 : IVHA E
3 : CTUNORY
4 : FMR OY
5 : OVGI E

    1234567
0 : LEIF  T
1 : UOT BNE
2 :  VHAIE
3 : OTUNCRY
4 : OMR FY
5 :  VGIOE

result:
LEIF  TUOT BNE VHAIE OTUNCRYOMR FY  VGIOE

decoded:
    1234567
0 :  EIFL T
1 : BOT UNE
2 : IVHA E
```

3. Plot a graph

Use the method introduced in homework 11 to plot the graph of

$$\mathbf{f(x) = \cos\left(\frac{x}{10}\pi\right), 0 \le x \le 20}$$



```
--------------------------------
Process exited after 0.8826 seconds with return value 0
請按任意鍵繼續 . . .
```

4. Ghost walk

Complete the **recursive** function **g_trace(…)** to make the ghost find the path to the player's position. The **g_trace(…)** will find out the path and record every step to the array named "path". After finding out the current path to the player, the ghost should follow the recorded steps to catch the player.

Since the player may move at any time, the ghost will update the path after a walk. That is, every time the ghost updates the path, it only takes the result of the next step. Please implement the function **g_move(…)** to make the ghost take a step.

The map and its size are defined in "maze.h", please include but copy & paste it.

Sample Code

```
#include <stdio.h>
#include <conio2.h>
#include <stdlib.h>
// something is missing here

// Complete the showMap definition
void showMap(int map…, int road_color, int wall_color);
void p_move(int *x, int *y, int road_color, int p_color);
```

```
// Complete the g_trace definition
void g_trace(int path…, int depth, int g_x, int g_y, int p_x,
int p_y, int *find);
void g_move(int next_dir, int *game, int* g_x, int* g_y, int
p_x, int p_y, int g_color, int road_color);
int dir[4][2] = {{0,-1},{0,1},{-1,0},{1,0}}; // up, down ,
left, right
int main(){

    int road_color = BLACK;
    int wall_color = WHITE;
    int ghost_color = YELLOW;
    int player_color = RED;

    int game = 2;
    int p_x=1, p_y=1;
    int path[1000], g_x=37, g_y=23;

    showMap(map, road_color, wall_color);
    gotoxy(p_x*2+1,p_y+1);
    textattr(RED);
    printf("█");

    while(game){
        p_move(&p_x, &p_y, road_color, player_color);

        int find = 0;
        g_trace(path, 0, g_x, g_y, p_x, p_y, &find);
        // Complete the call
        g_move(…, ghost_color, road_color);
        _sleep(100);

    }
    return 0;
}

// Complete the showMap definition
void showMap(int map…, int road_color, int wall_color) {
    for(int i=0; i<N ;i++){
        for(int j=0; j<M ;j++) {
            if(map[i][j]) {
                textattr(road_color);
                printf("█");
            } else {
                textattr(wall_color);
                printf("█");
            }
        }
        printf("\n");
    }
}

void p_move(int *x, int *y, int road_color, int p_color){
    char key;
    while(kbhit()){
        key = getch();

        int p_dir;
        switch(key) {
            case 'w':
                p_dir = 0;
                break;
            case 's':
                p_dir = 1;
                break;
            case 'a':
                p_dir = 2;
                break;
            case 'd':
                p_dir = 3;
                break;
            default:
                return;
        }

        gotoxy((*x)*2+1,*y+1);
        textattr(road_color);
        printf("■");

        int tx = (*x) + dir[p_dir][0];
        int ty = (*y) + dir[p_dir][1];
        if(map[ty][tx]){
            (*x) = tx;
            (*y) = ty;
        }

        gotoxy((*x)*2+1,*y+1);
        textattr(p_color);
        printf("■");
    }
}

// Complete the g_trace definition
void g_trace(int path…, int depth, int g_x, int g_y, int p_x,
int p_y, int* find){
    if(g_x==p_x && g_y==p_y){
        *find = 1;
        return;
    }
    map[g_y][g_x] = 0;
    int tx, ty;
    for(int i=0; i<4  && !(*find) ;i++){
        // Complete the loop
    }

    map[g_y][g_x] = 1;
}

void g_move(int next_dir, int *game, int* g_x, int* g_y, int
p_x, int p_y, int g_color, int road_color){

    // Complete the function
}
```