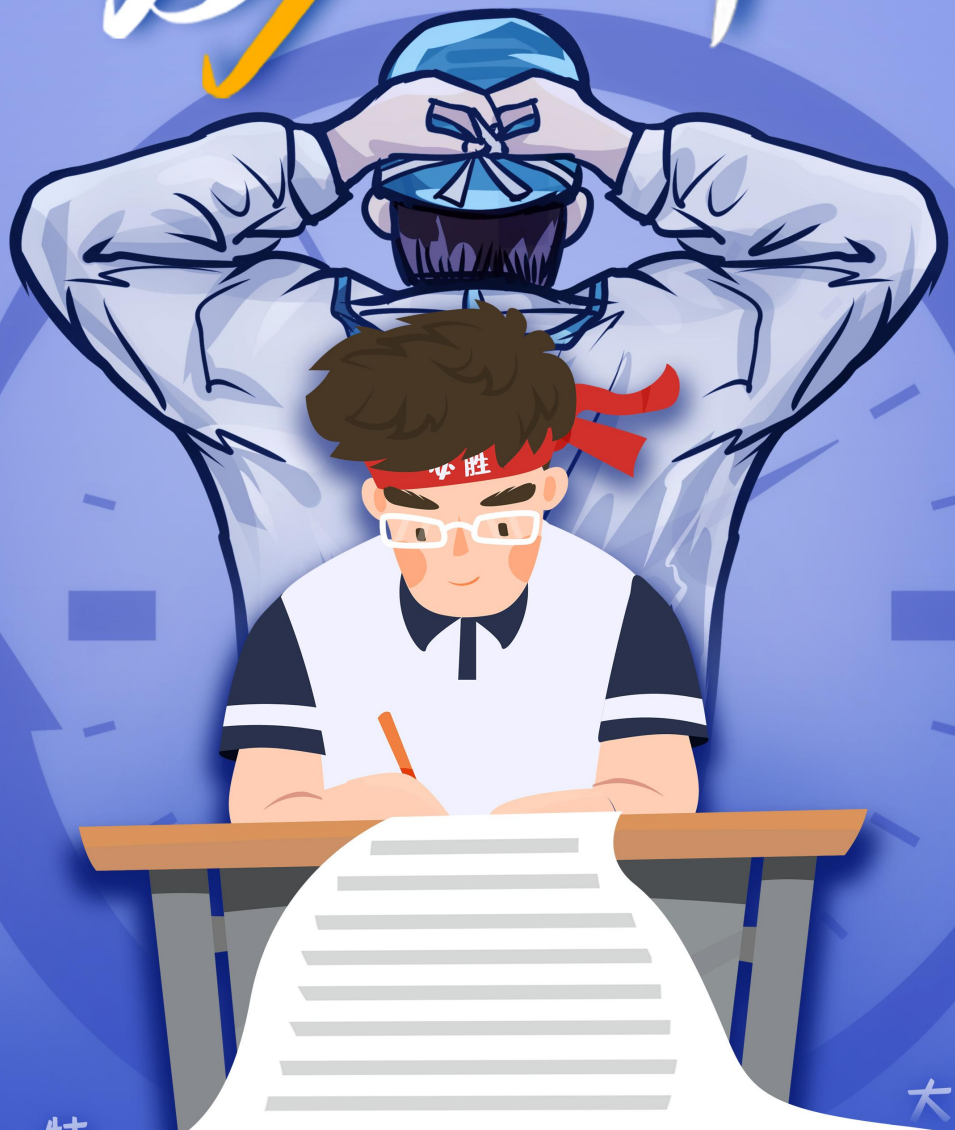
比特就业课 | 专注IT大学生就业的精品课程
www.bitejiuyeke.com

为梦想而战



小比特 大梦想

如果抗击疫情、救死扶伤是医疗逆行者的使命
那么努力奋斗、不断进步就是我们不可推卸的责任
做学习上的逆行者，向各个行业优秀的人看齐

Hello 2022 比特

比特就业课假期作业-C语言作业

出题老师：

> C选择题：黄坤（day01-day16） qq：3587670086
>
> C编程题：张文超（day01-day16）qq：3627274478

作业说明：

> 1、本次作业涵盖内容为C语言相关知识点
>
> 2、如果对试卷上的题目，或者答案有问题，可以联系对应老师哦~~
>
> 3、同学们添加老师时备注：姓名+比特班级哦~

day01

一、选择题

1、执行下面程序，正确的输出是（ C ）

```c
int x=5,y=7;
void swap()
{
    int z;
    z=x;
    x=y;
    y=z;
}
int main()
{
    int x=3,y=8;
    swap();
    printf("%d,%d\n", x, y);
    return 0;
}

/*解析:
存在全局变量 x = 5, y = 7;
main函数存在局部变量 x = 3, y = 8;
main函数中调用 swap 函数 但未传参
所以使用swap函数交换两全局变量x、y的值
但，main函数中输出 x、y未局部变量
所以输出局部变量x、y原值
即 3,8
*/
```

A: `5,7` B: `7,5` C: `3,8` D: `8,3`

2、以下不正确的定义语句是（ B ）

A: `double x[5] = {2.0, 4.0, 6.0, 8.0, 10.0};`

B: `char c2[] = {'\x10', '\xa', '\8'};`

C: `char c1[] = {'1','2','3','4','5'};`

D:`int y[5+3]={0, 1, 3, 5, 7, 9};`

3、`test.c`文件中包括如下语句，文件中定义的四个变量中，是指针类型的变量为【多选】（ `A,C,D` ）

```
#define INT_PTR int*
typedef int* int_ptr;
INT_PTR a, b;
int_ptr c, d;

/*
#define(宏定义)只是简单的字符串代换，并不在编译过程中进行，而是在预处理过程就已经完成了
typedef是为了增加可读性而为标识符另起的新名称，它具有一定的封装性，它是语言编译过程的一部分，但它并不实际分配内存空间。

所以 #define INT_PTR int* 仅仅是将 int* 换了一个名字
typedef int* int_ptr;    则是将int* 封装至int_ptr
所以使用 INT_PTR a, b;  与 int* a, b; 相同，相当于int* a; int b; 仅仅使 a 定义为整型指针
int_ptr c, d;    则相当于int* c; int* d;      c、d均定义为整型指针
*/
```

A:`a`　　B:`b`　　C:`c`　　D:`d`

4、 若给定条件表达式`(M)?(a++):(a--)`，则其中表达式`M`（ `C` ）

A:`和(M==0)等价`　　B:`和(M==1)等价`　　C:`和(M!=0)等价`　　D:`和(M!=1)等价`

5、有如下定义语句，则正确的输入语句是【多选】（ `A,B` ）

```
int b;
char c[10];
```

A:`scanf("%d%s",&b,&c);`　　B:`scanf("%d%s",&b,c);`

C:`scanf("%d%s",b,c);`　　D:`scanf("%d%s",b,&c);`

二、编程题

输入数字`n`，按顺序打印出从`1`到最大的`n`位十进制数。比如输入`3`，则打印出`1、2、3`一直到最大的`3`位数`999`。

- 用返回一个整数列表来代替打印
- `n`为正整数

OJ链接【牛客网题号：`JZ17`打印从1到最大的n位数】【难度：入门】

```
示例：
    输入：1
    返回值：[1,2,3,4,5,6,7,8,9]
```

```
//输入数字 n，按顺序打印出从 1 到最大的 n 位十进制数。比如输入 3，则打印出 1、2、3 一直到最大的 3 位数 999。
//1. 用返回一个整数列表来代替打印
//2. n 为正整数

/**
```

```
 * 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可
 *
 *
 * @param n int整型 最大位数
 * @return int整型一维数组
 * @return int* returnSize 返回数组行数
 *
 * C语言声明定义全局变量请加上static，防止重复定义
 */
static int arr[1000000];
int* printNumbers(int n, int* returnSize )
{
    int num = 1;
    for(int i = 0; i < n; i++)
    {//num 为输出上限
        num *= 10;
    }
    for(int i = 1; i < num; i++)
    {//从1到num 给数组赋值
        arr[i - 1] = i;
    }
    *returnSize = num - 1;
    return arr;
}
```

2、根据输入的日期，计算是这一年的第几天。输入保证年份为4位数且日期合法。

输入描述：输入一行，每行空格分割，分别是年，月，日。

输出描述：输出是这一年的第几天

OJ链接【牛客网题号：`HJ73` 计算日期到天数转换】【难度：简单】

```
示例：
    输入：2012 12 31          输入：1982 3 4
    输出：366                 输出：63
```

```
//根据输入的日期，计算是这一年的第几天。
//保证年份为4位数且日期合法。

#include <stdio.h>

int main()
{
    int Days[12]  = {31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334,
365};
    int year = 0;
    int month = 0;
    int day = 0;
    while(scanf("%d%d%d", &year, &month, &day) != EOF)
    {
        int days = 0;
        if(month >= 2)
            days = Days[month - 2] + day;
        else
```

```
            days = day;
        if( ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) &&
((month >2) || (month == 2 && day == 29)))
            days++;
        printf("%d\n", days);
    }

    return 0;
}
```

day 02

一、选择题

1、以下程序段的输出结果是（ A ）

```
#include<stdio.h>
int main()
{
    char s[] = "\\123456\123456\t";
    printf("%d\n", strlen(s));

    return 0;
}
// 解析：'\\'是一个字符 '\123'是一个字符 '\t'是一个字符
```

A:12　　B:13　　C:16　　D:以上都不对

2、若有以下程序，则运行后的输出结果是（ B ）

```
#include <stdio.h>
#define N 2
#define M N + 1
#define NUM (M + 1) * M / 2
int main()
{
    printf("%d\n", NUM);

    return 0;
}
// 解析：#define(预处理)不封装
// 所以 NUM = (N + 1 + 1) * N + 1 / 2 === 4 * 2 + 0.5 === 8.5
// 输出整型 8
```

