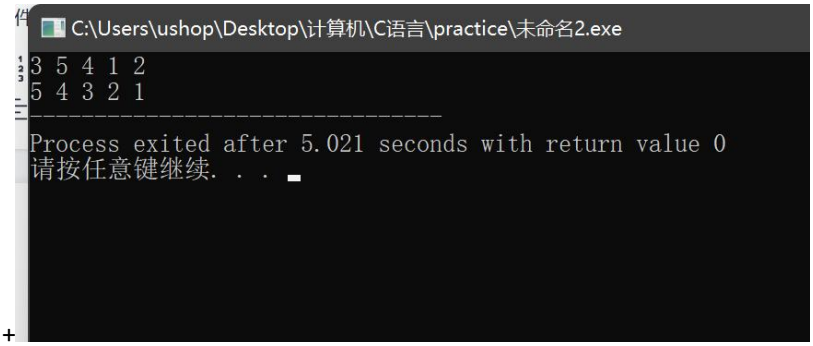
作业 1

一、数组与指针

1.

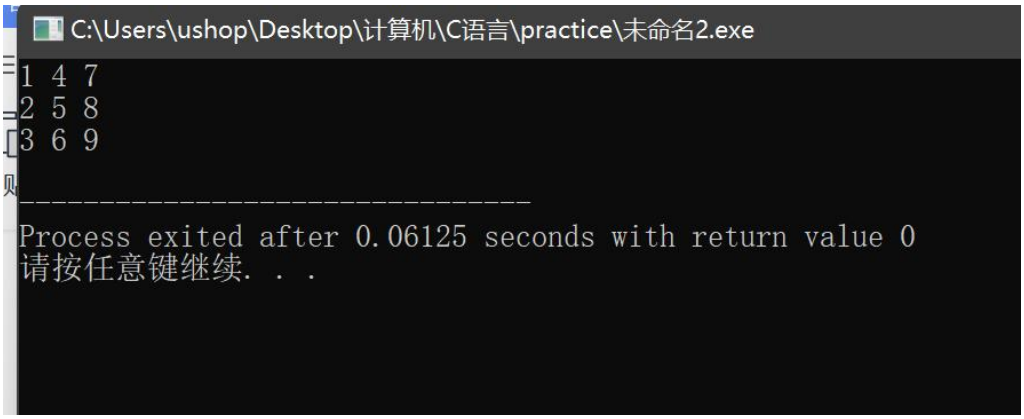
```
#include<stdio.h>

const int n = 5;

int main(){
	int a[4];
	for(int i = 0; i < n; ++ i){
		scanf("%d", &a[i]);
	}
	for(int i = 0; i < n - 1; ++ i){		//冒泡排序
		for(int j = 0; j < n - i - 1; ++ j){
			if(a[j] < a[j + 1]){
				int t = a[j];
				a[j] = a[j + 1];
				a[j + 1] = t;
			}
		}
	}
	for(int i = 0; i < n; ++ i) printf("%d ", a[i]);
	return 0;
}
```

验证：



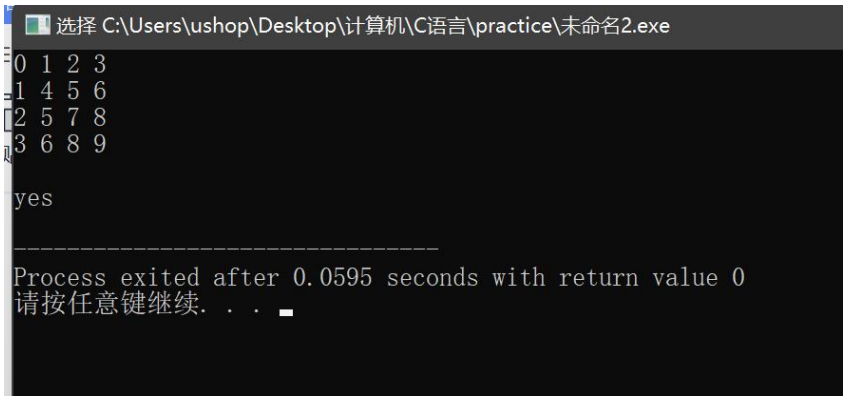
2.

```
#include<stdio.h>

const int n = 3;

int main()
{
    int a[3][3]={1,2,3,4,5,6,7,8,9};
    int b[3][3];

    for(int i = 0; i < n; ++ i)                    //另设一个 b 数组储存转置后的矩阵
       for(int j = 0; j < n; ++ j){
            b[j][i] = a[i][j];
       }

    for(int i = 0; i < n; ++ i){
       for(int j = 0; j < n; ++ j){
            a[i][j] = b[i][j];
            printf("%d ", a[i][j]);
      }
      printf("\n");
    }

    return 0;
}
```



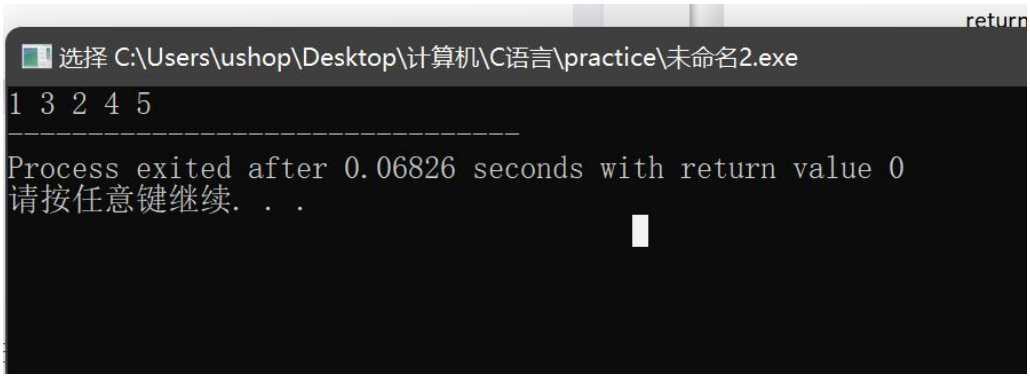
3.

```
#include<stdio.h>

const int n = 4;

int main()
{
  int a[ ][4]={0,1,2,3,1,4,5,6,2,5,7,8,3,6,8,9};
  int found=1;
  //判断方阵是否为对称阵，若不是，found 置为 0
  for(int i = 0; i < n; ++ i){
       for(int j = 0; j < n; ++ j){
             printf("%d ", a[i][j]);
         }
         puts("");
  }

  puts("");

  for(int i = 0; i < n; ++ i){
       for(int j = 0; j < n; ++ j){
             if(a[i][j] != a[j][i]){
                   found = 0;
                   break;
              }
         }
         if(!found) break;
  }
  if (found==0)
     printf("no\n");
  else
     printf("yes\n");
return 0;}
```



4.

```
#include<stdio.h>

const int n = 5;

int main()
{ int num[5]={1,3,5,4,2};

   int *p;

   int max = 0;
   for(int i = 0; i < n; ++ i)
      if(num[i] > max){
            max = num[i];
            p = num + i;
      }
     int t = *p;                          //指针交换
     *p = num[n - 1];
     num[n - 1] = t;

     for(int i = 0; i < n; ++ i) printf("%d ", *(num + i));
   return 0;
}
```

