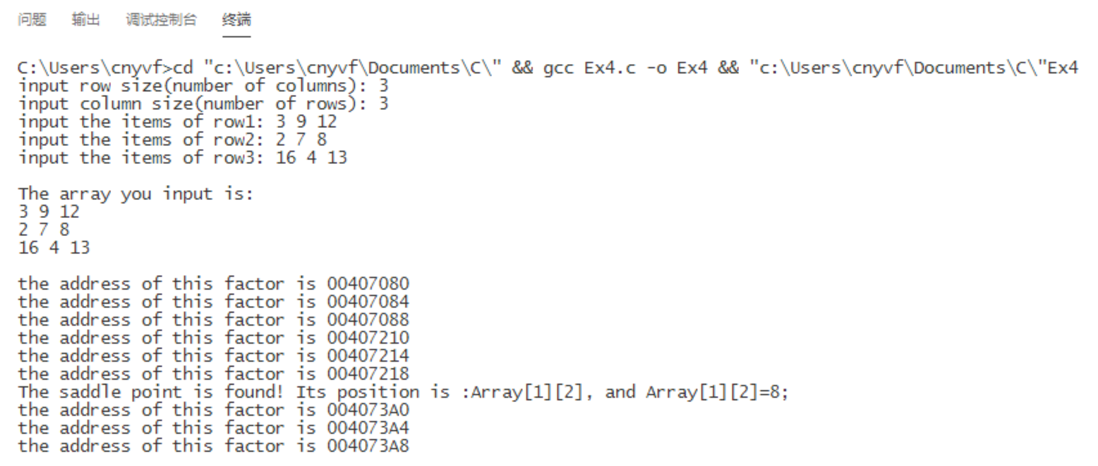
Exercise 1

Write a function to take the address of a two dimentional array, and find a saddle point for this array in this function. The saddle point is the maximum element in its row, and the minimum element in its column. The two dimentional array and its size should be input from the keyboard. Notice that the saddle point may not exist for an array.

If the saddle point does not exist, output the information that the saddle point does not exist. If there exists a saddle point, please output its row number, column number, and the element's value. **The input and all output results must be done in the main function.**

```c
#include <stdio.h>
int MassMent[100][100];
int maxR[100]; //用于储存每一行的最大值的列地址
int r,c;     //定义数组的长和宽
int temp=0; //用于临时存储数据
int text=0;
int pd;//判断是否为 saddle point

void findSaddlePoint(int i);

void main()
{
    printf("input row size(number of columns): ");
    scanf("%d",&r);
    printf("input column size(number of rows): ");
    scanf("%d",&c);
    for(int i=0;i<r;i++){   //输入二维数组的元素，并找出每一行最大值的地址
        printf("input the items of row%d: ",i+1);
        for(int j=0;j<c;j++){
            scanf("%d",&MassMent[i][j]);
            if(MassMent[i][j]>temp) {
                temp=MassMent[i][j];
                maxR[i]=j;
            }
        }
        temp=0;
    }
    printf("\nThe array you input is: \n");
    for(int i=0;i<r;i++){   //输出二维数组
        for(int j=0;j<c;j++){
            printf("%d ",MassMent[i][j]);
        }
        printf("\n");
    }
    printf("\n");
    for (int i = 0; i < r; i++) {//根据每一行的最大值的位置进行列遍历，找出鞍点
        findSaddlePoint(i);
        if(pd==1&&text==0){
            text=1;
        }
        if(pd==1){
            printf("The saddle point is found! Its position is :");
            printf("Array[%d][%d], and Array[%d][%d]=%d; \n",i,maxR[i],i,maxR[i
],MassMent[i][maxR[i]]);
        }
    }
    if(text==0){    //若无鞍点 输出错误提示
        printf("The saddle point is not found!");
    }
}

void findSaddlePoint(int i){
        pd=1;
        for(int j=0;j<r;j++){
            printf("the address of this factor is %p \n",&MassMent[i][j]); //输
出地址
            if(MassMent[i][maxR[i]]>MassMent[j][maxR[i]]){
                pd=0;
            }
        }
}
```