A:4　　B:8　　C:9　　D:6

3、如下函数的 f(1) 的值为（ C ）

```
int f(int n)
{
     static int i = 1;
     if(n >= 5)
         return n;
     n = n + i;
```

```
        i++;
        return f(n);
}
/*
f(1):
n = 1; i = 1;
n < 5
n += i ==> n = 2
i++ => i = 2
        ↓
f(2):
n = 2;i = 2;
n < 5
n += i ==> n = 4
i++ => i = 3
        ↓
f(4):
n = 4;i = 3;
n < 5
n += i ==> n = 7
i++ => i = 4
        ↓
f(7):
n = 7;i = 4;
n > 5;
return n;//n = 7
*/
```

A: 5　　B: 6　　C: 7　　D: 8

4、下面3段程序代码的效果一样吗（ B ）

```
int b;
(1)const int *a = &b;
(2)int const *a = &b;
(3)int *const a = &b;

// 看 const 在 '*' 的左边还是右边判断
```

A: (2)=(3)　　B: (1)=(2)　　C: 都不一样　　D: 都一样

5、对于下面的说法，正确的是（ D ）

A: 对于 struct X{short s;int i;char c;}，sizeof(X)等于sizeof(s) + sizeof(i) + sizeof(c)

B: 对于某个double变量 a，可以使用 a == 0.0 来判断其是否为零

C: 初始化方式 char a[14] = "Hello, world!"; 和char a[14]; a = "Hello, world!";的效果相同

D: 以上说法都不对

二、编程题

1、验证尼科彻斯定理，即：任何一个整数m的立方都可以写成m个连续奇数之和。例如：

```
1^3=1
2^3=3+5
3^3=7+9+11
4^3=13+15+17+19
```

输入一个正整数m（m≤100），将m的立方写成m个连续奇数之和的形式输出。

注意：本题含有多组输入数据。

输入描述：输入一个int整数

输出描述：输出分解后的string

OJ链接【牛客网题号：HJ76 尼科彻斯定理】【难度：简单】

```
示例：
    输入：6
    输出：31+33+35+37+39+41
```

```c
//验证尼科彻斯定理，即：任何一个整数m的立方都可以写成m个连续奇数之和。
//例如：
//1^3=1
//2^3=3+5
//3^3=7+9+11
//4^3=13+15+17+19
//输入一个正整数m（m≤100），将m的立方写成m个连续奇数之和的形式输出。
//本题含有多组输入数据。

#include <stdio.h>

int main()
{
    int m = 0;
    while (scanf("%d", &m) != EOF)
    {
        int m_num = m * m;
        if (m % 2 == 0)
        {
            m_num = m_num - 2 * (int)(m / 2) + 1;
            for (int i = 0; i < m; i++)
            {
                if (i == m - 1)
                    printf("%d", m_num);
                else
                    printf("%d+", m_num);
                m_num += 2;
            }
            printf("\n");
        }
        else
        {
            m_num = m_num - 2 * (int)(m / 2);
            for (int i = 0; i < m; i++)
```

```
            {
                if (i == m - 1)
                    printf("%d", m_num);
                else
                    printf("%d+", m_num);
                m_num += 2;
            }
            printf("\n");
        }
    }

    return 0;
}
```

2、等差数列 `2，5，8，11，14，...`。（从 `2` 开始的 `3` 为公差的等差数列），求等差数列前`n`项和。

注意：本题有多组输入

输入描述：输入一个正整数`n`。

输出描述：输出一个相加后的整数。

OJ链接【牛客网题号：`HJ100` 等差数列】【难度：简单】

```
示例：
    输入：2                    输入：275
    输出：7                    输出：113575
    说明：2+5=7                说明：2+5+...+821+824=113575
```

```
//等差数列 2，5，8，11，14。。。
//（从 2 开始的 3 为公差的等差数列）
//输出求等差数列前n项和
//本题有多组输入

#include <stdio.h>

int main()
{
    int n = 0;
    while(scanf("%d", &n) != EOF)
    {
        int sum = 0;
        int now_n = 2;
        for(int i = 0; i < n; i++)
        {
            sum += now_n;
            now_n += 3;
        }
        printf("%d\n", sum);
    }

    return 0;
}
```

day03

一、选择题

1、已知函数的原型是：`int fun(char b[10], int *a);`，设定义：`char c[10];int d;`，正确的调用语句是（ `A` ）

A:`fun(c,&d);`　　B:`fun(c,d);`　　C:`fun(&c,&d);`　　D:`fun(&c,d);`

2、请问下列表达式哪些会被编译器禁止【多选】（ `ABCD` ）

```c
int a = 248, b = 4;
int const *c = 21;
const int *d = &a;
int *const e = &b;
int const * const f = &a;
/* 解析：
const修饰的对象，无法被修改
判断const的位置(即const 在 '*' 左边还是右边)，进而判断const所修饰的对象
*/
```

A:`*c = 32;`　　B:`*d = 43`　　C:`e=&a`　　D:`f=0x321f`

3、以下程序的输出结果为（ `A` ）

```c
#include <stdio.h>
int i;
void prt()
{
    for (i = 5; i < 8; i++)
        printf("%c", '*');
    printf("\t");
}
int main()
{
    for (i = 5; i <= 8; i++)
        prt();
    return 0;
}

/* 解析：
代码中只有一个全局变量 i
main函数中,for循环从 i = 5开始第一次循环,调用 prt 函数
进入prt函数中,for循环从i = 5开始，循环三次 输出三个 '*'，i递增至 8
prt函数调用结束，再返回main函数for循环的第一次循环
此时 i = 8，在main函数的for循环中 i 再递增至 9
i > 8，main函数中循环条件不成立
所以只输出一次 '***'
选 A
*/
```

A:`***`　　B:`***　　***　　***　　***`　　C:`***　　***`　　D:`*　　*　　*`

4、下面代码段的输出是（ `D` ）

```c
int main()
```

```
{
    int a=3;
    printf("%d\n",(a+=a-=a*a));
    return 0;
}
/* 解析:
操作符优先级判断    *(乘号) 优先级 3     -= 和 += 优先级 14, 但从右向左计算
所以 计算顺序为
a * a
a -= a * a
a += a
结果是
a -= 9      a ==> -6
a += a      a ==> -12
选D
*/
```

A:`-6`　B:`12`　C:`0`　D:`-12`

5、下列不能实现死循环的是（ `D` ）

A:`while(1){}`　B:`for(;1;){}`　C:`do{}while(1);`　D:`for(;0;){}`

二、编程题

1、首先输入要输入的整数个数`n`，然后输入`n`个整数。输出为`n`个整数中负数的个数，和所有正整数的平均值，结果保留一位小数。

注意：`0`即不是正整数，也不是负数，不计入计算；本题有多组输入用例。

输入描述：首先输入一个正整数`n`，然后输入`n`个整数。

输出描述：输出负数的个数，和所有正整数的平均值。

OJ链接【牛客网题号：`HJ97 记负均正`】【难度：简单】

```
示例：
    输入：  5
            1 2 3 4 5
            10
            1 2 3 4 5 6 7 8 9 0

    输出：  0 3.0
            0 5.0
```

```
//首先输入要输入的整数个数n，然后输入n个整数。输出为n个整数中负数的个数，和所有正整数的平均值，结果保留一位小数
//0即不是正整数，也不是负数，不计入计算
//本题有多组输入用例

#include <stdio.h>

int main()
{
    int n = 0;
    while(scanf("%d", &n) != EOF)
```

```
    {
        int arr[2002] = { 0 };
        int pos_sum = 0;
        int count_neg = 0;
        int count_pos = 0;
        for(int i = 0; i < n; i++)
        {
            scanf("%d", &arr[i]);
            if (arr[i] < 0)
                count_neg++;
            else if (arr[i] > 0)
            {
                count_pos++;
                pos_sum += arr[i];
            }
        }
        printf("%d %.1lf\n", count_neg, (double)pos_sum / count_pos);
    }

    return 0;
}
```

2、有一个长度为 `n` 的非降序数组，比如 `[1,2,3,4,5]`，将它进行旋转，即把一个数组最开始的若干个元素搬到数组的末尾，变成一个旋转数组，比如变成了 `[3,4,5,1,2]`，或者 `[4,5,1,2,3]` 这样的。请问，给定这样一个旋转数组，求数组中的最小值。

数据范围：`1≤n≤10000`，数组中任意元素的值：`0≤val≤10000`

OJ链接【牛客网题号：`JZ11` 旋转数组的最小数字】【难度：简单】

```
示例：
    输入：[3,4,5,1,2]
    返回值：1
```

```
//有一个长度为 n 的非降序数组，比如[1,2,3,4,5]，将它进行旋转，即把一个数组最开始的若干个元素搬到数组的末尾，变成一个旋转数组，比如变成了[3,4,5,1,2]，或者[4,5,1,2,3]这样的。请问，给定这样一个旋转数组，求数组中的最小值。

int minNumberInRotateArray(int* rotateArray, int rotateArrayLen ) {
    int left = 0;
    int right = rotateArrayLen - 1;
    while (left < right)
    {
        int mid = (left + right) / 2;
        if (*(rotateArray + mid) < *(rotateArray + right))
            right = mid;
        else if (*(rotateArray + mid) > *(rotateArray + right))
            left = mid + 1;
        else
            right--;
    }

    return *(rotateArray + left);
}
```

day04

一、选择题

1、设变量已正确定义，以下不能统计出一行中输入字符个数（不包含回车符）的程序段是（ D ）

A: `n=0;while(ch=getchar()!='\n')n++;`　　B: `n=0;while(getchar()!='\n')n++;`

C: `for(n=0;getchar()!='\n';n++);`　　D: `n=0;for(ch=getchar();ch!='\n';n++);`

2、运行以下程序后，如果从键盘上输入`65 14<回车>`，则输出结果为（ B ）

```c
int main()
{
    int m, n;
    printf("Enter m,n;");
    scanf("%d%d", &m,&n);
    while (m!=n)          //1
    {
        while(m>n) m=m-n; //2
        while(n>m) n=n-m; //3
    }
    printf("m=%d\n",m);
    return 0;
}

/* 解析：
m = 64, n = 14
m > n: m = m - n == 64 - 14 = 50    m === 50    n === 14
m > n: m = m - n == 50 - 14 = 36    m === 36    n === 14
m > n: m = m - n == 36 - 14 = 22    m === 22    n === 14
m > n: m = m - n == 22 - 14 = 8     m === 8     n === 14
m < n: n = n - m == 14 - 8 = 6      m === 8     n === 6
m > n: m = m - n == 8 - 6 = 2       m === 2     n === 6
m < n: n = n - m == 6 - 2 = 4       m === 2     n === 4
m < n: n = n - m == 4 - 2 = 2       m === 2     n === 2
所以输出 m = 2
*/
```