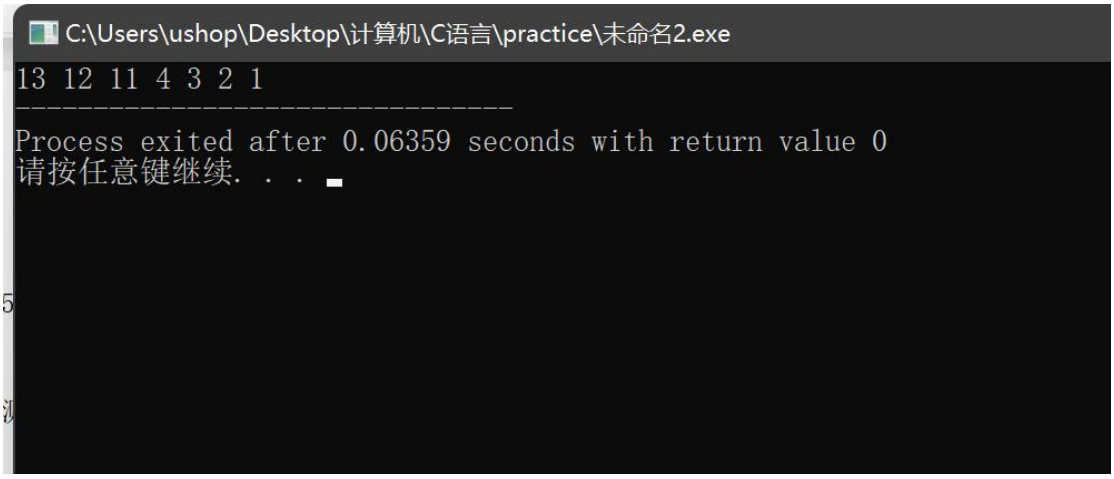


5.

```
#include<stdio.h>

#define N 7

int main()
{
   int a[N]={1, 2, 3, 4, 11, 12, 13};

   int *p = a;

   for(int i = 0; i < N / 2; ++ i){
         int t = *(p + i);
         *(p + i) = *(p + N - 1 - i);
         *(p + N - 1 - i) = t;
   }

   for(int *p = a; p < a + N; ++ p) printf("%d ", *p);    //偏移

   return 0;
}
```

6.

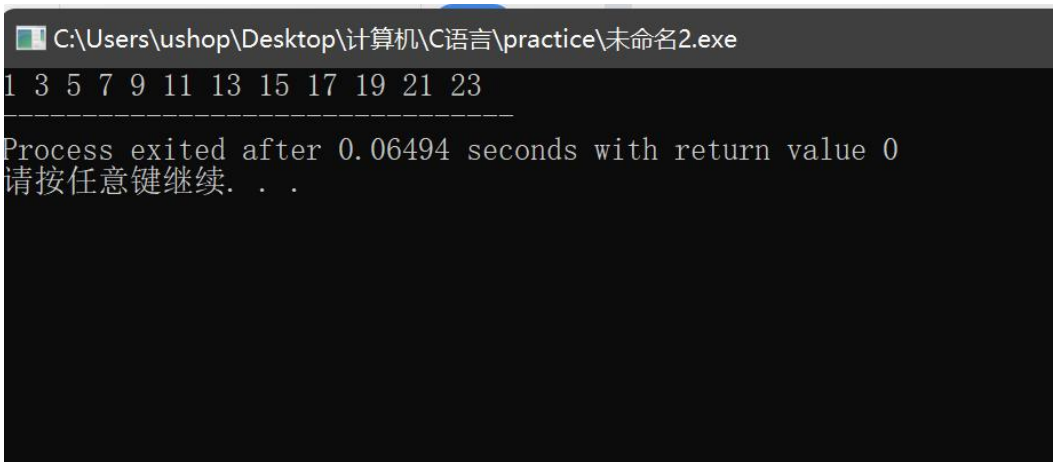
```
#include<stdio.h>

#define N 7

int main()
{
   int a[N]={1, 2, 3, 4, 11, 12, 13};

   int *p = a;

   for(int i = 0; i < N / 2; ++ i){                    //指针移动
         int t = *(p + i);
         *(p + i) = *(p + N - 1 - i);
         *(p + N - 1 - i) = t;
   }

   for(int *p = a; p < a + N; ++ p) printf("%d ", *p);

   return 0;
}
```



7.

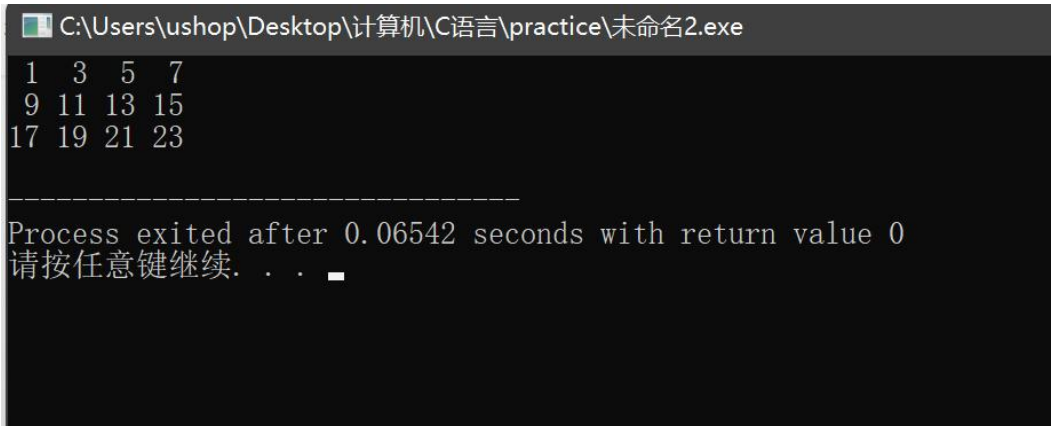
```
#include<stdio.h>

#define N 12

int main()
{
```

```
    int a[3][4]={1,3,5,7,9,11,13,15,17,19,21,23};

    for(int *p = a[0]; p < a[0] + 12; ++ p){
        printf("%d ", *p);
    }

    return 0;
}
```



8.

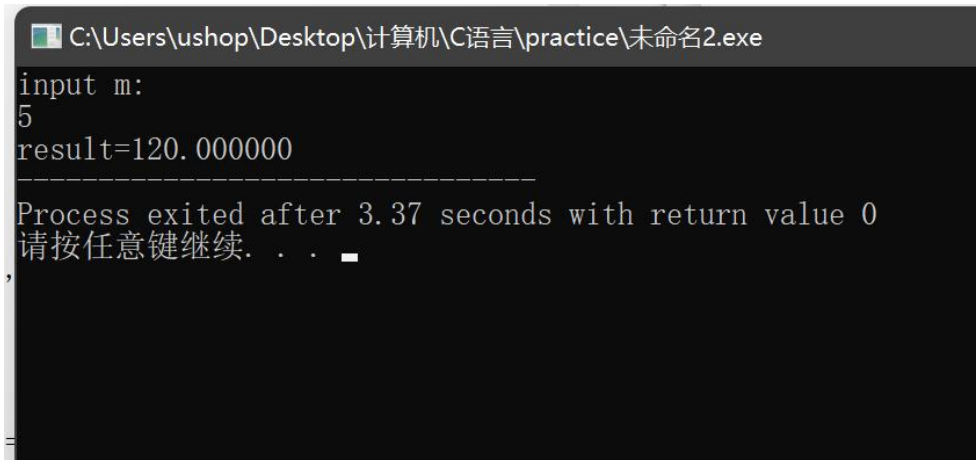
```
#include<stdio.h>

#define N 12

int main()
{
    int a[3][4]={1,3,5,7,9,11,13,15,17,19,21,23};

    int (*p)[4] = a;                    //行指针

    for(int i = 0; i < 3; ++ i){
        for(int j = 0; j < 4; ++ j){
                printf("%2d ", *(*(p + i) + j));
            }
        puts("");
    }

    return 0;
}
```



9.

