GESP C++六级样题

(满分：100 分　考试时间：180 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	**A**	**B**	**C**	**C**	**D**	**C**	**A**	**B**	**D**	**D**	**C**	**C**	**C**	**B**	**C**

1. 以下不属于计算机输出设备的有（　）。

A. 麦克风

B. 音箱

C. 打印机

D. 显示器

2. 小明想了一个 1 到 100 之间的整数。你可以做多次猜测，每次猜测之后，如果你没有猜中，小明会告诉你，你猜的数比他想的数大还是小。你希望你在运气最坏的情况下花费最少的次数猜中，请问你运气最坏的情况下会猜几次？（包括最后猜中的那次）

A. 6

B. 7

C. 8

D. 100

3. 关于分治算法，下列说法错误的是（　）。

A. 分治算法的核心思想是分而治之，即把问题转化为多个规模更小的子问题求解。

B. 分治算法可以不使用递归实现。

C. 分治算法的时间复杂度是$O(\log N)$，其中N表示问题的规模。

D. 二分法、快速排序等算法都是典型的使用分治思想的算法。

4. 下面关于 C++类的说法中，正确的是（　）。

A. 派生类不能和基类有同名成员函数，因为会产生歧义。

B. 派生类可以和基类有同名成员函数，派生类覆盖基类的同名成员函数。

C. 派生类可以和基类有同名成员函数，但是否覆盖同名成员函数需要取决于函数参数是否一致。

D. C++中派生类不继承基类的任何成员函数。

5. 关于下面 C++代码说法错误的是（　）。

```cpp
#include <iostream>
#include <string>
using namespace std;
class Pet {
public:
    string kind;
    int age;
    Pet(string _kind, int _age):kind(_kind), age(_age) {}
}

class Dog : public Pet {
public:
    string color;
    Dog(string _kind, int _age, string _color):Pet(_kind, _age), color(_color) {}
}

int main() {
    auto dog = Dog("dog", 3, "white");
    cout << "kind: " << dog.kind << endl;
    cout << "age: " << dog.age << endl;
    cout << "color: " << dog.color << endl;  // 输出行A
    return 0;
}
```

A. Pet 类是基类，Dog 类是子类。

B. Dog 类的构造函数中，将自动调用 Pet 类的构造函数。

C. dog 是 Dog 类的实例。

D. 最后一行（即输出行 A）输出代码会报错，因为 Pet 类中没有成员变量 color

6. 以下几个类定义中不能顺利通过编译的是（　）。

```
class A {
public:
    void func(int a, int b, int c) {};
};

class B {
public:
    void func(int a, int b=1, int c=1){}
};

class C {
public:
    void func(int a=3, int b, int c){}
};

class D {
public:
    void func(int a=3, int b=1, int c=0){}
};
```

A. class A

B. class B

C. class C

D. class D

7. 关于运算符重载，下列说法正确的是（　）。

A. 对于下列代码中的两个运算符重载实现：虽然它们的参数类型都是 int 与 Test 的组合，这段代码仍可以通过编译。这是因为在两个重载实现中，int 与 Test 在参数中的顺序不同，编译器能够唯一地确定调用哪个运算符实现代码。

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

class Test{
public:
    int data;
    Test(int d):data(d){}
};

Test operator +(const Test& a, const int& b) {
    return Test(a.data + b);
}

Test operator +(const int& b, const Test& a) {
    return Test(- a.data - b);
}

int main(){
    Test obj(1);
    int data = 2;
    cout << (obj + data).data << endl;
    cout << (data + obj).data << endl;
    return 0;
}
```

B. A 的说法是错误的。这是因为加法满足交换律，因此即便调换 A 和 int 的顺序，加法运算的实现仍必须保持一致，编译器在编译时也会检查这一点。如将后一处 int 改为 double，则可通过编译。

C. A 和 B 的说法都是错误的。运算符重载时，所有参数以及返回值的类型必须完全相同。因此，即便是下列代码中的运算符重载实现，也不能通过编译。

```cpp
class Test{
public:
    int data;
    Test(int d):data(d){}
};

Test operator +(const Test& a, const int& b) {
    return Test(a.data + b);
}
```

D. A、B、C 的说法都是错误的。

8. 关于 C++程序的异常处理，以下选项中描述错误的是（　）。

A. 编程语言中的异常和错误是不同的概念。

B. 异常一旦发生，程序便一定不能继续执行。

C. 通过 try、catch 等保留字提供异常处理功能。

D. 程序使用 throw 在任何地方抛出各种异常。

9. 有关下面C++代码的说法，正确的是（　）。