Case1:



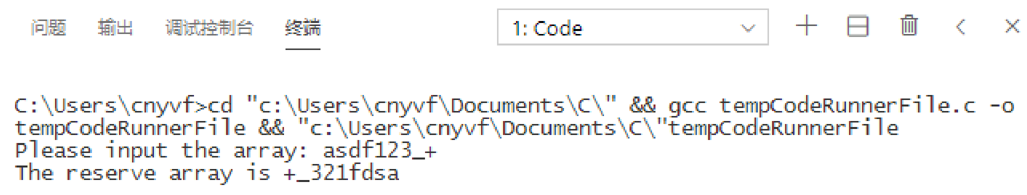
```
C:\Users\cnyvf>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4 && "c:\Users\cnyvf\Documents\C\"Ex4
input row size(number of columns): 3
input column size(number of rows): 3
input the items of row1: 3 9 12
input the items of row2: 2 7 8
input the items of row3: 16 4 13

The array you input is:
3 9 12
2 7 8
16 4 13

the address of this factor is 00407080
the address of this factor is 00407084
the address of this factor is 00407088
the address of this factor is 00407210
the address of this factor is 00407214
the address of this factor is 00407218
The saddle point is found! Its position is :Array[1][2], and Array[1][2]=8;
the address of this factor is 004073A0
the address of this factor is 004073A4
the address of this factor is 004073A8
```

Exercise 2

Write a function, to let a string to be stored reversely in an array, you can only use one array to achieve this purpose. The string must be input and output from the main function.

```c
#include <stdio.h>
char str[1000]; //数组
int count = 0;  //储存数组大小
char temp;  //临时储存字符

void saveArray();

void main()
{
    printf("Please input the array: ");
    while((temp=getchar())!='\n'){  //输入
        str[count]=temp;
        count++;
    }
    saveArray();
    printf("The reserve array is %s",str);  //输出
}

void saveArray(){   //反转数组
    for(int i=0;i<count/2;i++){
        temp=str[i];
        str[i]=str[count-1-i];
        str[count-1-i]=temp;
    }
}
```

Case1:



```
C:\Users\cnyvf>cd "c:\Users\cnyvf\Documents\C\" && gcc tempCodeRunnerFile.c -o
tempCodeRunnerFile && "c:\Users\cnyvf\Documents\C\"tempCodeRunnerFile
Please input the array: asdf123_+
The reserve array is +_321fdsa
```

Exercise 3

- Write a program which constructs a 5*5 matrix mat_c from another three matrixes mat_a, mat_b, and mat_c. The rule to build the matrix is as follows:

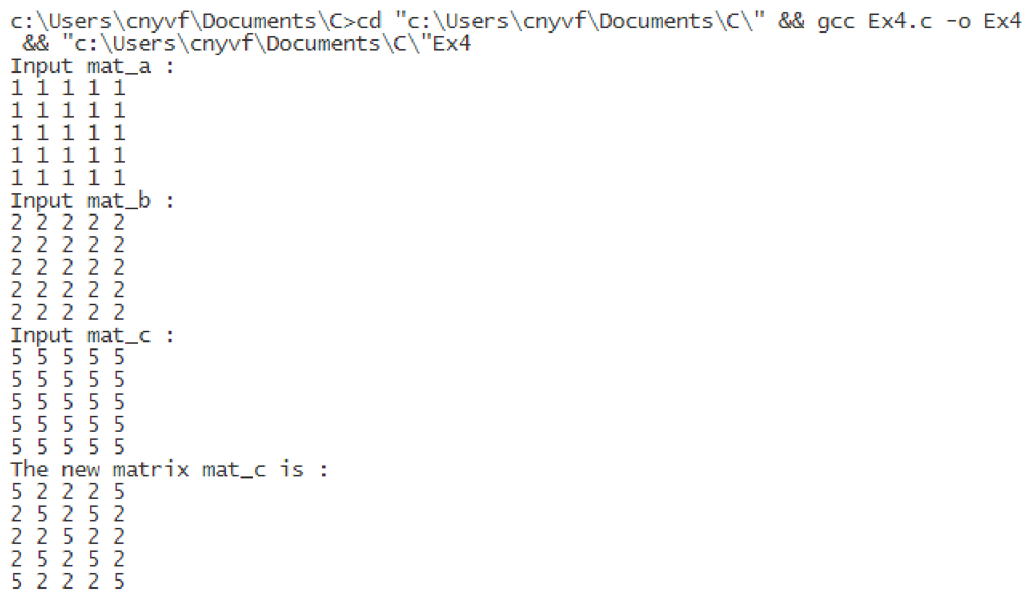
  The elements in diagonals of matrix mat_c are the biggest ones of the corresponding elements in mat_a, mat_b, and mat_c, and the rest elements in mat_c are the average of corresponding elements in mat_a, mat_b, and mat_c.