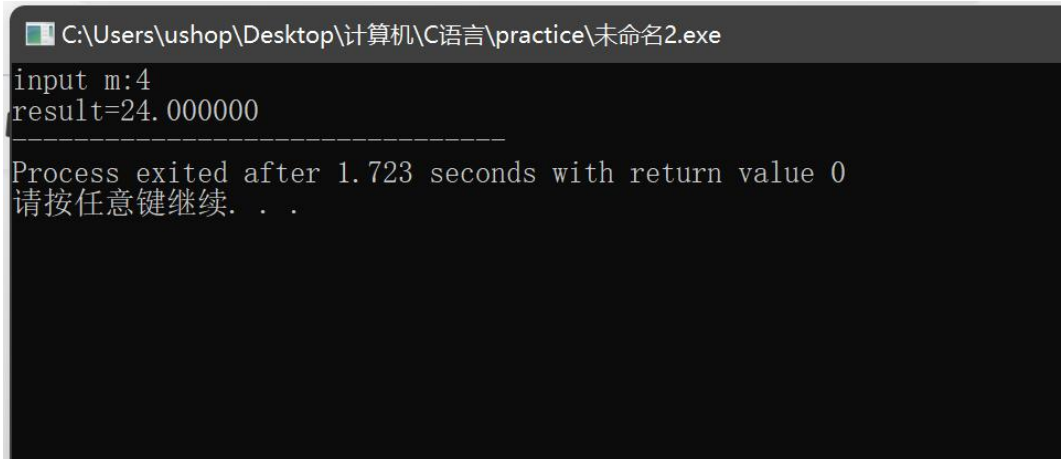
（1）

```
#include<stdio.h>

#define N 12

int fac(int n){
	if(n == 1) return 1;
	return n * fac(n - 1);		//递归
}

int main( )
{
   int m;
   float k;
   printf("input m:");
   scanf("%d",&m);

   k = fac(m);

   printf("result=%f",k);
}
```



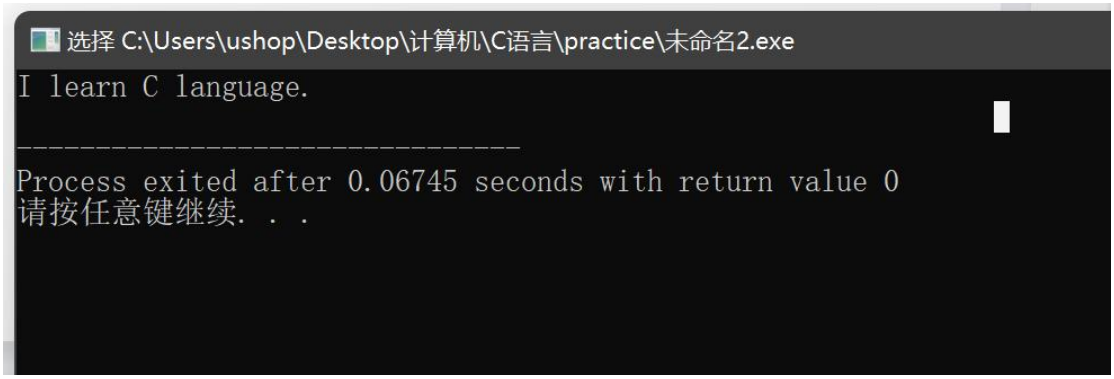
（2）

```
#include<stdio.h>     //通过指针传地址间接改变值

#define N 12

void fac(int *n){
	int sum = 1;
	for(int i = *n; i >= 1; -- i){
		sum *= i;
	}
	*n = sum;                    //利用指针可以不用返回值
}

int main( )
{
   int m;
   float k;
   printf("input m:");
   scanf("%d",&m);

   fac(&m);
   k = m;

   printf("result=%f",k);
}
```



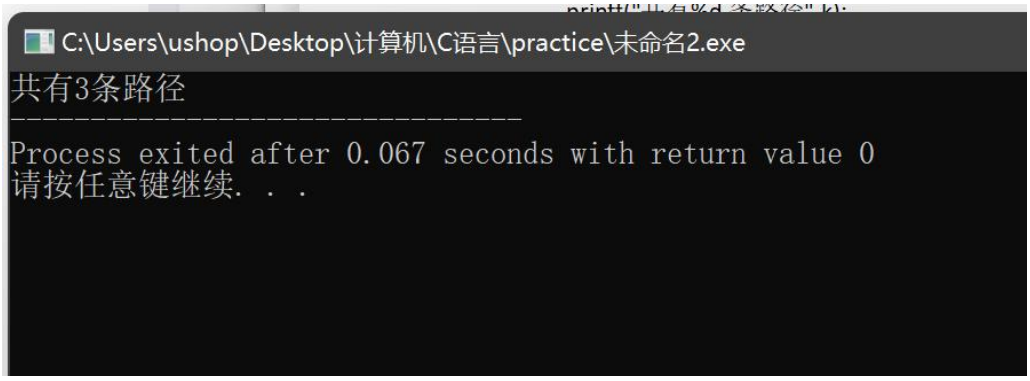
10.

```
#include<stdio.h>
#include<string.h>
#define N 12

char *strcat(char *s1, char *s2){
	char *s = s1;
	int i = sizeof(s1);

	for(int j = 0; s2[j] != '\0'; ++ j){
		s[i++] = s2[j];				//一个一个字符传送
	}

	s[i] = '\0';

	return s;

}


int main( )
{
   char str1[30] = "I learn ", *str2 = "C language.";
   char *s;

   s = strcat(str1,str2);//strcat 函数的返回值是指针
   printf("%s\n", s);
}
```



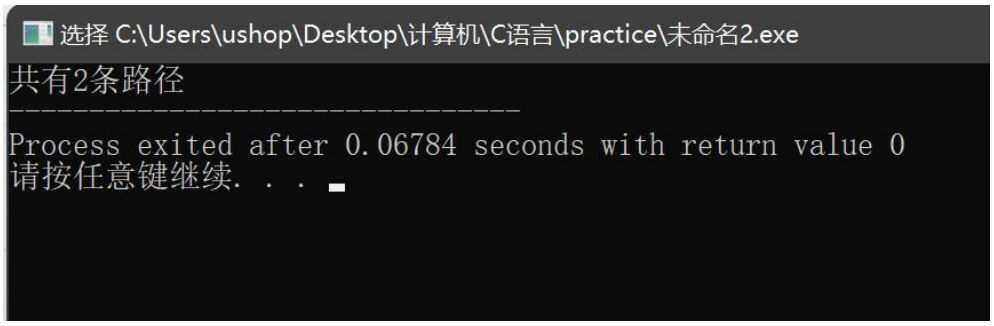
11.

```
#include<stdio.h>
#include<string.h>


int uniquePaths(int m, int n) {
   //补充函数，返回路径数

   if(m == 1 || n == 1) return 1;            //边界出口
   return uniquePaths(m - 1, n) + uniquePaths(m, n - 1);       //两种选择
}
int main(){
       int m=3,n=2;
       int k=uniquePaths(m,n);
       printf("共有%d 条路径",k);
       return 0;
}
```



12.