A: `3`　　B: `2`　　C: `1`　　D: `0`

3、若运行以下程序时，从键盘输入`ADescriptor<回车>`，则下面程序的运行结果是（ D ）

```c
#include <stdio.h>
int main()
{
    char c;
    int v0 = 0, v1 = 0, v2 = 0;
    do
    {
        switch(c=getchar())
        {
            case'a':case'A':
            case'e':case'E':
            case'i':case'I':
```

```
            case'o':case'O':
            case'u':case'U':v1 += 1;
            default: v0+= 1;v2+=1;
        }
    }while(c!='\n');
    printf("v0=%d,v1=%d,v2=%d\n",v0,v1,v2);
    return 0;
}
/* 解析:
输入 'A' 'E' 'I' 'O' 'U' 'a' 'e' 'i' 'o' 'u'  v0, v1, v2 均加一
输入其他    v0, v2 加一
*/
```

A: `v0=7,v1=4,v2=7`　　B: `v0=8,v1=4,v2=8`　　C: `v0=11,v1=4,v2=11`　　D: `v0=12,v1=4,v2=12`

4、以下程序段的功能是（ `C` ）

```
int a[] = {4, 0, 2, 3, 1}, i, j, t;
for(i = 1;i < 5;i++)
{
    t = a[i];
    j = i - 1;
    while(j >= 0 && t < a[j])
    {
        a[j + 1] = a[j];
        --j;
    }
    a[j + 1] = t;
}
/* 分析:

*/
```

A: `对数组a进行插入排序（升序）`　　B: `对数组a进行插入排序（升序）`

C: `对数组a进行选择排序（升序）`　　D: `对数组a进行选择排序（降序）`

5、执行下面的程序段，语句3的执行次数为（ `C` ）

```
for(i = 0; i <= n-1; i++)   // (1)
    for(j = n; j > i; j--)  // (2)
        state;              // (3)
```

A: `n(n+2)/2`　　B: `(n-1)(n+2)/2`　　C: `n(n+1)/2`　　D: `(n-1)(n+2)`

二、编程题

1、集合 `s` 包含从 `1` 到 `n` 的整数。不幸的是，因为数据错误，导致集合里面某一个数字复制了成了集合里面的另外一个数字的值，导致集合 丢失了一个数字 并且 有一个数字重复 。

给定一个数组 `nums` 代表了集合 `S` 发生错误后的结果。

请你找出重复出现的整数，再找到丢失的整数，将它们以数组的形式返回。

OJ链接 【`leetcode`题号：645. 错误的集合】【难度：简单】

```
示例：
    输入：nums = [1,2,2,4]      输入：nums = [1,1]
    输出：[2,3]                 输出：[1,2]
```

```c
int returnNum[2] = { 0 };
void sort(int* nums, int numsSize)
{
    int i, j, t;
    for (i = 1; i < numsSize; i++)
    {
        t = nums[i];
        j = i - 1;
        while (j >= 0 && t < nums[j])
        {
            nums[j + 1] = nums[j];
            --j;
        }
        nums[j + 1] = t;
    }
}
int* findErrorNums(int* nums, int numsSize, int* returnSize)
{
    sort(nums, numsSize);
    int* first = nums;
    int count = 0;
    if (*nums == *(nums + 1))
        returnNum[0] = *nums;
    while (*nums != *(nums + 1))
    {
        nums++;
        if (*nums == *(nums + 1))
        {
            returnNum[0] = *nums;
            break;
        }
    }
    nums = first;
    while (count < numsSize - 1)
    {
        if (*(nums + 1) - *nums == 2)
        {
            returnNum[1] = *nums + 1;
            break;
        }
        nums++;
        count++;
    }
    nums = first;
    if (*nums != 1)
        returnNum[1] = 1;
    else if (*(nums + numsSize - 1) != numsSize)
        returnNum[1] = numsSize;
    if(returnNum[0] != 0 && returnNum[1] != 0)
        *returnSize = 2;

    return returnNum;
```

```
}
//执行用时：960 ms，在所有 C 提交中击败了5.06%的用户
//内存消耗：6.8 MB，在所有 C 提交中击败了96.50%的用户
//蚌埠住了
```

2、小明同学最近开发了一个网站，在用户注册账户的时候，需要设置账户的密码，为了加强账户的安全性，小明对密码强度有一定要求：

- 1. 密码只能由大写字母，小写字母，数字构成；
- 2. 密码不能以数字开头；
- 3. 密码中至少出现大写字母，小写字母和数字这三种字符类型中的两种；
- 4. 密码长度至少为`8`

现在小明受到了`n`个密码，他想请你写程序判断这些密码中哪些是合适的，哪些是不合法的。

输入描述：输入一个数`n`，接下来有`n(n≤100)`行，每行一个字符串，表示一个密码，输入保证字符串中只出现大写字母，小写字母和数字，字符串长度不超过`100`。

输出描述：输入`n`行，如果密码合法，输出`YES`，不合法输出`NO`

OJ链接【牛客网题号：`OR141` 密码检查】【难度：简单】

```
示例：
    输入：1
         CdKfIfsiBgohWsydFYlMVRrGUpMALbmygeXdNpTmWkfyiZIKPtiflcgppuR
    输出：YES
```

```
//小明同学最近开发了一个网站，在用户注册账户的时候，需要设置账户的密码，为了加强账户的安全性，小明对密码强度有一定要求：
//1. 密码只能由大写字母，小写字母，数字构成；
//2. 密码不能以数字开头；
//3. 密码中至少出现大写字母，小写字母和数字这三种字符类型中的两种；
//4. 密码长度至少为8
//现在小明受到了n个密码，他想请你写程序判断这些密码中哪些是合适的，哪些是不合法的。

#include <stdio.h>
#include <string.h>
int main()
{
    int n = 0;
    scanf("%d", &n);
    for(int i = 0; i < n; i++)
    {
        int password_count = 0;
        int lower_count = 0;
        int num_count = 0;
        int upper_count = 0;
        int other_count = 0;
        char password[200] = { 0 };
        char* ri = password;
        scanf("%s", password);
        int password_len = strlen(password);
        while(*ri != '\0')
        {
            password_count++;
```

```
            if(*ri <= 'Z' && *ri >= 'A')
                upper_count++;
            else if(*ri <= 'z' && *ri >= 'a')
                lower_count++;
            else if(*ri <= '9' && *ri>= '0')
                num_count++;
            else
                other_count++;
            ri++;
        }
        if(other_count > 0 || password_count < 8 || (*password <= '9' &&
*password >= '0'))
            printf("NO\n");
        else if((upper_count > 0 && lower_count > 0) || (upper_count > 0 &&
num_count > 0) || (lower_count > 0 && num_count > 0))
            printf("YES\n");

    }

    return 0;
}
```

day05

一、选择题

1、如下程序的功能是（ D ）

```
#include <stdio.h>
int main()
{
    char ch[80] = "123abcdEFG*&";
    int j;
    puts(ch);
    for(j = 0; ch[j] != '\0'; j++)
        if(ch[j] >= 'A' && ch[j] <= 'Z')
            ch[j] = ch[j] + 'e' - 'E';
    puts(ch);
    return 0;
}

/* 解析：
'e' - 'E' === 32
大写字母 + 32 === 小写字母
*/
```

A: 测字符数组ch的长度

B: 将数字字符串ch转换成十进制数

C: 将字符数组ch中的小写字母转换成大写

D: 将字符数组ch中的大写字母转换成小写

2、对于代码段，下面描述正确的是（ B ）

```c
t=0;
while(printf("*"))
{
    t++;
    if (t<3)
        break;
}
```

A: 其中循环控制表达式与0等价　　B: 其中循环控制表达式与'0'等价

C: 其中循环控制表达式是不合法的　　D: 以上说法都不对

3、以下程序运行时，若输入 1abcedf2df<回车> 输出结果是（ C ）

```c
#include <stdio.h>
int main()
{
    char ch;
    while ((ch = getchar()) != '\n')
    {
        if (ch % 2 != 0 && (ch >= 'a' && ch <= 'z'))
            ch = ch - 'a' + 'A';
        putchar(ch);
    }
    printf("\n");
    return 0;
}

/* 解析：
ch是小写字母，并且字符ACSII码 必须是奇数才能转换为大写
'a' 'c' 'e' 'g' 'i' 'k' 'm' 'o' 'q' 's' 'u' 'w' 'y'
1abcedf2df
1AbCEdf2df
*/
```

A: 1abcedf2df　　B: 1ABCEDF2DF　　C: 1AbCEdf2df　　D: 1aBceDF2DF

4、下列条件语句中，功能与其他语句不同的是（ D ）

A: if(a) printf("%d\n",x); else printf("%d\n",y);

B: if(a==0) printf("%d\n",y); else printf("%d\n",x);

C: if (a!=0) printf("%d\n",x); else printf("%d\n",y);

D: if(a==0) printf("%d\n",x); else printf("%d\n",y);

5、我们知道C语言的 break 语句只能跳出离它最近的一层循环，可是有时候我们需要跳出多层循环，下列跳出多层循环的做法正确的是【多选】（ ABCD ）

A: 将程序写成函数用return结束函数，便可跳出循环

B: 修改外层循环条件例如

```
for( int i = 0 ; i < MAX1 ; i ++ )
{
    for( int j = 0 ; j < MAX2 ; j ++ )
    {
        if( condition )
        {
            i = MAX1;
            break;
        }
    }
}
```

C:在外层循环设置判断条件例如

```
for( ; symbol != 1 && condition2 ; )
{
    for( ; symbol != 1 && condition3 ; )
    {
        if( condition1 )
            symbol = 1 ;
    }
}
```