  Write a function construct_matrix() to construct such matrix mat_c, and write another function print_out() to output the result.

```
#include <stdio.h>
int mat_a[5][5],mat_b[5][5],mat_c[5][5]; //三个二维数组
int findMax(int a,int b,int c);
int findAvg(int a,int b,int c);
void construct_matrix();
void print_out();
void main()
{
    //分别输入三个矩阵
    printf("Input mat_a :\n");
    for(int i=0;i<5;i++){
        for (int j = 0; j < 5; j++) {
            scanf("%d",&mat_a[i][j]);
        }
    }
    printf("Input mat_b :\n");
    for(int i=0;i<5;i++){
        for (int j = 0; j < 5; j++) {
            scanf("%d",&mat_b[i][j]);
        }
    }
    printf("Input mat_c :\n");
    for(int i=0;i<5;i++){
        for (int j = 0; j < 5; j++) {
            scanf("%d",&mat_c[i][j]);
        }
    }
    construct_matrix();
    print_out();
}
int findMax(int a,int b,int c){ //返回对应位置的最大值
```

```
     if(a>=b&&a>=c){
         return a;
     }
     else if(b>=a&&b>=c){
         return b;
     }
     else{
         return c;
     }
}
int findAvg(int a,int b,int c){ //返回对应位置的平均值
    return ((a+b+c)/3);
}
void construct_matrix(){    //构建新矩阵
    for(int i = 0; i < 5; i++){
        for (int j = 0; j < 5; j++) {
            if(i==j||(i+j)==4){
                mat_c[i][j]=findMax(mat_a[i][j], mat_b[i][j], mat_c[i][j]);
            }
            else{
                mat_c[i][j] = findAvg(mat_a[i][j], mat_b[i][j], mat_c[i][j]);
            }
        }
    }
}
void print_out(){   //输出
    printf("The new matrix mat_c is : \n");
    for(int i = 0; i < 5; i++){
        for (int j = 0; j < 5; j++) {
            printf("%d ", mat_c[i][j]);
        }
        printf("\n");
    }
}
```

Case1:



```
c:\Users\cnyvf\Documents\C>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4
 && "c:\Users\cnyvf\Documents\C\"Ex4
Input mat_a :
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
Input mat_b :
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
Input mat_c :
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
The new matrix mat_c is :
5 2 2 2 5
2 5 2 5 2
2 2 5 2 2
2 5 2 5 2
5 2 2 2 5
```