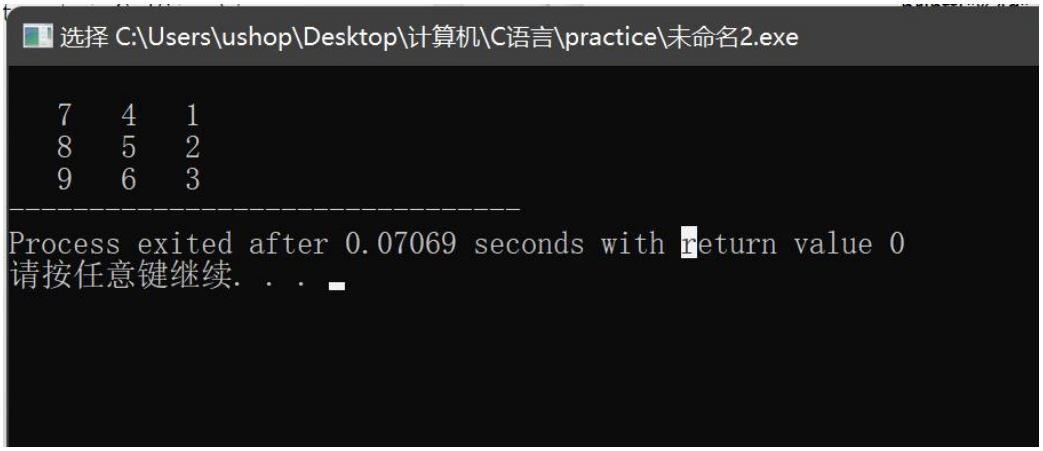
```
#include<stdio.h>
#include<string.h>


int uniquePathsWithObstacles(int* obstacleGrid, int obstacleGridRowSize, int obstacleGridColSize){
  //补充函数，返回路径数
    int r = obstacleGridRowSize, c = obstacleGridColSize;
    int *b = obstacleGrid;
    if( *(b + (r - 1) * 3 + c - 1) == 1) return 0;        //障碍物不能走
    if(r == 1 || c == 1) return 1;
    return uniquePathsWithObstacles(b, r - 1, c) + uniquePathsWithObstacles(b, r, c - 1);

}
int main(){
      int a[3][3]={0,0,0,0,1,0,0,0,0};
      int k=uniquePathsWithObstacles(a[0],3,3);
      printf("共有%d 条路径",k);

      return 0;
}
```



13.

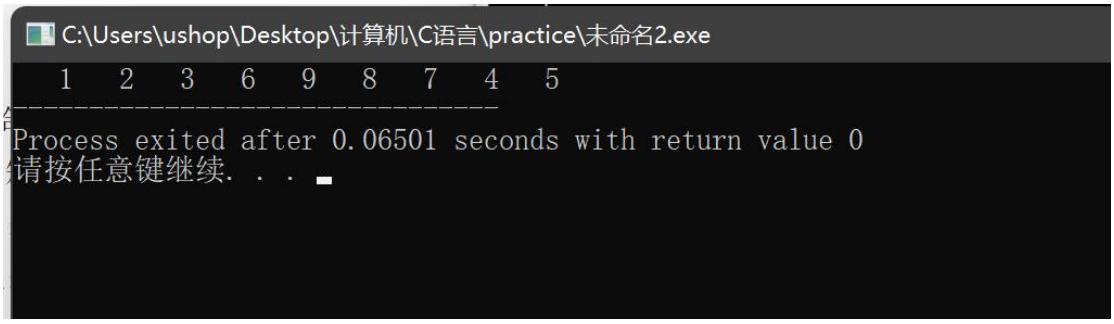
```
#include<stdio.h>
#include<string.h>

void rotate(int* matrix, int matrixRowSize, int matrixColSize){
    //补充函数，实现原地旋转功能
    //先上下颠倒再转置
    for(int i = 0; i < matrixRowSize / 2; ++ i){
```

```
        for(int j = 0; j < matrixColSize; ++ j){              //上下颠倒
                int t = *(matrix + i * matrixColSize + j);
                *(matrix + i * matrixColSize + j) = *(matrix + (matrixRowSize - i - 1) * matrixColSize +
j);
                *(matrix + (matrixRowSize - i - 1) * matrixColSize + j) = t;
            }
    }

    for(int i = 0; i < matrixColSize; ++ i){              //转置
      for(int j = 0; j < i; ++ j){
            int t = *(matrix + i * matrixColSize + j);
                *(matrix + i * matrixColSize + j) = *(matrix + j * matrixColSize + i);
                *(matrix + j * matrixColSize + i) = t;
           }
    }
}

int main(){
      int a[3][3]={1,2,3,4,5,6,7,8,9};
      rotate(a[0],3,3);

      int *p;
      for(p=a[0];p<a[0]+9;p++)
      {

            if((p-a[0])%3==0) printf("\n");

            printf("%4d",*p);
       }
      return 0;
}
```



14.

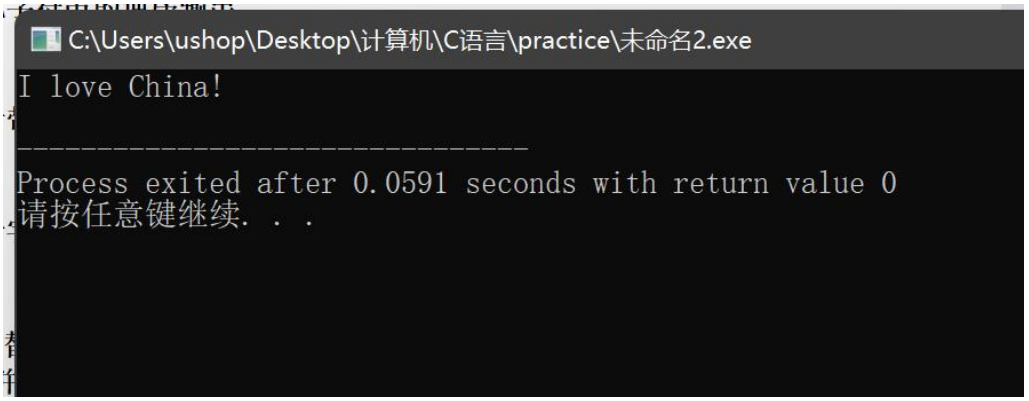
```
#include<stdio.h>
#include<string.h>

//给定一个包含 m x n 个元素的矩阵（m 行, n 列），请按照顺时针螺旋顺序，返回矩阵中的所有元素。

int res[100];   //不放全局变量会出错

int *spiralOrder(int* matrix, int matrixRowSize, int matrixColSize)
{

    int c1 = 0, r1 = 0, rh = matrixRowSize - 1, ch = matrixColSize - 1, count = 0;

    while(1)            //蛇形 4 个方向一个循环
    {
        for(int j = c1; j <= ch; ++ j) res[count ++] = *(matrix + c1 * matrixColSize + j);
        if(++ r1 > rh) break;
        for(int i = r1; i <= rh; ++ i) res[count ++] = *(matrix + i * matrixColSize + ch);
        if(-- ch < c1) break;
        for(int j = ch; j >= c1; -- j) res[count ++] = *(matrix + rh * matrixColSize + j);
        if(-- rh < r1) break;
        for(int i = rh; i >= r1; -- i) res[count ++] = *(matrix + i * matrixColSize + c1);
        if(++ c1 > ch) break;

    }
//  for(int i = 0; i < 9; ++ i) printf("%d ", res[i]);
//  puts("");
    return res;
}
int main(){
    int a[3][3]={1,2,3,4,5,6,7,8,9};
    int matrixRowSize=3,matrixColSize=3;
    int *returnnum=spiralOrder(a[0], matrixRowSize, matrixColSize);
    int *p;
    for(p=returnnum;p<returnnum+9;p++)
    {
        printf("%4d",*p);
     }
    return 0;
}
```



二、字符串与指针

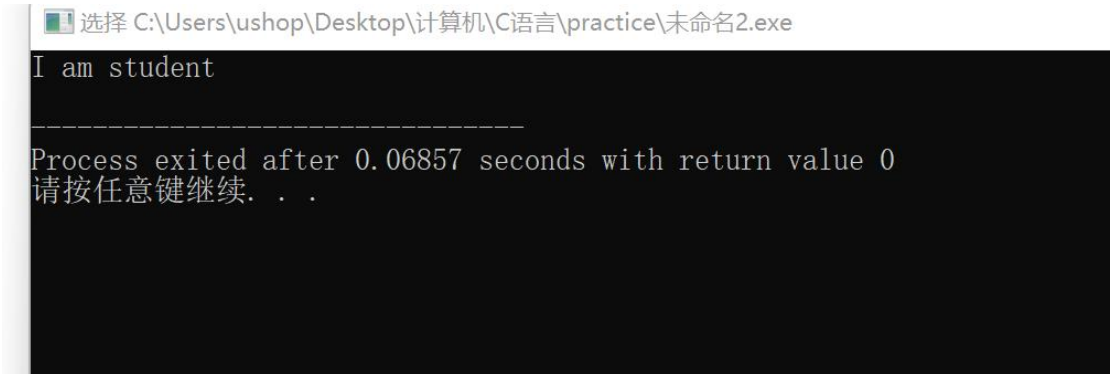
1.

```
#include<stdio.h>
#include<string.h>

int main(){
	char *s = "I love China!";
	puts(s);
	return 0;
}
```



