GESP C++四级认证试题

（满分：100 分　考试时间：90 分钟）

学校：_____　　姓名：_____

题目	一	二	三	总分
得分				

一、单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	D	C	C	D	A	D	D	B	C	C	A	A	B	D	A

1. 高级语言编写的程序需要经过以下（　）操作，可以生成在计算机上运行的可执行代码。

A. 编辑

B. 保存

C. 调试

D. 编译

2. 排序算法是稳定的（Stable Sorting），就是指排序算法可以保证，在待排序数据中有两个相等记录的关键字 R 和 S（R 出现在 S 之前），在排序后的列表中 R 也一定在 S 前。下面关于排序稳定性的描述，正确的是（　）。

A. 冒泡排序是不稳定的。

B. 插入排序是不稳定的。

C. 选择排序是不稳定的。

D. 以上都不正确。

3. 下列关于 C++语言中指针的叙述，不正确的是（　）。

A. 指针变量中存储的是内存地址。

B. 定义指针变量时必须指定其指向的类型。

C. 指针变量只能指向基本类型变量，不能指向指针变量。

D. 指针变量指向的内存地址不一定能够合法访问。

4. 下列关于 C++语言中数组的叙述，不正确的是（　）。

A. 一维数组在内存中一定是连续存放的。

B. 二维数组是一维数组的一维数组。

C. 二维数组中的每个一维数组在内存中都是连续存放的。

D. 二维数组在内存中可以不是连续存放的。

5. 下列关于 C++语言中函数的叙述，正确的是（　）。

A. 函数必须有名字。

B. 函数必须有参数。

C. 函数必须有返回值。

D. 函数定义必须写在函数调用前。

6. 下列关于 C++语言中变量的叙述，正确的是（　）。

A. 变量定义后可以一直使用。

B. 两个变量的变量名不能是相同的。

C. 两个变量的变量名可以相同，但它们的类型必须是不同的。

D. 两个变量的变量名可以相同，但它们的作用域必须是不同的。

7. 一个二维数组定义为 `double array[3][10];`，则这个二维数组占用内存的大小为（ ）。

A. 30

B. 60

C. 120

D. 240

8. 一个变量定义为 `int *p = nullptr;`，则下列说法正确的是（ ）。

A. 该指针变量的类型为 `int`。

B. 该指针变量指向的类型为 `int`。

C. 该指针变量指向的内存地址是随机的。

D. 访问该指针变量指向的内存会出现编译错误。

9. 一个二维数组定义为 `int array[5][3];`，则 `array[1][2]`和 `array[2][1]` 在内存中的位置相差多少字节？（ ）

A. 2 字节。

B. 4 字节。

C. 8 字节。

D. 无法确定。

10. 如果 `a` 为 `int` 类型的变量，且 `a` 的值为 `6`，则执行 `a &= 3;`之后，`a` 的值会是（ ）。

A. 3

B. 9

C. 2

D. 7

11. 一个数组定义为 `int a[5] = {1, 2, 3, 4, 5};`，一个指针定义为 `int * p = &a[2];`，则执行 `a[1] = *p;`后，数组 a 中的值会变为（ ）。

A. `{1, 3, 3, 4, 5}`

B. `{2, 2, 3, 4, 5}`

C. `{1, 2, 2, 4, 5}`

D. `{1, 2, 3, 4, 5}`

12. 以下哪个函数声明在调用时可以传递二维数组的名字作为参数？（ ）

A. `void BubbleSort(int a[][4]);`

B. `void BubbleSort(int a[3][]);`

C. `void BubbleSort(int a[][]);`

D. `void BubbleSort(int ** a);`

13. 在下列代码的横线处填写（ ），可以使得输出是“20 10”。

```
#include <iostream>
using namespace std;
void xchg(_____________) { // 在此处填入代码
    int t = *x;
    *x = *y;
    *y = t;
}
int main() {
    int a = 10, b = 20;
    xchg(&a, &b);
    cout << a << " " << b << endl;
    return 0;
}
```

A. `int x, int y`

B. `int * x, int * y`

C. `int a, int b`

D. `int & a, int & b`

14. 执行以下C++语言程序后，输出结果是（ ）。

```cpp
#include <iostream>
using namespace std;
int main() {
    int array[3][3];
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            array[i][j] = i * 10 + j;
    int sum;
    for (int i = 0; i < 3; i++)
        sum += array[i][i];
    cout << sum << endl;
    return 0;
}
```