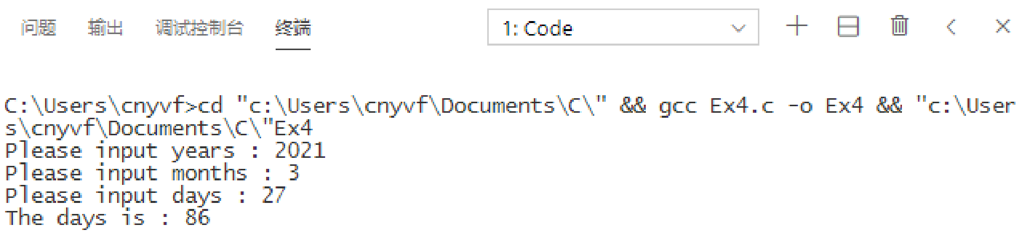
Exercise 4

Input a year, a month, and a day, then calculate how many days have passed in this year. You are required to write a function to calculate the days, and another function to judge if this year is a leap year.

```c
#include <stdio.h>
int years,months,days;  //分别表示年月日
int daysmuch[]={0,31,30,31,30,31,30,31,30,31,30,31,30};
int sumDays=0;

int judgeMent();
void dayMuch();

void main()
{
    //输入年月日
    printf("Please input years : ");
    scanf("%d",&years);
    printf("Please input months : ");
    scanf("%d",&months);
    printf("Please input days : ");
    scanf("%d",&days);
    dayMuch();
    printf("The days is : %d",sumDays);
}

int judgeMent(){    //判断是否为闰年
```

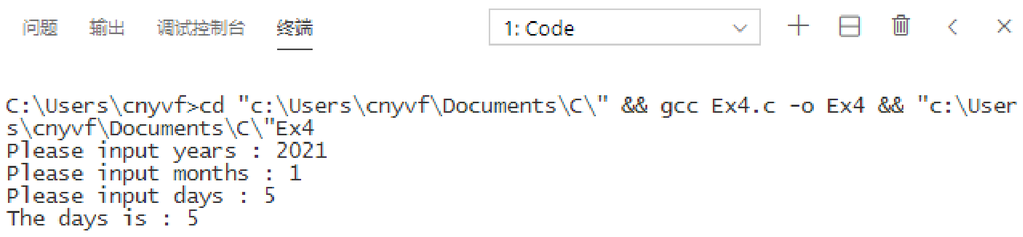
```
    if(years%4==0&&years%100!=0){
        return -1;
    }
    else if(years%400==0){
        return -1;
    }
    else{
        return -2;
    }
}

void dayMuch(){ //计算天数
    if(months==1){
        sumDays=days;
    }
    else if(months==2){
        sumDays=daysmuch[1]+days;
    }
    else{
        for(int i=1;i<=months-1;i++){
            sumDays+=daysmuch[i];
        }
        sumDays+=days;
        sumDays+=judgeMent();
    }
}
```

Case1: Non-Leap Year



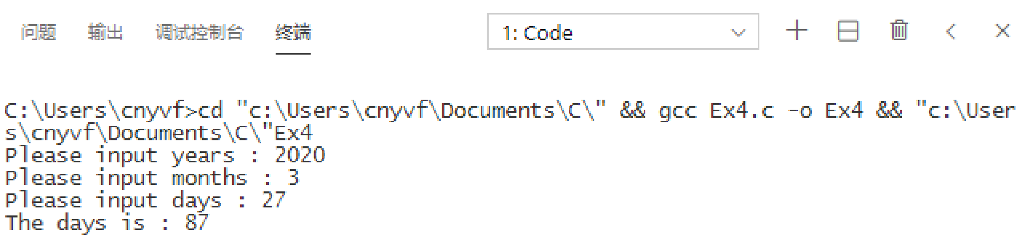
```
C:\Users\cnyvf>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4 && "c:\User
s\cnyvf\Documents\C\"Ex4
Please input years : 2021
Please input months : 3
Please input days : 27
The days is : 86
```