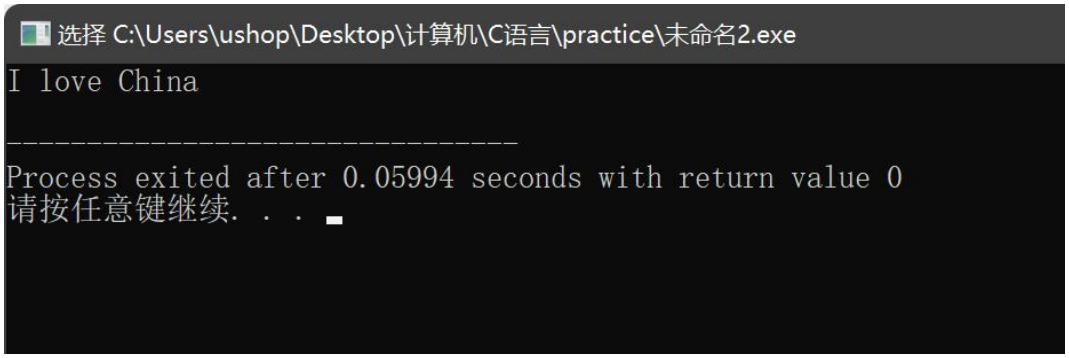
2.

```
#include<stdio.h>
#include<string.h>


int main(){
	char *a = "I am student";
	char *b = a;                    //让 b 指向 a 的地址
	puts(b);
	return 0;
}
```



3.

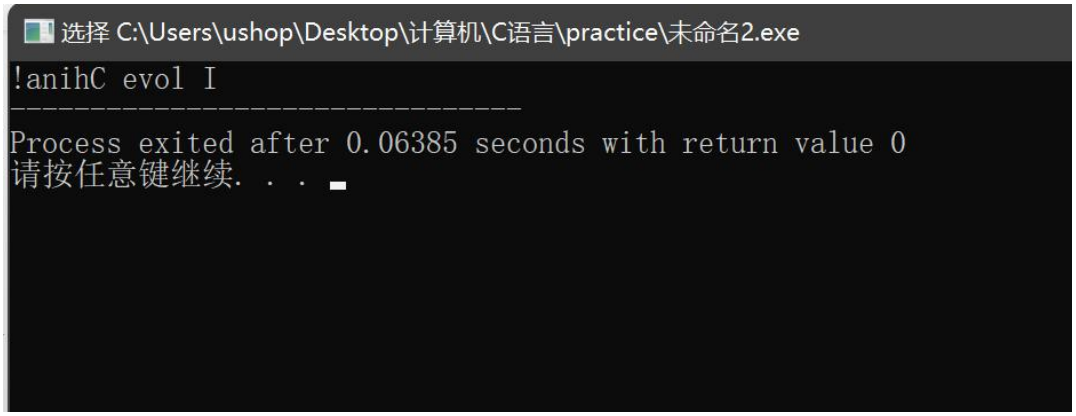
```
#include<stdio.h>
#include<string.h>


int main(){
	char a[50] = "I love ";   //用字符数组形式或者申请空间  否则无法实现
	char b[] = "China";
	strcat(a, b);
	puts(a);
	return 0;
}
```



4.

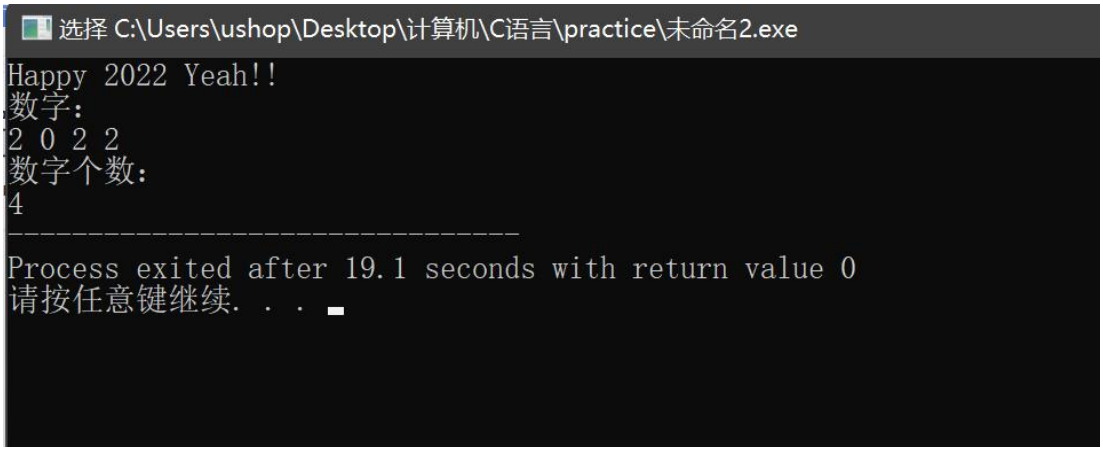
```
#include<stdio.h>
#include<string.h>
```

```
int main(){
	char a[] = "I love China!";  //如果用字符数组，那开的数组大小就是字符串的大小
	//printf("%d", sizeof(a));
	for(int i = sizeof(a) - 2; i >= 0; -- i) //最后一位是休止符
	printf("%c", a[i]);  //不能用 a + i 因为这是单个字符输出 但 a + i 是字符串
	return 0;
}
```



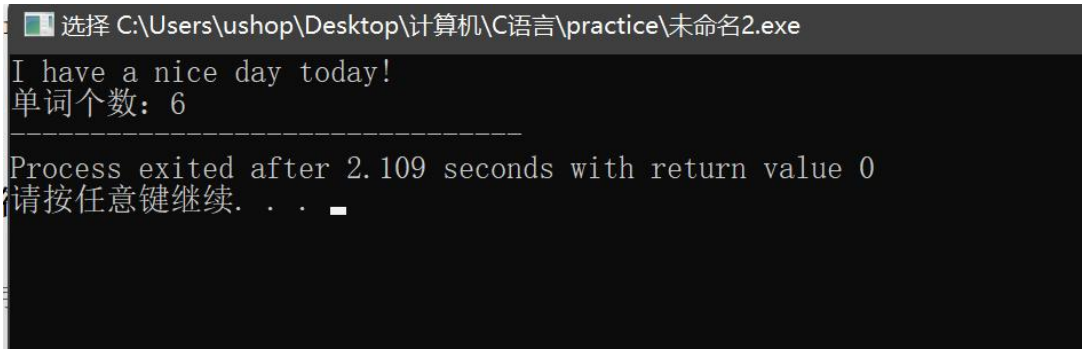
5.

```
#include<stdio.h>
#include<string.h>


int main(){
	char s[100];
	gets(s);
	int cnt = 0;                                    //计数
	puts("数字： ");
	for(int i = 0; i < strlen(s); ++ i){
		if(s[i] >= '0' && s[i] <= '9'){
			++ cnt;
			printf("%c ", s[i]);
		}
	}
	puts("");
	puts("数字个数： ");
	printf("%d", cnt);
	return 0;
}
```