D:在外层循环后面加入break例如

```
for( ; condition2 ; )
{
    for( ; condition3 ; )
    {
        if( condition1 )
            symbol = 1 ;
    }
    if( symbol == 1 )
        break ;
}
```


二、编程题

1、给定一个长度为 n 的非降序数组和一个非负数整数 k ，要求统计 k 在数组中出现的次数

数据范围：0≤n≤1000,0≤k≤100，数组中每个元素的值满足 0≤val≤100

OJ链接【牛客网题号：JZ53 数字在升序数组中出现的次数】【难度：简单】

```
示例：
    输入：[1,2,3,3,3,3,4,5],3
    返回值：4
```

```
/**
 *
 * @param data int整型一维数组
 * @param dataLen int data数组长度
 * @param k int整型
```

```
 * @return int整型
 *
 * C语言声明定义全局变量请加上static，防止重复定义
 */
static int k_count = 0;
int GetNumberOfK(int* data, int dataLen, int k) {
    int left = 0;
    int right = dataLen - 1;
    int len = 0;
    if (k < *data || k > *(data + dataLen - 1) || data == NULL)
        return 0;
    else if (dataLen == 1 && *data == k)
        return 1;
    else if (*data == *(data + right))
        return dataLen;
    else
    {
        while (len != dataLen)
        {
            int mid = (left + right) / 2;
            if (k > *(data + mid))
                left = mid;
            else if (k < *(data + mid))
                right = mid - 1;
            else
            {
                k_count++;
                if (mid > 0)
                    left = mid - 1;
                if (mid < dataLen)
                    right = mid + 1;
                while (k == *(data + left))
                {
                    k_count++;
                    if (left == 0)
                        break;
                    if (left != 0)
                        left--;
                }
                while (k == *(data + right))
                {
                    k_count++;
                    if (right == dataLen)
                        break;
                    if (right != dataLen)
                        right++;
                }
                if (k_count != 0)
                    return k_count;
            }
            len++;
        }
        return k_count;
    }
}
```

2、整数转换。编写一个函数，确定需要改变几个位才能将整数`A`转成整数`B`。

OJ链接【`leetcode`题号：面试题 05.06. 整数转换】【难度：简单】

```
示例：
    输入：A = 29 （或者0b11101），B = 15（或者0b01111）          输入：A = 1，B = 2
    输出：2                                                    输出：2
```

```
int convertInteger(int A, int B){
    unsigned int tmp = A ^ B;
    int returnNum = 0;
    while (tmp != 0)
    {
        int ret = tmp & 1;
        if (ret == 1)
            returnNum++;
        tmp = tmp >> 1;
    }

    return returnNum;
}
```

day06

一、选择题

1、以下叙述中正确的是（ `A` ）

A:只能在循环体内和switch语句体内使用break语句

B:当break出现在循环体中的switch语句体内时，其作用是跳出该switch语句体，并中止循环体的执行

C:continue语句的作用是：在执行完本次循环体中剩余语句后，中止循环

D:在while语句和do-while语句中无法使用continue语句

2、下列`for`循环的次数为（ `A\D` ）

```
for(int i = 0 ; i || i++ < 5;);
// 括号外的';'算，就选A
// 括号外的';'不算，就选D
```

A:`0`　　B:`5`　　C:`1`　　D:无限

3、以下描述中正确的是（ `C` ）

A:由于do-while循环中循环体语句只能是一条可执行语句，所以循环体内不能使用复合语句

B:do-while循环由do开始，用while结束，在while(表达式)后面不能写分号

C:在do-while循环体中，不一定要有能使while后面表达式的值变为零("假")的操作

D:do-while循环中，根据情况可以省略while

4、设函数`fun`和实参数组的说明是如下形式，则对函数的调用语句中，正确的是（ `D` ）

```
void fun(char ch,float x[]);
float a[10];
```

A: `fun("asd" , a[]);`　　B: `fun('x' , A);`　　C: `fun('68' , 2.8);`　　D: `fun(32 , a);`

5、在c语言中，一个函数不写返回值类型，默认的返回类型是（ `A` ）

A: `int`　　B: `char`　　C: `void`　　D: `都不是`

二、编程题

1、给你一个整数数组 `nums` ，其中总是存在 唯一的 一个最大整数 。请你找出数组中的最大元素并检查它是否 至少是数组中每个其他数字的两倍 。如果是，则返回 最大元素的下标 ，否则返回 -1 。

OJ链接【`leetcode`题号：747. 至少是其他数字两倍的最大数】【难度：简单】

```
示例：
    输入：nums = [3,6,1,0]
    输出：1
    解释：6 是最大的整数，对于数组中的其他整数，6 大于数组中其他元素的两倍。6 的下标是 1 ，所以返回 1 。

    输入：nums = [1,2,3,4]
    输出：-1
    解释：4 没有超过 3 的两倍大，所以返回 -1 。

    输入：nums = [1]
    输出：0
    解释：因为不存在其他数字，所以认为现有数字 1 至少是其他数字的两倍。
```

```
int dominantIndex(int* nums, int numsSize) {
    if (numsSize == 1)
        return 0;
    int max = *nums;
    int max_2 = 0;
    int returnSign = 0;
    for (int i = 0; i < numsSize; i++)
    {
        if (*(nums + i) > max)
        {
            returnSign = i;
            max = *(nums + i);
        }
    }
    for (int i = 0; i < numsSize; i++)
    {
        if (*(nums + i) > max_2 && *(nums + i) < max)
            max_2 = *(nums + i);
    }
    if (max - 2 * max_2 >= 0)
        return returnSign;
    else
        return -1;
}
```

2、给定两个数组，编写一个函数来计算它们的交集。

OJ链接【leetcode题号：349. 两个数组的交集】【难度：简单】

```
示例：
    输入：nums1 = [1,2,2,1], nums2 = [2,2]
    输出：[2]

    输入：nums1 = [4,9,5], nums2 = [9,4,9,8,4]
    输出：[9,4]
```

```c
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int cmp(const void *a, const void *b)
 {
    return (*(int*)a - *(int*)b);
}
int* intersection(int* nums1, int nums1Size, int* nums2, int nums2Size, int* returnSize)
{
    if (!nums1 || !nums2)
    {
        *returnSize = 0;
        return NULL;
    }
    /* 快排对两个数组升序排序 (刚学的排序这不就用上了？doge)*/
    qsort(nums1, nums1Size, sizeof(nums1[0]), cmp);
    qsort(nums2, nums2Size, sizeof(nums2[0]), cmp);

    int* returnNum = (int *)malloc(sizeof(int) * (nums1Size + nums2Size));

    int tmp = 0;// 记录 重复数字的个数
    for (int i = 0, j = 0; i < nums1Size && j < nums2Size; )
    {
        if (*(nums1 + i) < *(nums2 + j))        //两数组元素一对一对比，哪个小 哪个位置递增
            i++;
        else if (*(nums1 + i) > *(nums2 + j))
            j++;
        else                                    //两数组元素相等就存入需要返回的数组中
        {
            *(returnNum + tmp) = *(nums1 + i);
            tmp++;
            i++;
            j++;
            //存入两个以上的时候，判断是否有重复存入第2个 就判断 returnNum[0] 和 returnNum[1]，即tmp - 1 和 tmp - 2
            if (tmp > 1 && *(returnNum+ tmp - 1) == *(returnNum+ tmp - 2))
                tmp--;
        }
    }

    *returnSize = tmp;
    return returnNum;
}
```

day07

一、选择题

1、以下对C语言函数的有关描述中，正确的有【多选】（ AB ）

A: 在C语言中，一个函数一般由两个部分组成，它们是函数首部和函数体

B: 函数的实参和形参可以是相同的名字

C: 在main()中定义的变量都可以在其它被调函数中直接使用

D: 在C程序中，函数调用不能出现在表达式语句中

2、在C语言中，以下正确的说法是（ A ）

A: 实参和与其对应的形参各占用独立的存储单元

B: 实参和与其对应的形参共占用一个存储单元

C: 只有当实参和与其对应的形参同名时才共占用存储单元

D: 形参是虚拟的，不占用存储单元

3、在上下文及头文件均正常的情况下，下列代码的输出是（ A ）（注：print已经声明过）

```c
int main()
{
    char str[] = "Geneius";
    print(str);
    return 0;
}
print(char *s)
{
    if(*s)
    {
        print(++s);
        printf("%c", *s);
    }
}

/* 解析：
print 函数内部，递归部分传参为 ++s
而且是先递归，后打印
直到 *s === '\0' 再返回
所以从 原字符串的最后一位开始打印
但是不打印'G'，是因为递归返回至最外一层的时候，已经 ++s了
已经不是 *str 了，而是 *(str+1) === 'e'
*/
```

A: suiene　　B: neius　　C: run-time error　　D: suieneG

4、对于函数 void f(int x);，下面调用正确的是（ B ）

A: int y=f(9);　　B: f(9);　　C: f(f(9));　　D: x=f();

5、给定fun函数如下，那么fun(10)的输出结果是（ C ）

```c
int fun(int x)
{
    return (x==1) ? 1 : (x + fun(x-1));
}
/* 解析：
x = 10: return ( 10 + fun (10 - 1) );
x = 9: return ( 9 + fun (9 - 1) );
x = 8: return ( 8 + fun (8 - 1) );
x = 7: return ( 7 + fun (7 - 1) );
x = 6: return ( 6 + fun (6 - 1) );
x = 5: return ( 5 + fun (5 - 1) );
x = 4: return ( 4 + fun (4 - 1) );
x = 3: return ( 3 + fun (3 - 1) );
x = 2: return ( 2 + fun (2 - 1) );
x = 1: return 1;
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 === 55
*/
```