Case2: Non-Leap Year



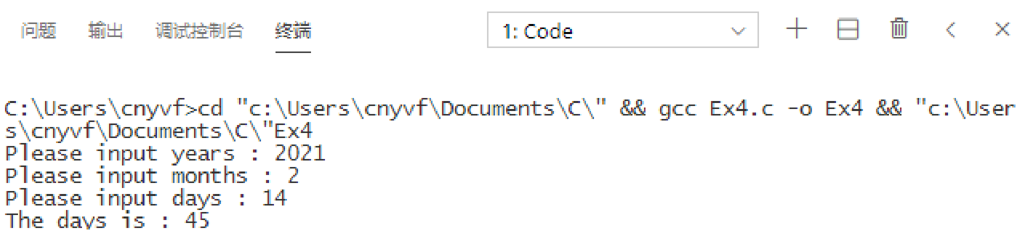
```
C:\Users\cnyvf>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4 && "c:\User
s\cnyvf\Documents\C\"Ex4
Please input years : 2021
Please input months : 1
Please input days : 5
The days is : 5
```

Case3: Leap Year



```
C:\Users\cnyvf>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4 && "c:\User
s\cnyvf\Documents\C\"Ex4
Please input years : 2020
Please input months : 3
Please input days : 27
The days is : 87
```

Case4: Non-Leap Year



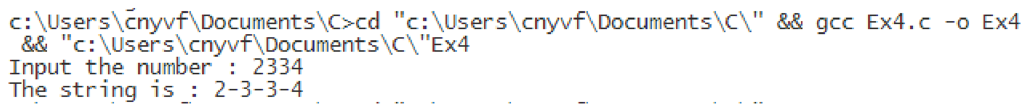
```
C:\Users\cnyvf>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4 && "c:\User
s\cnyvf\Documents\C\"Ex4
Please input years : 2021
Please input months : 2
Please input days : 14
The days is : 45
```

Exercise 5

Transform an integer *n* into a character string by using recursive calling, *n* could be any integer and it could be less than 0. For example, if you input 483, the output must be string character 4-8-3; if you input -2657, the output must be -2-6-5-7.

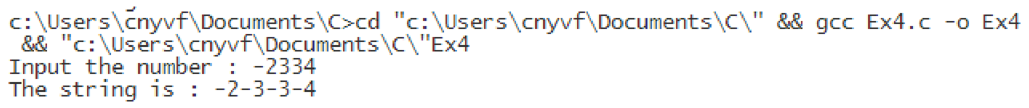
```
#include <stdio.h>

int tranString(int a);

void main()
{
   int nums;    //整数
   printf("Input the number : ");
   scanf("%d",&nums);
   printf("The string is : ");
   if(nums<0){
       printf("-");
       nums=-nums;
   }
   printf("%d",tranString(nums));   //输出
}
int tranString(int a){  //递归算法，转换整数为字符串
    int temp;
    temp=a%10;
    a/=10;
    if(a!=0){
        printf("%d-",tranString(a));
    }
    return temp;
}
```

Case1:



```
c:\Users\cnyvf\Documents\C>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4
 && "c:\Users\cnyvf\Documents\C\"Ex4
Input the number : 2334
The string is : 2-3-3-4
```

Case2:



```
c:\Users\cnyvf\Documents\C>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4
 && "c:\Users\cnyvf\Documents\C\"Ex4
Input the number : -2334
The string is : -2-3-3-4
```

Exercise 6

The Fibonacci numbers are defined recursively as follows:

$F_1=1$

$F_2=1$

$F_n=F_{n-1} + F_{n-2}, \quad n>2$

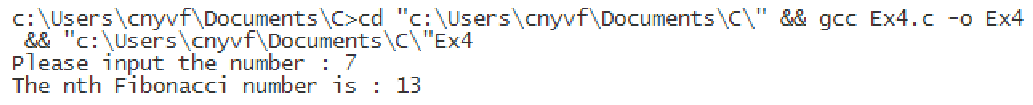
Therefore, the first few numbers in the sequence are 1,1,2,3,5,8,13.