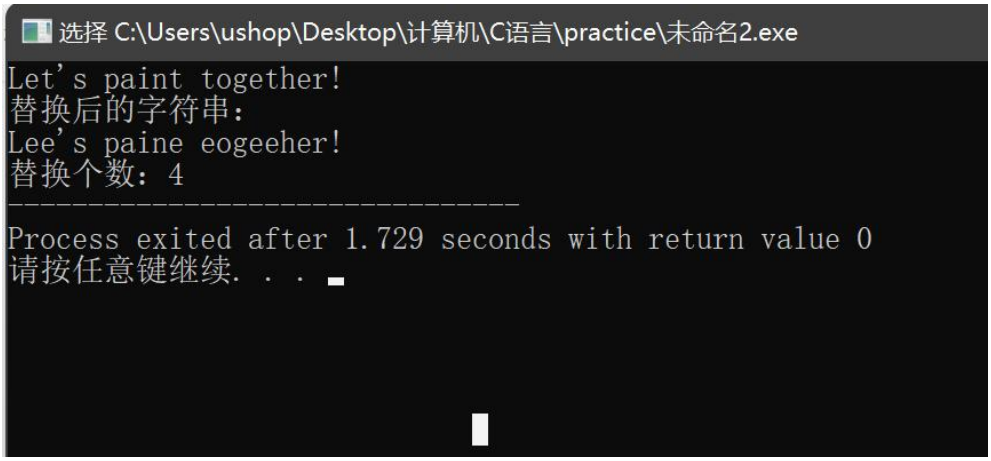
6.

```
#include<stdio.h>
#include<string.h>


int main(){
	char s[100];
	gets(s);
	int cnt = 0;

	bool word = true;	//标记单词首字母

	for(int i = 0; s[i] != '\0'; ++ i){
		if(s[i] != ' ' && word){	//不是空格且是单词首字母
			++ cnt;
			word = false;	//防止重复统计
		}
		else if(s[i] == ' '){	//是空格则把标记量改为真
			word = true;
		}
	}
	printf("单词个数：");
	printf("%d", cnt);

	return 0;
}
```



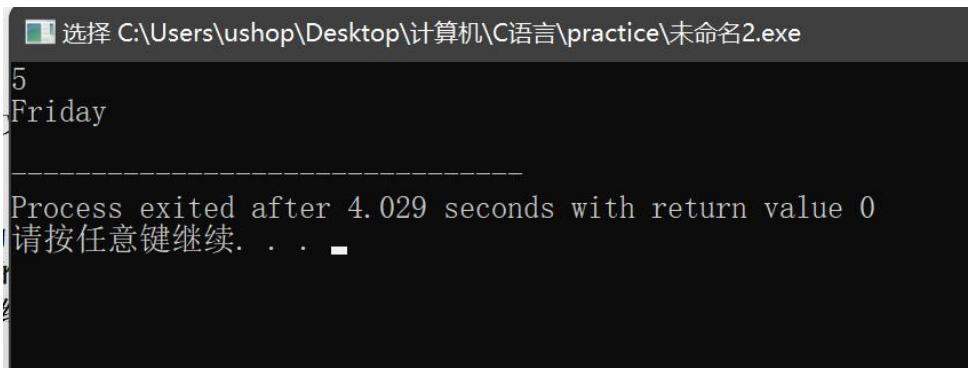
7.

```
#include<stdio.h>
#include<string.h>


int main(){
	char s[100];
	gets(s);
	int cnt = 0;

	for(int i = 0; s[i] != '\0'; ++ i){
		if(s[i] == 't'){
			s[i] = 'e'; ++ cnt;
		}
		else if(s[i] == 'T'){
			s[i] = 'E'; ++ cnt;
		}
	}
	puts("替换后的字符串：");
	puts(s);

	printf("替换个数：");
	printf("%d", cnt);

	return 0;
}
```



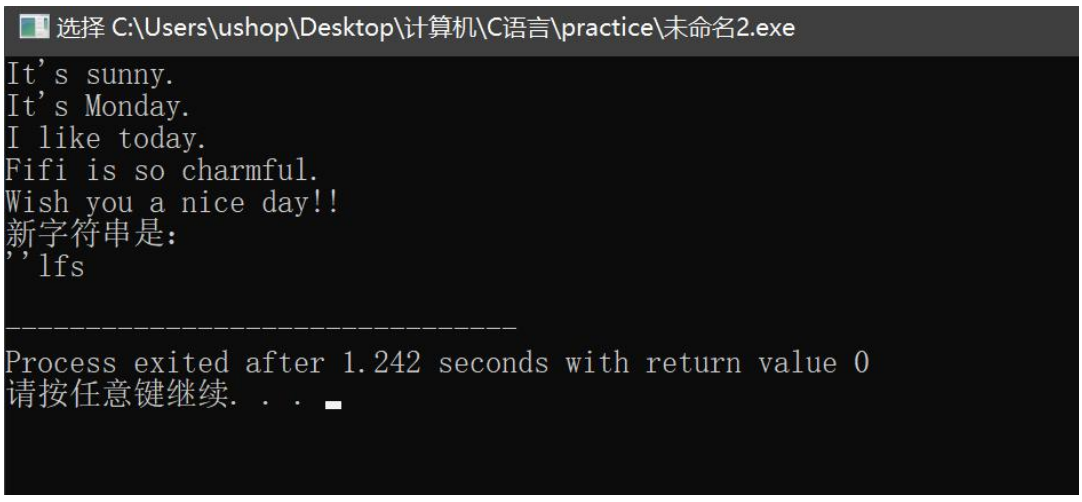
8.

```
#include<stdio.h>
#include<string.h>

char s[7][10] = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Satday", "Sunday"};
                                        //字符串常量
int main(){
    int week;
    scanf("%d", &week);
    char (*p)[10] = s;
    puts(*(p + week - 1));

    return 0;
}
```



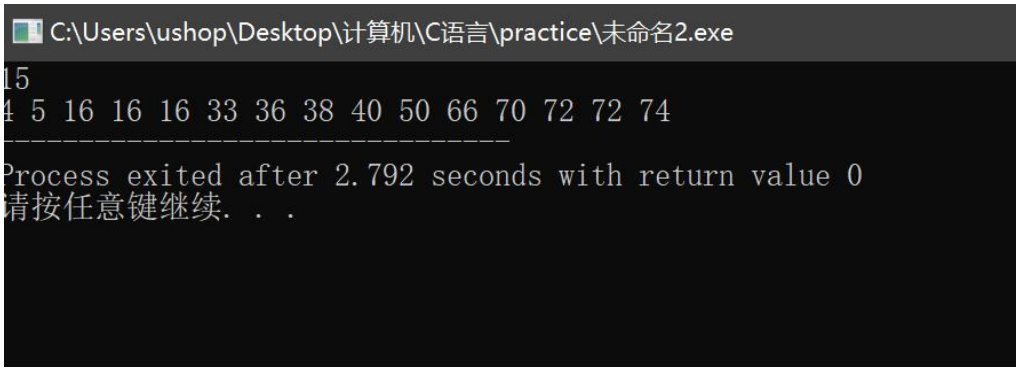
9.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

const int N = 105;

int cmp(const void *a, const void *b){                  //自定义比较
    return strlen((char*)a) > strlen((char*)b);
}

int main(){
    char s[5][N];

    for(int i = 0; i < 5; ++ i) gets(s[i]);

    qsort(s, 5, sizeof(s[0]), cmp);            //快排

    char s1[N];

    for(int i = 0; i < 5; ++ i){
        if(strlen(s[i]) < 3) s1[i] = ' ';
        else s1[i] = *(s[i] + 2);
    }
    puts("新字符串是：");
    puts(s1);

    return 0;
}
```



10.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>

void swap(int *a, int *b){
	int t = *a;
	*a = *b;
	*b = t;
	return;
}

void Arr(int *array, int n){
	srand((int) time(0));		//随机函数
	for(int i = 0; i < n; ++ i){
		array[i] = rand() % 100;
	}
	for(int i = 0; i < n - 1; ++ i){
		for(int j = 0; j < n - i - 1; ++ j){
			if(array[j] > array[j + 1]) swap(&array[j], &array[j + 1]);
		}
	}
	return;
}

int main(){

	int n;
	int *array = NULL;

	scanf("%d", &n);
	array = (int*)malloc(sizeof(int) * n);

	Arr(array, n);

	for(int *p = array; p < array + n; ++ p) printf("%d ", *p);

	return 0;
}
```