A: `0`　　B: `10`　　C: `55`　　D: `3628800`

二、编程题

1、Lily上课时使用字母数字图片教小朋友们学习英语单词，每次都需要把这些图片按照大小（ASCII码值从小到大）排列收好。请大家给Lily帮忙，通过C语言解决。

输入描述：Lily使用的图片包括 `"A"` 到 `"Z"`、`"a"` 到 `"z"`、`"0"` 到 `"9"`。输入字母或数字个数不超过 `1024`。

输出描述：Lily的所有图片按照从小到大的顺序输出

OJ链接【牛客网题号：`HJ34` 图片整理】【难度：中等】

```
示例：
    输入：Ihave1nose2hands10fingers
    输出：0112Iaadeeefghhinnnorsssv
```

```c
//啥也别说了 直接快排

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

int cmp_char(const void* a, const void* b)
{
    return (*(char*)a - *(char*)b);
}

int main()
{
    char chs[1025] = { 0 };
    while (scanf("%s", &chs) != EOF)
    {
        int ch_len = strlen(chs);
        qsort(chs, ch_len, sizeof(chs[0]), cmp_char);
        printf("%s\n", chs);
    }
```

```
20
21      return 0;
22  }
```

2、给你一个整数数组 `nums` ，请计算数组的 中心下标 。

数组 中心下标 是数组的一个下标，其左侧所有元素相加的和等于右侧所有元素相加的和。

如果中心下标位于数组最左端，那么左侧数之和视为 0 ，因为在下标的左侧不存在元素。这一点对于中心下标位于数组最右端同样适用。

如果数组有多个中心下标，应该返回 最靠近左边 的那一个。如果数组不存在中心下标，返回 -1 。

OJ链接【`leetcode`题号：724. 寻找数组的中心下标】【难度：简单】

```
1   示例：
2       输入：nums = [1, 7, 3, 6, 5, 6]
3       输出：3
4       解释：
5           中心下标是 3 。
6           左侧数之和 sum = nums[0] + nums[1] + nums[2] = 1 + 7 + 3 = 11 ，
7           右侧数之和 sum = nums[4] + nums[5] = 5 + 6 = 11 ，二者相等。
8       输入：nums = [2, 1, -1]
9       输出：0
10      解释：
11          中心下标是 0 。
12          左侧数之和 sum = 0 ，（下标 0 左侧不存在元素），
13          右侧数之和 sum = nums[1] + nums[2] = 1 + -1 = 0 。
```

```
1   int pivotIndex(int* nums, int numsSize)
2   {
3       for(int i = 0; i < numsSize; i++)
4       {//从中心点为 0 开始，计算左右两边和
5           int sum_left = 0;
6           int sum_right = 0;
7           for(int j = 0; j < numsSize; j++)
8           {//i 每递增一次  左边多一个元素，右边少一个元素
9               if (j < i)//小于i  的都是左元素
10                  sum_left += *(nums + j);
11              else if (j > i)//大于i  的都是右元素
12                  sum_right += *(nums + j);
13          }
14          if(sum_right == sum_left)
15           return i;
16      }
17
18      return -1;
19  }
```

day08

一、选择题

1、如下程序的运行结果是（ D ）

```
char c[5]={'a', 'b', '\0', 'c', '\0'};
printf("%s", c);
```

A: 'a' 'b'　　B: ab\0c\0　　C: ab c　　D: ab

2、若有定义：int a[2][3];，以下选项中对a数组元素正确引用的是（ D ）

A: a[2][0]　　B: a[2][3]　　C: a[0][3]　　D: a[1>2][1]

> 对 a[2][3] 其元素引用成立的有：
>
> a[0][0] a[0][1] a[0][2] a[1][0] a[1][1] a[1][2]
>
> a[1>2][1] 中：　1 > 2 为假 等价于 0，所以 a[1>2][1] === a[0][1]

3、在下面的字符数组定义中，哪一个有语法错误（ D ）

A: char a[20]="abcdefg";　　B: char a[]="x+y=5.";　　C: char a[15];　　D: char a[10]='5';

> D. 字符串初始化用 ""

4、下列定义数组的语句中正确的是【多选】（ AB ）

A:

```
#define  size  10
char str1[size], str2[size+2];
```

B: char str[];　　C: int num['10'];　　D: int n=5; int a[n][n+2];

5、已知i，j都是整型变量，下列表达式中，与下标引用x[i][j]不等效的是【多选】（ BC ）

A: *(x[i]+j)　　B: *(x+i)[j]　　C: *(x+i+j)　　D: *(*(x+i)+j)

> x[i][j] === *(*(x+i)+j)

二、编程题

1、编写一个函数，计算字符串中含有的不同字符的个数。字符在 ASCII 码范围内(0~127 ，包括 0 和 127)，换行表示结束符，不算在字符里。不在范围内的不作统计。多个相同的字符只计算一次

例如，对于字符串 abaca 而言，有 a、b、c 三种不同的字符，因此输出 3 。

数据范围： 0 <= n <= 500

输入描述：输入一行没有空格的字符串。

输出描述：输出 输入字符串 中范围在(0~127，包括0和127)字符的种数。

OJ链接【牛客网题号： HJ10 字符个数统计】【难度：简单】

```
示例：
    输入：abc                    输入：aaa
    输出：3                      输出：1
```

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

int main()
{
    char chs[501] = {0};
    gets(chs);
    int Len = strlen(chs);
    char returnChs[129] = { 0 };
    int count = 0;
    for(int i = 0; i < Len; i++)
    {
        int j = (int)chs[i];
        if(returnChs[j] == 0)
        {
            returnChs[j] = j;
            count++;
        }
    }
    printf("%d", count);

    return 0;
}
```

2、给定一个大小为 n 的数组，找到其中的多数元素。多数元素是指在数组中出现次数 大于 ⌊ n/2 ⌋ 的元素。

你可以假设数组是非空的，并且给定的数组总是存在多数元素。

OJ链接【leetcode题号：169. 多数元素】【难度：简单】

```
示例：
    输入：[3,2,3]
    输出：3

    输入：[2,2,1,1,1,2,2]
    输出：2
```

```c
int cmp_int(const void* e1, const void* e2)
{
    return (*(int*)e1 - *(int*)e2);
}

int majorityElement(int* nums, int numsSize)
{
    qsort(nums, numsSize,sizeof(nums[0]),cmp_int);

    return *(nums + numsSize/2);
}
```

day09

一、选择题

1、下列程序的输出是（ D ）

```c
#include<stdio.h>
int  main()
{
    int a [12]= {1,2,3,4,5,6,7,8,9,10,11,12}, *p[4],i;
    for(i=0;i<4;i++)
        p[i]=&a [i*3];
    printf("%d\n", p[3][2]);
    return 0;
}
/* 解析：
p：指针数组
p[0] === &a[0](1)
p[1] === &a[3](4)
p[2] === &a[6](7)
p[3] === &a[9](10)
p[3][2] === *(&a[9] + 2) === *(a+11) === a[11] === 12
*/
```

A: 上述程序有错误　　B: 6　　C: 8　　D: 12

2、二维数组X按行顺序存储，其中每个元素占1个存储单元。若 x[4][4] 的存储地址为 0xf8b82140，x[9][9] 的存储地址为 0xf8b8221c，则 x[7][7] 的存储地址为（ A ）

A: 0xf8b821c4　　B: 0xf8b821a6　　C: 0xf8b82198　　D: 0xf8b821c0

3、以下哪个选项可以正确描述 sizeof(double)（ A ）

A: 一个整型表达式　　B: 一个双精度型表达式　　C: 一个不合法的表达式　　D: 一种函数调用

> sizeof()，单目操作符，结果为整型，使用可看作整形表达式

4、下列代码运行后的结果是什么（ A ）

```c
int main()
{
    char a = 'a',b;
    printf("%c,", ++a);
    printf("%c\n", b = a++);
    return 0;
}
```

A: b,b　　B: b,c　　C: a,b　　D: a,c

5、以下逗号表达式的值为（ A ）

```
(x = 4 * 5 , x * 5) , x + 5;
/* 解析：
逗号表达式：从左向右依次计算，结果为最后一个表达式的结果
x = 20, 100, 25
*/
```

A:`25`　　B:`20`　　C:`100`　　D:`45`

二、编程题

1、自除数 是指可以被它包含的每一位数除尽的数。例如，`128` 是一个自除数，因为 `128 % 1 == 0`，`128 % 2 == 0`，`128 % 8 == 0`。还有，自除数不允许包含 `0` 。给定上边界和下边界数字，输出一个列表，列表的元素是边界（含边界）内所有的自除数。

OJ链接【`leetcode`题号：728. 自除数】【难度：简单】

```
示例：
    输入：上边界left = 1，下边界right = 22
    输出：  [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 22]
```

```
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* selfDividingNumbers(int left, int right, int* returnSize)
{
    int* returnNum = (int *)malloc(sizeof(int) * (right - left));
    (*returnSize) = 0;
    for(int i = left; i < right + 1; i++)
    {
        int flag = i;
        int sum = 0;
        int tmp;
        while(tmp = flag % 10)
        {
            sum += i % tmp;
            flag /= 10;
            if(sum == 0 && flag == 0)
            {
                returnNum[(*returnSize)++] = i;
            }
        }
    }

    return returnNum;
}
```


2、给你一个长度为 `n` 的整数数组 `nums`，其中 `n > 1`，返回输出数组 `output` ，其中 `output[i]` 等于 `nums` 中除 `nums[i]` 之外其余各元素的乘积。

提示：题目数据保证数组之中任意元素的全部前缀元素和后缀（甚至是整个数组）的乘积都在 32 位整数范围内。

OJ链接【`leetcode`题号：238. 除自身以外数组的乘积】【难度：中等】

```
示例：
    输入：[1,2,3,4]
    输出：[24,12,8,6]
```

```c
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* productExceptSelf(int* nums, int numsSize, int* returnSize)
{
    /*
     先从左往右，把 i 左边的乘积，放入目标数组
     然后再从右往左，把 i 右边的乘积，乘入目标数组的对应位置
     然后返回目标数组
     */
    int* output = (int *)malloc(sizeof(int) * numsSize);
    *returnSize = numsSize;
    int product_right = 1;
    int product_left = 1;
    for (int i = 0; i < numsSize; i++)
    {
        output[i] = product_left;
        product_left *= nums[i];
    }
    for (int i = numsSize - 1; i >= 0; i--)
    {
        output[i] *= product_right;
        product_right *= nums[i];
    }

    return output;
}
```


day10

一、选择题

1、求函数返回值，传入`-1`，则在64位机器上函数返回（ `C` ）

```c
int func(int x)
{
    int count = 0;
    while (x)
    {
        count++;
        x = x&(x - 1);//与运算
    }
    return count;
}
/* 解析：
x = x&(x - 1);
-1:
11111111 11111111 11111111 11111111(补码)
x = (-1)&(-2) === -2
-2:
```