A. 3

B. 30

C. 33

D. 无法确定。

15. 在下列代码的横线处填写（ ），完成对有 n 个 int 类型元素的数组 array 由小到大排序。

```cpp
void SelectionSort(int array[], int n) {
    int i, j, min, temp;
    for (i = 0; i < n - 1; i++) {
        min = i;
        for (j = i + 1; j < n; j++)
            if (_________________) // 在此处填入代码
                min = j;
        temp = array[min];
        array[min] = array[i];
        array[i] = temp;
    }
}
```

A. `array[min] > array[j]`

B. `array[min] > array[i]`

C. `min > array[j]`

D. `min > array[i]`

二、判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	×	√	×	×	√	×	×	×	√	√

1. 域名是由一串用点分隔的名字来标识互联网上一个计算机或计算机组的名称，CCF 编程能力等级认证官方网站的域名是 gesp.ccf.org.cn，其中顶级域名是 gesp。

2. 数列 1，1，2，3，5，8 ... 是以意大利数学家列昂纳多·斐波那契命名的数列，从第三个数开始，每个数是前面两项之和。如果计算该数列的第 n 项（其中 n>3）fib(n)，我们采用如下方法：① 令 fib(1)=fib(2)=1 ②用循环 for i=3 to n 分别计算 f(i) ③输出 fib(n)。这体现了递推的编程思想。

3. 在 C++语言中，函数的参数默认以引用传递方式进行传递。

4. 在 C++语言中，可以定义四维数组，但在解决实际问题时不可能用到，因为世界是三维的。

5. 在 C++语言中，一个函数没有被调用时，它的参数不占用内存。

6. 在 C++语言中，如果一个函数可能抛出异常，那么一定要在 `try` 子句里调用这个函数。

7. 如果希望记录 10 个最长为 99 字节的字符串，可以将字符串数组定义为 `chars[100][10];`。

8. 字符常量`'0'`和`'\0'`是等价的。

9. >=和>>=都是 C++语言的运算符。

10. 由于文件重定向操作，程序员在使用 C++语言编写程序时无法确定通过 cout 输出的内容是否会被输出到屏幕上。

三、编程题（每题 25 分，共 50 分）

题号	1	2
答案		

1. 幸运数

【问题描述】

小明发明了一种“幸运数”。一个正整数，其偶数位不变（个位为第 1 位，十位为第 2 位，以此类推），奇数位做如下变换：将数字乘以 7，如果不大于 9 则作为变换结果，否则把结果的各位数相加，如果结果不大于 9 则作为变换结果，否则（结果仍大于 9）继续把各位数相加，直到结果不大于 9，作为变换结果。变换结束后，把变换结果的各位数相加，如果得到的和是 8 的倍数，则称一开始的正整数为幸运数。

例如，16347：第 1 位为 7，乘以 7 结果为 49，大于 9，各位数相加为 13，仍大于 9，继续各位数相加，最后结果为 4；第 3 位为 3，变换结果为 3；第 5 位为 1，变换结果为 7。最后变化结果为 76344，对于结果 76344 其各位数之和为 24，是 8 的倍数。因此 16347 是幸运数。

【输入描述】

输入第一行为正整数N，表示有N个待判断的正整数。约定 $1 \leq N \leq 20$。

从第 2 行开始的N行，每行一个正整数，为待判断的正整数。约定这些正整数小于10^{12}。

【输出描述】

输出N行，对应N个正整数是否为幸运数，如是则输出'T'，否则输出'F'。

提示：不需要等到所有输入结束在依次输出，可以输入一个数就判断一个数并输出，再输入下一个数。

【样例输入 1】

```
2
16347
76344
```

【样例输出 1】

```
T
F
```

【参考程序】

```
#include <iostream>
using namespace std;

// 奇数位要做的数字变换
int trans(int t) {
    if (t == 0)
        return 0;
    return (t * 7 - 1) % 9 + 1;
}
// 判断是否为幸运数
bool judge(long long x) {
    int sum = 0;
    for (int d = 1; x > 0; d++, x /= 10) {
        int t = (int)(x % 10);
        if (d % 2 == 0)
            sum += t;
        else
            sum += trans(t);
    }
    return (sum % 8 == 0);
}
int main() {
    int N = 0;
    cin >> N;
    for (int n = 0; n < N; n++) {
        long long x = 0;
```

```
        cin >> x;
        if (judge(x))
            cout << "T" << endl;
        else
            cout << "F" << endl;
    }
    return 0;
}
```

2. 图像压缩

【问题描述】

图像是由很多的像素点组成的。如果用 0 表示黑，255 表示白，0 和 255 之间的值代表不同程度的灰色，则可以用一个字节表达一个像素（取值范围为十进制 0-255、十六进制 00-FF）。这样的像素组成的图像，称为 256 级灰阶的灰度图像。