Write a program to generate and print the *n*th Fibonacci number.

```
#include <stdio.h>

int f1=1,f2=1,f3; //三个临时变量，分别表示斐波那契数列的当前数和两个前序数
int n;  //指定第几个数

int fnumSpec(int a);
int fnum(int a);

void main()
{
   printf("Please input the number : ");
   scanf("%d",&n);  //输入
   printf("The nth Fibonacci number is : %d",fnumSpec(n));
}

int fnumSpec(int a){
    if(a<=2){   //特殊情况判断
        return 1;
    }
```

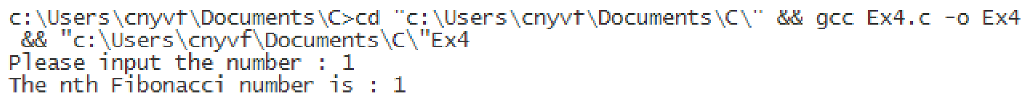
```c
    else{   //进入递归
        return fnum(a-2);
    }
}

int fnum(int a){    //递归求值
    if(a!=0){
        f3=f1+f2;
        f1=f2;
        f2=f3;
        return fnum(a-1);
    }
    else{
        return f3;
    }
}
```

Case1:



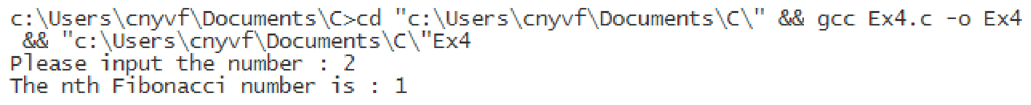
```
c:\Users\cnyvf\Documents\C>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4
 && "c:\Users\cnyvf\Documents\C\"Ex4
Please input the number : 7
The nth Fibonacci number is : 13
```

Case2:



```
c:\Users\cnyvf\Documents\C>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4
 && "c:\Users\cnyvf\Documents\C\"Ex4
Please input the number : 1
The nth Fibonacci number is : 1
```

Case3:



```
c:\Users\cnyvf\Documents\C>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4
 && "c:\Users\cnyvf\Documents\C\"Ex4
Please input the number : 2
The nth Fibonacci number is : 1
```

Exercise 7

- Write a function to calculate the summation of each digit's square of an integer *n*, the value of *n* could be any integer, and the output result must be as follows:

  Input an integer: 3698

  The output result is: 3*3+6*6+9*9+8*8=190

  Try to use two different methods to solve this problem:

  (1) Use the recursive calling and global variable to solve this problem.

  (2) Use the recursive calling and static local variable (do not use global variable) to solve this problem.

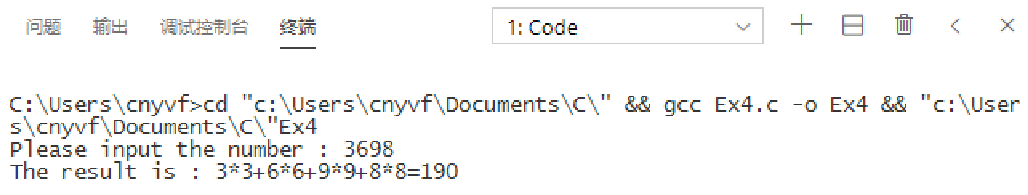
Exercise 7.1

```
#include <stdio.h>
int num;    //所需要的整数
int sum=0;  //求和
int addSum(int a);

void main()
{
    int test;   //临时值
    printf("Please input the number : ");
    scanf("%d",&num);
    printf("The result is : ");
    test=addSum(num);   //进入递归
    printf("%d*%d=%d",test,test,sum);   //尾部输出
}

int addSum(int a){  //递归
    int test;   //临时值
    int temp;
    temp=num%10;
    sum+=(temp*temp);
    num/=10;
    if(num!=0){
        test=addSum(a);
        printf("%d*%d+",test,test); //前置输出
        return temp;
    }
    else{
        return temp;
    }
}
```