```
11111111 11111111 11111111 11111110(补码)
x = (-2)&(-3) === -4
-4:
11111111 11111111 11111111 11111100(补码)

可见每一次循环，x 的二进制补码有一位 变为0
直到 00000000 00000000 00000000 00000000 === 0
int 32 位，每一位置零，循环 32 次
所以选 32
*/
```

A: 死循环　　B: 64　　C: 32　　D: 16

2、读代码选结果（ D ）

```
int count = 0;
int x = -1;
while(x)
{
    count++;
    x = x >> 1;
}
printf("%d",count);

/* 解析：
x === -1
-1:
11111111 11111111 11111111 11111111(补码)
右移操作符，有符号位左边补符号位，所以 补码一直是 11111111 11111111 11111111
11111111
即 x === -1 恒等
所以死循环
*/
```

A: 1　　B: 2　　C: 32　　D: 死循环，没结果

3、下述赋值语句错误的是（ C ）

A: a = (b = (c = 2 , d = 3))　　B: i++　　C: a/b = 2　　D: a = a < a + 1

> C. 表达式不能做左值

4、若有 int w=1, x=2, y=3, z=4; 则条件表达 w < x ? w : y < z ? y : z 的值是（ A ）

A: 1　　B: 2　　C: 3　　D: 4

> exp1 ? exp2 : exp3：三目操作符，exp1 为真，结果为 exp2；为假，结果为exp3
>
> 1 < 2 ? 1 : (3 < 4 ? 3 : 4)　　1 < 2为真，结果为 1

5、以下程序运行后的输出结果是（ A ）

```
int main()
{
    int a=1,b=2,m=0,n=0,k;
    k=(n=b<a)&&(m=a);
    printf("%d,%d\n",k,m);
    return 0;
}
```

```
/* 解析：
首先对于 (n = b < a) && (m = a)
左表达式中 = 优先级最低， 所以先算 b < a
明显 b < a 为假，即 n = 0
&& 逻辑操作符，左边为真，右边才计算
(n = b < a) && (m = a)  左边为假，所以 m = a 不计算
所以 k === 0, m === 0
即输出 0,0
*/
```

A:`0,0`　　B:`0,1`　　C:`1,0`　　D:`1,1`

二、编程题

1、写一个函数，求两个整数之和，要求在函数体内不得使用+、-、*、/四则运算符号。

数据范围：两个数都满足 `0≤n≤1000`

OJ链接【牛客网题号：`JZ65` 不用加减乘除做加法】【难度：简单】

```
示例：
    输入：1,2
    返回值：3
```

```
int Add(int num1, int num2 ) {
//循环：一个自减 1，另一个自增 1



}
```


2、给你一个含 `n` 个整数的数组 `nums` ，其中 `nums[i]` 在区间 `[1, n]` 内。请你找出所有在 `[1, n]` 范围内但没有出现在 `nums` 中的数字，并以数组的形式返回结果。

OJ链接【`leetcode`题号：448. 找到所有数组中消失的数字】【难度：简单】

```
示例：
    输入：nums = [4,3,2,7,8,2,3,1]
    输出：[5,6]

    输入：nums = [1,1]
    输出：[2]
```

```
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* findDisappearedNumbers(int* nums, int numsSize, int* returnSize){
    *returnSize = 0;
    int num = *returnSize;
    int* returnNums = (int*)malloc(sizeof(int) * numsSize);
    for(int i = 1; i <= numsSize; i++)
    {
```

```
        int count_A = 0;
        int count_B = 0;
        for(int j = 0; j < numsSize; j++)
        {
            if(i == *(nums+j))
            {
                count_A++;
                break;
            }
            else if(j == numsSize - 1)
                count_B = i;
        }
        if(count_A == 0)
        {
            returnNums[num++] = count_B;
            *returnSize ++;
        }
    }

    return returnNums;
}
```

day11

一、选择题

1、声明以下变量，则表达式: `ch/i + (f*d - i)` 的结果类型为（ `D` ）

```
char ch;
int i;
float f;
double d;
/* 解析：
不同类型的运算，每次运算会将结果转换为 其中的上级类型
*/
```

A: `char`　　B: `int`　　C: `float`　　D: `double`

2、关于代码的说法正确的是（ `A` ）

```
#include <stdio.h>
int main()
{
    int x = -1;
    unsigned int y = 2;
    if (x > y)
    {
        printf("x is greater");
    }
    else
    {
        printf("y is greater");
    }
    return 0;
```

```
}
/* 解析：
有符号数和无符号数比较大小
有符号数要先转换为无符号数
然后再比较大小
*/
```

A: `x is greater` B: `y is greater` C: `依赖实现` D: `随机`

3、已知有如下各变量的类型说明，则以下不符合C语言语法的表达式是（ `A` ）

```
int k, a, b;
unsigned int w = 5;
double x = 1.42;
/* 解析：
求余操作符：
操作符两边必须均为 整型
*/
```

A: `x%3` B: `w+=-20` C: `k=(a=200,b=300)` D: `a+=a-=a=9`

4、下面函数的输出结果是（ `C` ）

```
void func()
{
    int k = 1 ^ (1 << 31 >> 31);
    printf("%d\n", k);
}
/* 解析：
<< : 左移操作符，右边末尾补 0
>> : 右移操作符，前边补符号位
1 << 31
00000000 00000000 00000000 00000001
10000000 00000000 00000000 00000000 >> 31
11111111 11111111 11111111 11111111 (补码)
11111111 11111111 11111111 11111110
10000000 00000000 00000000 00000001 (原码) => -1
1 ^ -1
00000000 00000000 00000000 00000001
11111111 11111111 11111111 11111111
(相同为 0 相异为 1)
11111111 11111111 11111111 11111110 (补码)
11111111 11111111 11111111 11111101
10000000 00000000 00000000 00000010 (原码) => -2
*/
```

A: `0` B: `-1` C: `-2` D: `1`

5、如下代码的输出结果是（ `A` ）

```c
#include <stdio.h>
int main()
{
    int i = 1;
    sizeof(i++);
    printf("%d\n", i);
    return 0;
}
/* 解析：
sizeof() ： 操作符内部表达式，不真正运算
sizeof(); 此操作符语句，在编译过程中就已经执行，并不是在项目运行过程中执行的，所以不真正运算
*/
```

A: 1　　B: 4　　C: 2　　D: 8

二、编程题

1、给定一个二进制数组， 计算其中最大连续 1 的个数。

OJ链接【leetcode题号：485. 最大连续 1 的个数】【难度：简单】

```
示例：
    输入：[1,1,0,1,1,1]
    输出：3
    解释：开头的两位和最后的三位都是连续 1 ，所以最大连续 1 的个数是 3.
```

```c
int findMaxConsecutiveOnes(int* nums, int numsSize)
{
    int count1 = 0;
    int count_max = 0;
    for(int i = 0; i < numsSize; i++)
    {
        if(nums[i])
            count1++;
        else
        {
            if(count1 > count_max)
                count_max = count1;
            count1 = 0;
        }
    }
    if(count1 > count_max)
       count_max = count1;

    return count_max;
}
```

2、求输出n以内(含n)完全数的个数。完全数（Perfect number），又称完美数或完备数，是一些特殊的自然数。它所有的真因子（即除了自身以外的约数）的和（即因子函数），恰好等于它本身。

例如：28，它有约数1、2、4、7、14、28，除去它本身28外，其余5个数相加，1+2+4+7+14=28。

注意：本题输入含有多组样例。

输入描述：输入一个数字n

输出描述：输出不超过n的完全数的个数

OJ链接【牛客网题号：HJ56 完全数计算】【难度：简单】

```
示例：
    输入：1000 7 100
    输出：3 1 2
```

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

int main()
{
    int n = 0;
    while (scanf("%d", &n) != EOF)
    {
        int count = 0;
        for (int i = 2; i <= n; i++)    // 不算 1
        {
            int sum = 0;
            for (int j = 1; j <= sqrt(i); j++)
            {
                if (i % j == 0)
                    sum += (j + i / j);
            }
            if (sum - i == i)
                count++;
        }
        printf("%d\n", count);
    }

    return 0;
}
```


day12

一、选择题

1、请阅读以下程序，其运行结果是（ D ）

```c
int main()
{
    char c='A';
    if('0'<=c<='9') printf("YES");
    else printf("NO");
    return 0;
}
/* 解析：
if条件 判断大小无法连续判断
if('0' <= c && c <= '9')
*/
```

A: `YES`　　B: `NO`　　C: `YESNO`　　D: `语句错误`

2、假设编译器规定`int`和`short`类型长度分别为32位和16位，若有下列C语言语句，则`y`的机器数为（ `B` ）

```c
unsigned short x = 65530;
unsigned int y = x;
/* 解析：
65530   十六进制    FFFA
*/
```

A: `0000 7FFA`　　B: `0000 FFFA`　　C: `FFFF 7FFA`　　D: `FFFF FFFA`

3、下列程序的输出结果是什么（ `B` ）

```c
#include<stdio.h>
int main()
{
    int n = 1001;
    int ans = 0;
    for(int i = 1; i <= n; ++i)
    {
        ans ^= i % 3;
    }
    printf("%d",ans);
    return 0;
}
/* 解析：
'^': 按位异或-相同为 0，相异为 1
i % 3 结果只会有 1 2 0 循环
发现
000 001 === 1
001 010 === 3
011 000 === 3
011 001 === 2
010 010 === 0
000 000 === 0
经过 两轮从 1 ~ 0 后
ans 重新置 0
1001 % 6 = 5
即 最后一次 为：
010 010 === 0
*/
```

A:`-2` B:`0` C:`1` D:`2`

4、C 语言中，下列运算符优先级最高的是（ `A` ）

A:`!` B:`%` C:`>>` D:`==`

5、要使`a`的低四位翻转，需要进行操作是（ `C` ）

A:`a|0xF` B:`a&0xF` C:`a^0xF` D:`~a`

```
/* 解析：
0xF 是 00000000 00000000 00000000 00001111
与之异或
即可翻转低四位
*/
```