三、位操作

1. 判断系统是逻辑右移还是算术右移。

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>

int main() {
	char x=0xfe;	//-2
	int y=x>>1;
	printf("%d",y);

	return 0;
}
```

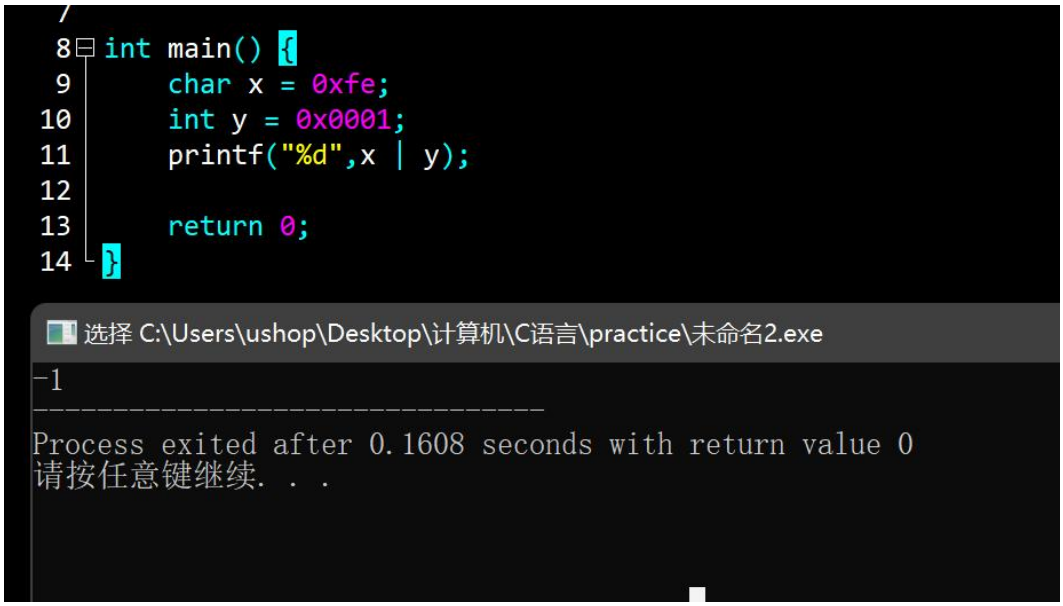
因为输出是-1，说明带符号，则首位是 1。所以我的系统是算术右移（高位补 1）。

2.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>
```

```
int main() {
	char x = 0xfe;
	int y = 0x0001;
	printf("%d",x | y);

	return 0;
}
```

x 是 char 类型的负数，y 是 int 类型。x 比 y 短。从结果来看，输出的是-1，说明结果 8 位之后的高位是 0，这说明在进行逻辑运算时，本系统会在较短的负数前面补 0.

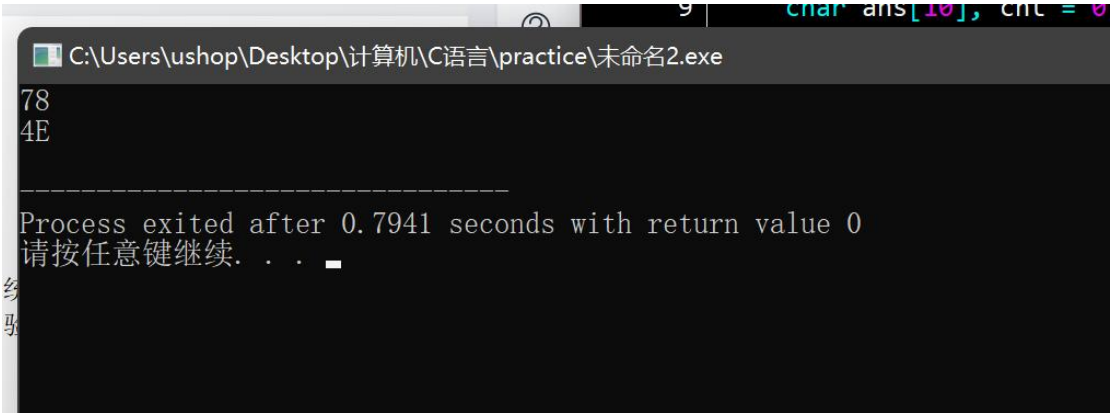


3.

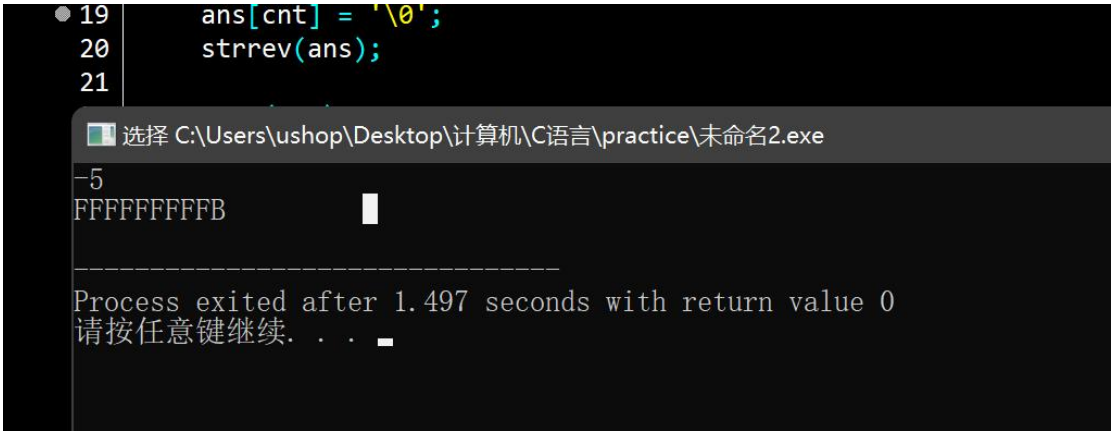
```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

const char str[] = "0123456789ABCDEF";

int main() {
	int x;
	char ans[10], cnt = 0;
	scanf("%d", &x);
	//printf("%d\n", x >> 4);

	while(x){
		int i = x & 15; //printf("%d\n", x);	//与 15 相与可以取出后四位
		ans[cnt ++] = str[i];
```

```
        x >>= 4; //printf("%d\n", x);     //右移四位处理后续
    }

    ans[cnt] = '\0';
    strrev(ans);          //翻转

    puts(ans);

    //printf("%x", x);
    return 0;
}
```