现在希望将 256 级灰阶的灰度图像压缩为 16 级灰阶，即每个像素的取值范围为十进制 0-15、十六进制 0-F。压缩规则为：统计出每种灰阶的数量，取数量最多的前 16 种灰阶（如某种灰阶的数量与另外一种灰阶的数量相同，则以灰阶值从小到大为序），分别编号 0-F（最多的编号为 0，以此类推）。其他灰阶转换到最近的 16 种灰阶之一，将某个点灰阶数与 16 种灰阶种的一种相减，绝对值最小即为最近，如果绝对值相等，则编号较小的灰阶更近。

【输入描述】

输入第 1 行为一个正整数N，表示接下来有N行数据组成一副 256 级灰阶的灰度图像。约定 $10 \leq N \leq 20$。

第 2 行开始的N行，每行为长度相等且为偶数的字符串，每两个字符用十六进制表示一个像素。约定输入的灰度图像至少有 16 种灰阶。约定每行最多 20 个像素。

【输出描述】

第一行输出压缩选定的 16 种灰阶的十六进制编码，共计 32 个字符。

第二行开始的N行，输出压缩后的图像，每个像素一位十六进制数表示压缩后的灰阶值。

【样例输入 1】

```
10
00FFCFAB00FFAC09071B5CCFAB76
00AFCBAB11FFAB09981D34CFAF56
01BFCEAB00FFAC0907F25FCFBA65
10FBCBAB11FFAB09981DF4CFCA67
00FFCBFB00FFAC0907A25CCFFC76
00FFCBAB1CFFCB09FC1AC4CFCF67
01FCCBAB00FFAC0F071A54CFBA65
10EFCBAB11FFAB09981B34CFCF67
01FFCBAB00FFAC0F071054CFAC76
1000CBAB11FFAB0A981B84CFCF66
```

【样例输出 1】

```
ABCFFF00CB09AC07101198011B6776FC
321032657CD10E
36409205ACC16D
B41032657FD16D
8F409205ACF14D
324F326570D1FE
3240C245FC411D
BF4032687CD16D
8F409205ACC11D
B240326878D16E
83409205ACE11D
```

【样例解释 1】

灰阶‘AB’、‘CF’和‘FF’出现 14 次，‘00’出现 10 次，‘CB’出现 9 次，‘09’出现 7 次，‘AC’出现 6 次，‘07’出现 5 次，‘10’、‘11’和‘98’出现 4 次，‘01’、‘1B’、‘67’、‘76’和‘FC’出现 3 次。

【参考程序】

```
#include <iostream>
#include <cstring>
```

```cpp
using namespace std;

int image[20][20];
int cpimg[20][20];
int his[256];
int color[16];
// 一位十六进制字符转换为数字
int trans(char a) {
    if (a <= '9')
        return (a - '0');
    return (a - 'A' + 10);
}
// 一位十六进制数字转换为字符
char itrans(int n) {
    if (n >= 10)
        return (char)(n - 10 + 'A');
    return (char)(n + '0');
}
// 寻找离 c 最近的灰阶
int compress(int c) {
    int dis = 256, res = -1;
    for (int i = 0; i < 16; i++) {
        int d = c - color[i];
        if (d < 0)
            d = -d;
        if (d < dis) {
            dis = d;
            res = i;
        }
    }
    return res;
}
int main() {
    int N = 0, M = 0;
    cin >> N;
    // 灰阶计数，初始化为 0
    for (int i = 0; i < 256; i++)
        his[i] = 0;
    // 输入图像，并对灰阶计数
    for (int i = 0; i < N; i++) {
        char line[50];
        cin >> line;
        M = strlen(line) / 2;
```

```
        for (int j = 0; j < M; j++) {
            int c = trans(line[j * 2]) * 16 + trans(line[j * 2 + 1]);
            image[i][j] = c;
            his[c]++;
        }
    }
    // 选取出现次数最多的 16 个灰阶
    for (int c = 0; c < 16; c++) {
        int max = -1, max_id = -1;
        for (int i = 0; i < 256; i++)
            if (his[i] > max) {
                max = his[i];
                max_id = i;
            }
        color[c] = max_id;
        his[max_id] = -1;
    }
    // 将 image 的灰阶压缩为 cpimg
    for (int i = 0; i < N; i++)
        for (int j = 0; j < M; j++)
            cpimg[i][j] = compress(image[i][j]);
    // 输出选取的 16 个灰阶
    for (int c = 0; c < 16; c++)
        cout << itrans(color[c] / 16) << itrans(color[c] % 16);
    cout << endl;
    // 输出压缩后的图像
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++)
            cout << itrans(cpimg[i][j]);
        cout << endl;
    }
    return 0;
}
```