二、编程题

1、输入一个整数，将这个整数以字符串的形式逆序输出，程序不考虑负数的情况，若数字含有0，则逆序形式也含有0，如输入为100，则输出为001。

数据范围： `1 <= n <= 2^30 - 1`

输入描述：输入一个int整数。

输出描述：将这个整数以字符串的形式逆序输出。

OJ链接【牛客网题号：`HJ11` 数字颠倒】【难度：简单】

```
示例：
    输入：1516000
    输出：0006151
```

```c
#include <stdio.h>

int main()
{
    int n = 0;
    scanf("%d", &n);
    if(n == 0)
    {
        printf("0");
        return 0;
    }
    while(n)
    {
        printf("%d", n % 10);
        n /= 10;
    }

    return 0;
}
```

2、对字符串中的所有单词进行倒排。

说明：

- 1、构成单词的字符只有26个大写或小写英文字母；
- 2、非构成单词的字符均视为单词间隔符；
- 3、要求倒排后的单词间隔符以一个空格表示；如果原字符串中相邻单词间有多个间隔符时，倒排转换后也只允许出现一个空格间隔符；
- 4、每个单词最长20个字母；

OJ链接【牛客网题号：HJ31 单词倒排】【难度：简单】

```
示例：
    输入：I am a student            输入：$bo*y gi!r#l
    输出：student a am I            输出：l r gi y bo
```

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

void reverse_char(char* arr_left, char* arr_right)
{
    while (arr_left < arr_right)
    {
        char tmp = *arr_left;
        *arr_left = *arr_right;
        *arr_right = tmp;
        arr_left++;
        arr_right--;
    }
}

int main()
{
    char arr[10001] = { 0 };
    gets(arr);
    size_t arr_len = strlen(arr);
    reverse_char(arr, arr + arr_len - 1);

    char* arr_left = arr;
    while (*arr_left)
    {
        char* char_end = arr_left;
        while ((*char_end <= 'z' && *char_end >= 'a') || (*char_end <= 'Z' && *char_end >= 'A'))
            char_end++;     //次循环记录一个单词的最后一个字母的后一位地址
        reverse_char(arr_left, char_end - 1);
        if (*char_end)
        {
            arr_left = char_end + 1;
            *char_end = ' ';
        }
        else
            arr_left = char_end;
    }

    printf("%s", arr);

    return 0;
}
```

```
// ps：牛客网可以通过，但是这段代码其实没办法解决  两个单词之间如果有多个分隔符，只输出一个空格  的问题

// 再ps：答案的写法应该并没有是原字符串单词逆序
```

day13

一、选择题

1、如果`x=2014`，下面函数的返回值是（ `C` ）

```
int fun(unsigned int x)
{
     int n = 0;
     while(x + 1)
     {
         n++;
         x = x | (x + 1);
     }
     return n;
}
/* 解析：
'^' 按位或- 有 1 为 1
2014    00000000 00000000 00000111 11011110
每次循环 x 的二进制位 有一个 0 变为 1
直到 x == 11111111 11111111 11111111 11111111 共 23 次循环
n === 23
此时 无符号整数 x 再 +1 === 0
结束循环
*/
```

A:`20` B:`21` C:`23` D 25

2、下列语句定义`x`为指向`int`类型变量`a`的指针，其中哪一个是正确的（ `A` ）

A:`int a , *x = a;` B:`int a , *x = &a;` C:`int *x = &a , a;` D:`int a , x = a;`

3、下面有关空指针和未初始化指针，说法错误的是（ `D` ）

A:`对0x0这个地址取值是非法的`

B:`空指针可以确保不指向任何对象或函数；而未初始化指针则可能指向任何地方`

C:`空指针与任何对象或函数的指针值都不相等`

D:`malloc在其内存分配失败时返回的是一个未初始化的指针`

4、若有定义`int a[8];`，则以下表达式中不能代表数组元素`a[1]`的地址的是（ `C` ）

A:`&a[0]+1` B:`&a[1]` C:`&a[0]++` D:`a+1`

5、以下选项中，对基本类型相同的两个指针变量不能进行运算的运算符是（ `A` ）

A:`+` B:`-` C:`=` D:`==`

二、编程题

1、有一只兔子，从出生后第3个月起每个月都生一只兔子，小兔子长到第三个月后每个月又生一只兔子，假如兔子都不死，问第`n`个月的兔子总数为多少?

注意：本题有多组数据。

数据范围：每组输入满足 `1 <= n <= 31`

输入描述：多行输入，一行输入一个int型整数表示第n个月

输出描述：每一行输出对应的兔子总数

OJ链接【牛客网题号：`HJ37` 统计每个月兔子的总数】【难度：简单】

```
示例：
    输入：1 2 3 4 5 6 9
    输出：1 1 2 3 5 8 34
```

```c
#include <stdio.h>
// 推断出结果是斐波那契数列
int main()
{
    int n = 0;
    int rubbit[32] = {
0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,1
7711,28657,46368,75025,121393,196418,317811,514229,832040,1346269 };
    while (scanf("%d", &n) != EOF)
    {
        printf("%d\n", rubbit[n]);
    }

    return 0;
}
```

2、数列的定义如下：数列的第一项为n，以后各项为前一项的平方根，求数列的前m项的和。

输入描述：

输入数据有多组，每组占一行，由两个整数`n(n<10000)`和`m(m<1000)`组成，n和m的含义如前所述。

输出描述：

对于每组输入数据，输出该数列的和，每个测试实例占一行，要求精度保留2位小数。

OJ链接【牛客网题号：`ZJ16` 数列的和】【难度：简单】

```
示例：
    输入：81 4
         2  2
    输出：94.73
         3.41
```

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

int main()
```

```c
{
    int n = 0;
    int m = 0;
    while(scanf("%d%d", &n, &m) != EOF)
    {
        double sum = 0;
        double ret = (double)n;
        while(m--)
        {
            sum = sum + ret;
            ret = sqrt(ret);
        }
        printf("%.2lf\n", sum);
    }

    return 0;
}
```

day14

一、选择题

1、有以下函数，该函数的功能是（ C ）

```c
int fun(char *s)
{
    char *t = s;
    while(*t++);
    return(t-s);
}
```

A: 比较两个字符的大小　　B: 计算s所指字符串占用内存字节的个数

C: 计算s所指字符串的长度　　D: 将s所指字符串复制到字符串t中

2、若有“ `float a[3]={1.5,2.5,3.5},*pa = a;*(pa++) *= 3;` ”，则 `*pa` 的值是（ B ）

A: `1.5`　　B: `2.5`　　C: `3.5`　　D: `4.5`

3、以下程序运行后的输出结果是（ A ）

```c
#include <stdio.h>
int main()
{
    int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}, *p = a + 5, *q = NULL;
    *q = *(p+5);
    printf("%d %d\n", *p, *q);
    return 0;
}
/* 解析：
q 是空指针
空指针没有指向空间，不能对空指针解引用赋值
*/
```

A: 运行后报错　　B: 6 6　　C: 6 11　　D: 5 10

4、设有定义 char *p[]={"Shanghai","Beijing","Honkong"}; 则结果为 j 字符的表达式是（ B ）

A: *p[1] +3　　B: *(p[1] +3)　　C: *(p[3] +1)　　D: p[3][1]

5、以下叙述中正确的是（ B ）

A: 即使不进行强制类型转换，在进行指针赋值运算时，指针变量的基类型也可以不同

B: 如果企图通过一个空指针来访问一个存储单元，将会得到一个出错信息

C: 设变量p是一个指针变量，则语句p=0;是非法的，应该使用p=NULL;

D: 指针变量之间不能用关系运算符进行比较

二、编程题

1、珠玑妙算游戏（the game of master mind）的玩法如下。

计算机有4个槽，每个槽放一个球，颜色可能是红色（R）、黄色（Y）、绿色（G）或蓝色（B）。例如，计算机可能有RGGB 4种（槽1为红色，槽2、3为绿色，槽4为蓝色）。作为用户，你试图猜出颜色组合。打个比方，你可能会猜YRGB。要是猜对某个槽的颜色，则算一次“猜中”；要是只猜对颜色但槽位猜错了，则算一次“伪猜中”。注意，“猜中”不能算入“伪猜中”。

给定一种颜色组合solution和一个猜测guess，编写一个方法，返回猜中和伪猜中的次数answer，其中answer[0]为猜中的次数，answer[1]为伪猜中的次数。

OJ链接【leetcode题号：面试题 16.15. 珠玑妙算】【难度：简单】

```
示例：
    输入： solution="RGBY",guess="GGRR"
    输出： [1,1]
    解释： 猜中1次，伪猜中1次。
```

```c
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* masterMind(char* solution, char* guess, int* returnSize)
{
    int* answer = (int*)malloc(sizeof(int) * 2);
    *returnSize = 2;
    int Great = 0;
    int great = 0;
    for(int i = 0; i < 4; i++)
    {
        if(solution[i] == guess[i])
        {
            Great++;
            solution[i] = 0;
            guess[i] = 0;
        }
    }
    for(int i = 0; i < 4; i++)
    {
        for(int j = 0; j < 4; j++)
        {
```

```
            if(solution[i] == guess[j] && solution[i] != 0 && guess[j] != 0)
            {
                great++;
                solution[i] = 0;
                guess[j] = 0;
                break;
            }
        }
    }
    answer[0] = Great;
    answer[1] = great;

    return answer;
}
```