Exercise 7.2

```c
#include <stdio.h>
int addSum(int a,int *b );

void main()
{
    int num;    //所需要的整数
    int sum=0;  //求和
    int test;   //临时值
    printf("Please input the number : ");
    scanf("%d",&num);
    printf("The result is : ");
    test=addSum(num,&sum);   //进入递归
    printf("%d*%d=%d",test,test,sum);   //尾部输出
}

int addSum(int a,int *b){  //递归
    static int test;   //临时值
    int temp;
    temp=a%10;
    *b+=(temp*temp);
    a/=10;
    if(a!=0){
        test=addSum(a,b);
        printf("%d*%d+",test,test); //前置输出
        return temp;
    }
    else{
        return temp;
    }
}
```

Case1:



```
C:\Users\cnyvf>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4 && "c:\User
s\cnyvf\Documents\C\"Ex4
Please input the number : 3698
The result is : 3*3+6*6+9*9+8*8=190
```

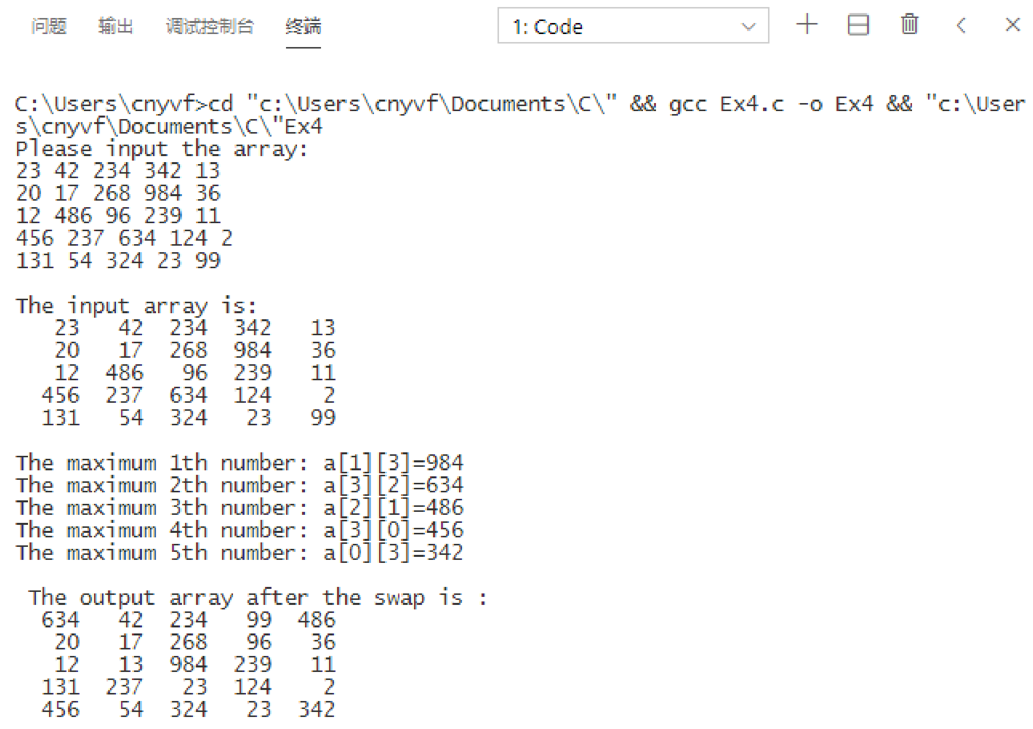
Exercise 8

There is a 5*5 matrix (two dimentional array), find the maximum five elements of the array, and exchange the location of the elements: let the maximum element be placed in the center of the matrix, and let the minimum 4 elements be placed in the 4 corners of the matrix. You must exchange the positions with the original elements. The placement sequence of the 4 elements is from left to right, from top to down, in decreasing order). Write a function to find the maximum elements and exchange their locations. Output this two dimentional array before and after the transformation from the main function. As the example illustrated in the next page.

```
#include <stdio.h>
#include <stdlib.h>
int massMent[5][5]; //二维数组
int maxNums[5]; //记录前五最大值
int saveI[5],saveJ[5];  //记录最大值对应的坐标
int smallpoint=0;   //记录最大值中的最小值
int m=5,n=5;    //横纵大小
int temp; //储存临时值

struct maxPoint{    //结构体用于绑定最大值及其坐标，便于排序
    int num;
    int ix;
    int jx;
}mp[5];

int cmp(const void*a,const void*b)  //比较函数,通过比较结构体中的 num 来逆序排序
{
    struct maxPoint xx=*(struct maxPoint*)a;
    struct maxPoint yy=*(struct maxPoint*)b;
    return yy.num-xx.num;
}

void findAndChange(){
    printf("\nThe input array is: \n");
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            printf("%5d",massMent[i][j]);
            if(massMent[i][j]>maxNums[smallpoint]){ //搜索二维数组并获取前五最大值
                maxNums[smallpoint]=massMent[i][j];