正数：



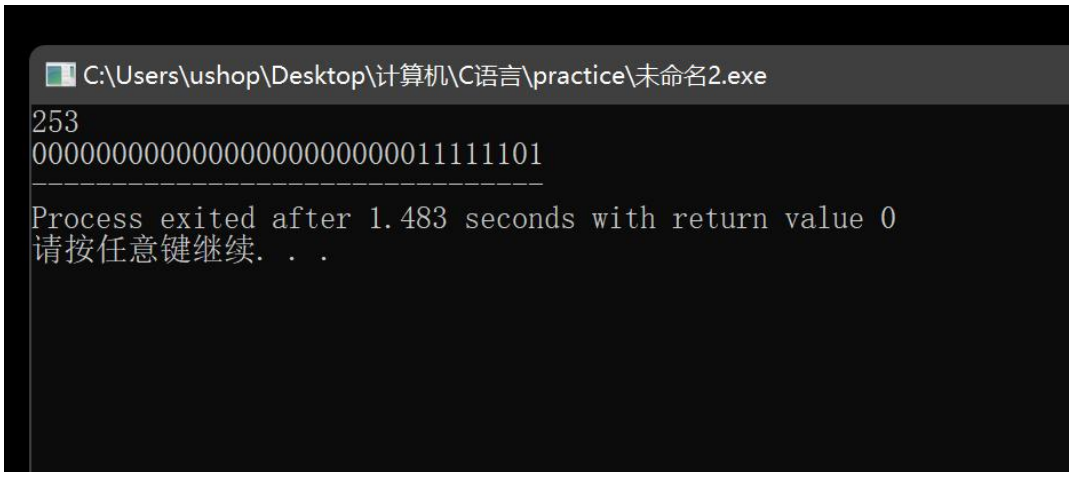
负数：



4.

```
#include<stdio.h>

int main() {
    int x;

    scanf("%d", &x);

    unsigned i = 1 << 31;    //从 int 的最高位开始（是 31 不是 32）
```

```
    for(; i; i >>= 1){    //每次往右移动一位（不要忘记等于号）
        printf("%d", x & i ? 1 : 0);    //如果该位是 1 则相与结果大于 0（是实际数字）
    }

    return 0;
}
```

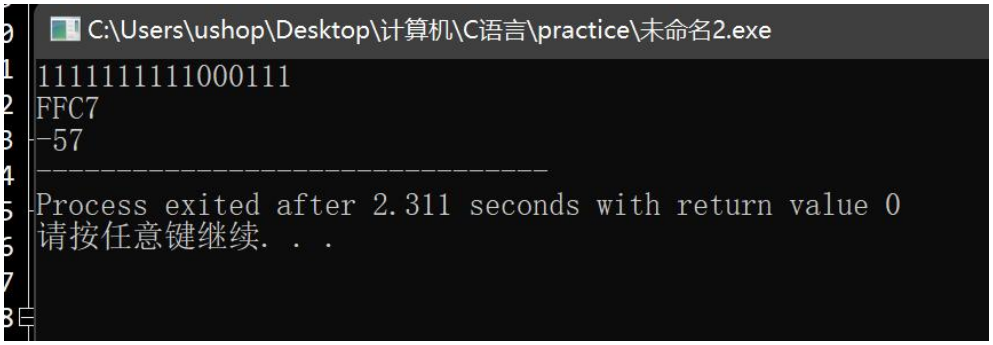


5.

```
#include<stdio.h>
#include<string.h>
#include<math.h>

const char str[20] = "0123456789ABCDEF";


int main() {

    char s[17], ansx[4], s1[17];

    int ans = 0, tmp;
    bool flag = false;

    gets(s);
    strcpy(s1, s);

    for(int i = 0; i < 16; i += 4) {     //十六进制部分
        int tmp = 0;
        for(int j = 0; j < 4; ++ j) {
            if(s[i + j] == '1') {
                tmp += pow(2, 3 - j);
```

```
                }
            }
            ansx[i / 4] = str[tmp];
        }

        ansx[4] = '\0';    //两个字符串的存储地址是挨着的
        puts(ansx);
        strcpy(s, s1);


        //printf("%c\n", s[0]);

        if(s[0] == '1') {     //处理负数
            flag = true;
            if(s[15] == '1') s[15] = '0';
            else {
                s[15] = '1';
                int i = 14;
                while(s[i] = '0') s[i --] = '1';
                s[i] = '0';
            }
            for(int i = 0; i < 16; ++ i) s[i] = '0' + '1' - s[i];
        }
        //puts(s);

        for(int i = 0; i < 16; ++ i) {
            if(s[i] == '1')
                ans += pow(2, 15 - i);
        }
        if(flag) printf("-");
        printf("%d", ans);

        return 0;
}
```

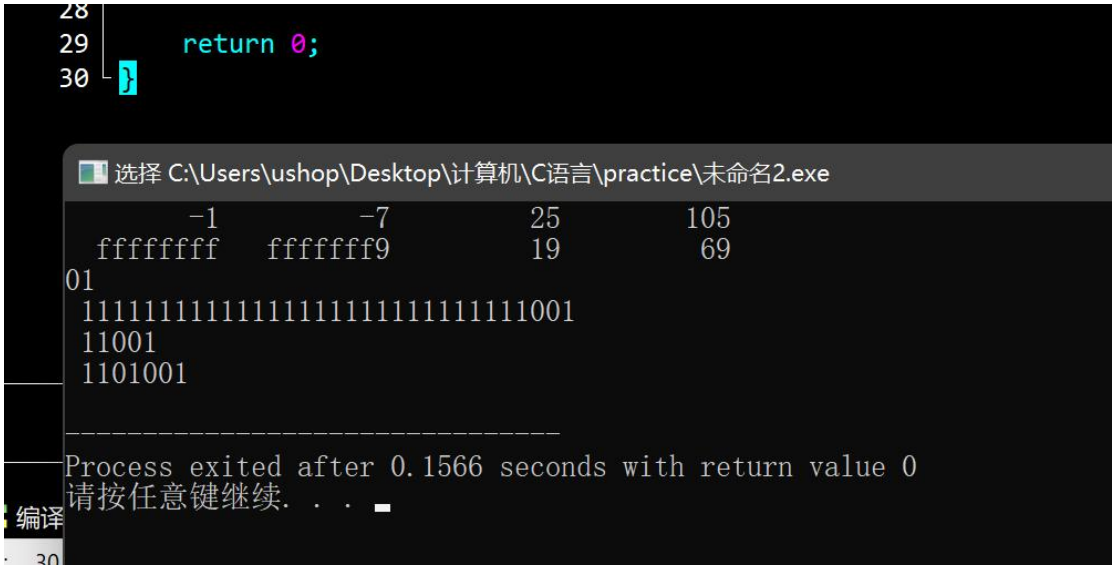
```
C:\Users\ushop\Desktop\计算机\C语言\practice\未命名2.exe
0000000000111001
0039
57
--------------------------------
Process exited after 12.85 seconds with return value 0
请按任意键继续. . .
```



6.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

struct S {
	int a : 3;
	int b : 5;
	int c : 6;
	int d : 9;
}st;


int main() {

	st.a = 95; st.b = 25; st.c = 25; st.d = 105;

	printf("%10d %10d %10d %10d\n", st.a, st.b, st.c, st.d);
	printf("%10x %10x %10x %10x\n", st.a, st.b, st.c, st.d);

	char s1[30], s2[30], s3[30], s4[30];
	itoa(st.a, s1, 2);
	itoa(st.b, s2, 2);
	itoa(st.c, s3, 2);
	itoa(st.d, s4, 2);
	printf("%s\n %s\n %s\n %s\n", s1, s2, s3, s4);

	return 0;
}
```

//高位会截断



7.

```
#include <stdio.h>
#include <string.h>
#include <math.h>

char str[42], s[40];


int main() {
	gets(str);
	int n = 0;

	for(int i = 0; str[i] != '\0'; ++ i){	//去掉空格
		if(str[i] != ' '){
			s[n ++] = str[i];
		}
	}

	if(s[0] == '1') printf("-");	//判断符号位

	int e = 0;
	for(int i = 1; i < 9; ++ i){		//计算指数位
		e = e * 2 + s[i] - '0';
	}
	e -= 127;

	int zs = 1;				//因为 IEEE 标准制会去掉首位的 1，所以这里初始值设置为 1
	float xs = 0.0, ans = 0.0;	//zs 是整数，xs 是小数，ans 存答案
```

```
    for(int i = 9; i < 9 + e; ++ i){       //整数处理
        zs = zs * 2 + s[i] - '0';
    }

    double w = 0.5;                                 //注意不要用 int
    for(int i = 9 + e; i < n; ++ i){       //小数处理
        xs += (s[i] - '0') * w;
        w /= 2;

    }

    ans = zs + xs;

    printf("%f", ans);

    return 0;
}
```