2、给出一个整型数组 `numbers` 和一个目标值 `target`，请在数组中找出两个加起来等于目标值的数的下标，返回的下标按升序排列。

OJ链接【牛客网题号：`NC61` 两数之和】【难度：简单】

```
注意：本题只需要找到第一组符合要求的数据下标即可。不需要返回多组
示例：
    输入：[3,2,4],6
    返回值：[2,3]
    说明：因为 2+4=6 ， 而 2的下标为2 ， 4的下标为3 ，又因为 下标2 < 下标3 ，所以输出[2,3]
```

```
int* twoSum(int* numbers, int numbersLen, int target, int* returnSize ) {
    *returnSize = 2;
    static int target_num[2] = {0};
    memset(target_num, 0, sizeof(target_num));
    for(int i = 0; i < numbersLen; i++)
    {
        for(int j = i + 1; j < numbersLen; j++)
        {
            if((numbers[i] + numbers[j]) == target)
            {
                target_num[0] = i+1;
                target_num[1] = j+1;
                return target_num;
            }
        }
    }

    *returnSize = 0;
    return NULL;
}
```

day15

一、选择题

1、有如下代码，则 `*(p[0]+1)` 所代表的数组元素是（ `C` ）

```
int a[3][2] = {1, 2, 3, 4, 5, 6}, *p[3];
p[0] = a[1];
/* 解析：
p[0] === a[1][0]
*(p[0] + 1) === a[1][1]
*/
```

A: `a[0][1]`　　B: `a[1][0]`　　C: `a[1][1]`　　D: `a[1][2]`

2、关于指针下列说法正确的是【多选】（ `ABD` ）

A: `任何指针都可以转化为void *`　　B: `void *可以转化为任何指针`

C: `指针的大小为8个字节`　　D: `指针虽然高效、灵活但可能不安全`

3、以下 `scanf` 函数调用选项中，错误的是（ `D` ）

```
struct T
{
    char name[20];
    int age;
    int sex;
} a[5], *pa=a;
```

A: `scanf("%s",a[0].name);`　　B: `scanf("%d", &pa[0].age);`

C: `scanf("%d",&(pa->age));`　　D: `scanf("%d", pa->age);`

4、如下函数 `fun` 计算 `prod=1*2*3*…*n`，并返回计算结果值。但当 `n>12` 时，返回值不正确。要找出该程序的错误，正确的调试方法是（ `A` ）

```
int fun(int n)
{
    int prod = 1 , i = 0;
    for(i = 1;i <= n;i++)
    {
        prod *= i;
    }
    return prod;
}
/* 解析：
首先，肯定需要监控 prod 的值
所以排除 C D
其次 需要逐步手动进行循环
所以选 A
*/
```

A: `监视变量prod的值，在prod *= i;行处设置断点，然后单步运行，直到发现错误原因`

B: `监视变量prod的值，在return prod;行处设置断点，程序中断后，即可发现错误原因`

C:在prod=1;处设置断点，然后在函数调用堆栈中即可发现错误原因

D:监视变量i的值，在for (i=1; i<=n; i++)行处设置断点，然后单步运行，直到发现错误原因

5、下列给定程序中，函数fun的功能是：把形参a所指数组中的奇数按原顺序依次存放到a[0]、a[1]、a[2]…中，把偶数从数组中删除，奇数个数通过函数值返回。 例如，若a所指数组中的数据最初排列为：9,1,4,2,3,6,5,8,7，删除偶数后，a所指数组中的数据为：9,1,3,5,7，返回值为5。请在程序的下画线处填入正确的内容并将下画线删除，使程序得出正确的结果（ D ）

```
int fun(int a[], int n)
{
    int i, j;
    j=0;
    for (i=0; i<n; i++)
    if (a[i]%2== _________ )
    {
        a[j]=a[i];
        _________;
    }
    return _________;
}
/* 解析：
每次需要判断是否为 奇数，然后将其按顺序放入
每次if内，j++
所以返回 j 即可
*/
```

A:0 j++ j　　B:1 j++ j+1　　C:0 j++ j+1　　D:1 j++ j

二、编程题

1、现在有一个长度为 n 的正整数序列，其中只有一种数值出现了奇数次，其他数值均出现偶数次，请你找出那个出现奇数次的数值。

输入描述：第一行：一个整数n，表示序列的长度。第二行：n个正整数ai，两个数中间以空格隔开。

输出描述：一个数，即在序列中唯一出现奇数次的数值。

OJ链接【牛客网题号：KS33 寻找奇数】【难度：简单】

```
示例：
    输入：5
         2 1 2 3 1
    输出：3
```

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

int cmp_int(const void* x, const void* y)
{
    return *(int*)x - *(int*)y;
}

int main()
{
```

```
    int n = 0;
    scanf("%d", &n);
    int* arr = (int*)malloc(sizeof(int) * n);
    for(int i = 0; i < n; i++)
    {
       scanf("%d", arr + i);
    }
    if(n == 1)
    {
        printf("%d\n", *arr);
        return 0;
    }
    else
    {
        qsort(arr, n, sizeof(arr[0]), cmp_int);
        for(int i = 0; i < n; i += 2)
        {
            if(arr[i] != arr[i + 1])
            {
                printf("%d\n", arr[i]);
                return 0;
            }
        }
    }
}
```

2、给定一个长度为n的数组`nums`，请你找到峰值并返回其索引。数组可能包含多个峰值，在这种情况下，返回任何一个所在位置即可。

1.峰值元素是指其值严格大于左右相邻值的元素。严格大于即不能有等于

2.假设 `nums[-1] = nums[n] = 负无穷小`

3.对于所有有效的 i 都有 `nums[i] != nums[i + 1]`

OJ链接【牛客网题号：`NC107 寻找峰值`】【难度：简单】

```
示例：
    输入：[2,4,1,2,7,8,4]
    返回值：1
    说明：4和8都是峰值元素，返回4的索引1或者8的索引5都可以

    输入：[5,3,4,2,6]
    返回值：0
    说明：-1作为下标或者n下标位置都表示负无穷小， 则0号下标5是峰值，或者4号下标6也是峰值
```

```
/**
 * 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可
 *
 *
 * @param nums int整型一维数组
 * @param numsLen int nums数组长度
 * @return int整型
 *
 * C语言声明定义全局变量请加上static，防止重复定义
 */
int findPeakElement(int* nums, int numsLen )
```

```
{
    for(int i = 0; i < numsLen; i++)
    {
        if(nums[0] > nums[1])
            return 0;
        else if(nums[numsLen - 1] > nums[numsLen - 2])
            return numsLen - 1;
        else
        {
            if(nums[i] > nums[i - 1] && nums[i] > nums[i + 1])
                return i;
        }
    }

    return 0;
}
```

day16

一、选择题

1、指出下列代码的缺陷【多选】 （ BC ）

```
float f[10];
// 假设这里有对f进行初始化的代码
for(int i = 0; i < 10;)
{
    if(f[++i] == 0)
        break;
}
```

A: `for(int i = 0; i < 10;)这一行写错了`　　B: `f是float型数据直接做相等判断有风险`

C: `f[++i]应该是f[i++]`　　D: `没有缺陷`

2、请指出以下程序的错误【多选】 （ AC ）

```
void GetMemory(char **p, int num)
{
    if(NULL == p && num <= 0)//1
        return;
    *p = (char*)malloc(num);
    return;
}
int main()
{
    char *str = NULL;
    GetMemory(&str, 80); //2
    if(NULL != str)
    {
        strcpy(&str, "hello"); //3
        printf(str); //4
    }
    return 0;
}
```

A: 1　　B: 2　　C: 3　　D: 4

3、请问下列代码的输出结果有可能是哪些【多选】（ ）

```c
#include <stdio.h>
typedef union
{
    int a;
    struct
    {
        short b;
        short c;
    };
}X;
int main()
{
    X x;
    x.a = 0x20150810;
    printf("%x,%x\n", x.b, x.c);
    return 0;
}
```

A: 2015,810　　B: 50810,201　　C: 810,2015　　D: `20150,810

4、下面这个程序执行后会有什么错误或者效果【多选】（ AB ）

```c
#define MAX 255
int main()
{
     unsigned char A[MAX], i;
     for(i = 0; i <= MAX; i++)
         A[i] = i;
    return 0;
}
```

A: 数组越界　　B: 死循环　　C: 栈溢出　　D: 内存泄露

5、请问下列程序的输出是多少（ B ）

```c
#include<stdio.h>
int main()
{
    unsigned char i = 7;
    int j = 0;
    for(;i > 0;i -= 3)
    {
       ++j;
    }
    printf("%d\n", j);
    return 0;
}
```

A: 2　　B: 死循环　　C: 173　　D: 172

二、编程题

1、牛牛以前在老师那里得到了一个正整数数对`(x, y)`，牛牛忘记他们具体是多少了。但是牛牛记得老师告诉过他`x`和`y`均不大于`n`，并且`x`除以`y`的余数大于等于`k`。牛牛希望你能帮他计算一共有多少个可能的数对。

输入描述：输入包括两个正整数`n,k(1 <= n <= 10^5, 0 <= k <= n - 1)`。

输出描述：对于每个测试用例，输出一个正整数表示可能的数对数量。

OJ链接【牛客网题号：`WY49`数对】【难度：简单】

```
示例：
    输入：5 2
    输出：7
    说明：满足条件的数对有(2,3),(2,4),(2,5),(3,4),(3,5),(4,5),(5,3)
```

```
#include <stdio.h>

int main()
{
    long n, k;
    while(~scanf("%ld %ld", &n, &k))
    {
        if (k == 0)
        {
            printf("%ld\n", n * n);
            continue;
        }
        long count = 0;
        for(long y = k + 1; y <= n; y++)
        {
            count += ((n / y) * (y - k)) + ((n % y < k) ? 0 : (n % y - k + 1));
        }
        printf("%ld\n", count);
    }

    return 0;
}
// 这个
// 数学不好
// 自己想是真想不出来
```


2、输入一个字符串和一个整数 k ， 截取字符串的前k个字符并输出

输入描述：

- 1.输入待截取的字符串
- 2.输入一个正整数k，代表截取的长度

输出描述：截取后的字符串

OJ链接【牛客网题号：`HJ46`截取字符串】【难度：简单】

```
示例：
    输入：abABCcDEF 6

    输出：abABCc
```

```c
#include <stdio.h>

int main()
{
    char str[101];
    while(scanf("%s", str) != EOF)
    {
        int n;
        scanf("%d", &n);
        str[n] = '\0';
        printf("%s\n", str);
    }

    return 0;
}
```

请乘理想之马
挥鞭从此起程
路上春色正好
天上太阳正晴

出题人：黄坤、张文超
审核人：张鹏伟
后期制作排版：王海斌、高博、贾小英