                saveI[smallpoint]=i;
                saveJ[smallpoint]=j;
            }
            for(int x=0;x<5;x++){   //刷新前五最大值中的最小值点
                if(maxNums[x]<maxNums[smallpoint]){
                    smallpoint=x;
                }
            }
        }
        printf("\n");
    }
    printf("\n");
    for(int i=0;i<5;i++){   //将已有的数值导入结构体
        mp[i].num=maxNums[i];
        mp[i].ix=saveI[i];
        mp[i].jx=saveJ[i];
    }
    qsort(mp,5,sizeof(mp[0]),cmp);  //结构体排序
    for(int i=0;i<5;i++){   //输出结果
        printf("The maximum %dth number: a[%d][%d]=%d \n",i+1,mp[i].ix,mp[i].jx
,mp[i].num);
    }

    //交换位置:

    temp=massMent[2][2];
    massMent[2][2]=mp[0].num;
    massMent[mp[0].ix][mp[0].jx]=temp;

    temp=massMent[0][0];
    massMent[0][0]=mp[1].num;
    massMent[mp[1].ix][mp[1].jx]=temp;

    temp=massMent[0][4];
    massMent[0][4]=mp[2].num;
    massMent[mp[2].ix][mp[2].jx]=temp;

    temp=massMent[4][0];
    massMent[4][0]=mp[3].num;
    massMent[mp[3].ix][mp[3].jx]=temp;

    temp=massMent[4][4];
    massMent[4][4]=mp[4].num;
    massMent[mp[4].ix][mp[4].jx]=temp;
```

```c
}

void main()
{
    printf("Please input the array: \n");
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            scanf("%d",&massMent[i][j]);    //输入二维数组
        }
    }
    findAndChange();
    printf("\n The output array after the swap is : \n");
    for (int i = 0; i < m; i++) {   //输出转换后的数组
        for (int j = 0; j < n; j++) {
            printf("%5d",massMent[i][j]);
        }
        printf("\n");
    }
}
```

Case1:



```
C:\Users\cnyvf>cd "c:\Users\cnyvf\Documents\C\" && gcc Ex4.c -o Ex4 && "c:\User
s\cnyvf\Documents\C\"Ex4
Please input the array:
23 42 234 342 13
20 17 268 984 36
12 486 96 239 11
456 237 634 124 2
131 54 324 23 99

The input array is:
   23   42  234  342   13
   20   17  268  984   36
   12  486   96  239   11
  456  237  634  124    2
  131   54  324   23   99

The maximum 1th number: a[1][3]=984
The maximum 2th number: a[3][2]=634
The maximum 3th number: a[2][1]=486
The maximum 4th number: a[3][0]=456
The maximum 5th number: a[0][3]=342

 The output array after the swap is :
  634   42  234   99  486
   20   17  268   96   36
   12   13  984  239   11
  131  237   23  124    2
  456   54  324   23  342
```