```cpp
#include <iostream>
#include <cassert>

using namespace std;

class MoreData {
    int* __data;
    int head, tail, capacity;
public:
    MoreData(int cap) {
        capacity = cap;
        __data = new int[capacity];
        head = tail = 0;
    }
    MoreData& push(int val) {
        assert(tail < capacity);
        __data[tail++] = val;
        return *this;
    }
    int pop() {
        assert(head < tail);
        return __data[--tail];
    }
    int size() {
        return tail - head;
    }
};

int main() {
    auto myData = MoreData(100);
    myData.push(4).push(5);
    cout << myData.pop() << endl;
    cout << myData.pop() << endl;
    cout << myData.pop() << endl;
    return 0;
}
```

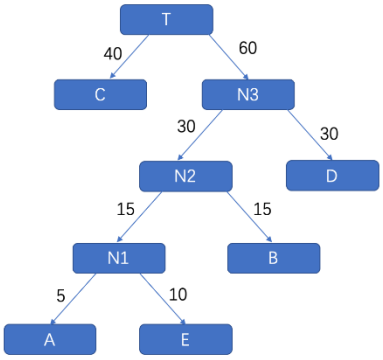
A. moreData类可用于构造队列（queue）数据结构。

B. myData.push(4).push(5);的连续push()用法将导致错误。

C. 前两个cout << myData.pop() << endl;代码可以正确运行，分别输出4和5。

D. 最后一个cout << myData.pop() << endl;代码可以通过编译，但不能正常运行，因为pop函数中的断言assert(head < tail);会失败。

10. 如下图所示的哈夫曼树，按照哈夫曼编码规则，假设图中字符C的编码为0，则E的编码为（　）。



A. 1111

B. 1010

C. 1101

D. 1001

11. 下面有关格雷码的说法，错误的是（ ）。

A. 格雷码是无权码。

B. 格雷码是循环码。

C. 011 和 100 是一组相邻的格雷码。

D. 格雷码相邻的码组间仅有一位不同。

12. 在具有$2N$个结点的完全二叉树中，叶子结点个数为（ ）。

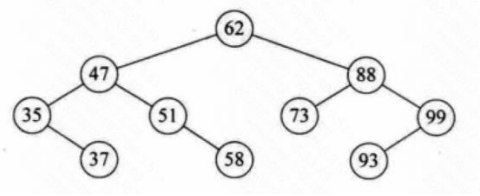
A. $N/2$

B. $N-1$

C. N

D. $N+1$

13. 有关下图的二叉树，说法正确的有（ ）个。



- 是完全二叉树
- 是二叉搜索树
- 是平衡二叉树

A. 0

B. 1

C. 2

D. 3

14. 现希望存储整数N的所有质因子，请问其空间复杂度上界为是（ ）。

A. $O(\log\log N)$

B. $O(\log N)$

C. $O(\sqrt{N})$

D. $O(N)$

15. 下面C++代码实现了某种排序算法，其中代码片段my_sort(arr, begin, i);my_sort(arr, i + 1, end);采用的算法思想是（ ）。

```
void my_sort(int arr[], int begin, int end) {
    if (begin >= end - 1) return;
    int i = begin, j = end - 1, x = arr[begin];
    while (i < j)
    {
        while(i < j && arr[j] >= x)
            j--;
        if(i < j)
            arr[i++] = arr[j];

        while(i < j && arr[i] < x)
            i++;
        if(i < j)
            arr[j--] = arr[i];
    }
    arr[i] = x;
    my_sort(arr, begin, i);
    my_sort(arr, i + 1, end);
}
```

A. 递推

B. 贪心

C. 分治

D. 搜索

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. 质数的判定和筛法的目的并不相同，质数判定旨在判断特定的正整数是否为质数，而质数筛法意在筛选出范围内的所有质数。
2. 唯一分解定理指的是分解质因数只有唯一的一种算法。
3. 一般情况下，在 C++中定义一个类时，构造函数和析构函数都不是必须手动定义的。
4. 如果一个对象具有另一个对象的性质，那么它们之间就是继承关系。
5. 哈夫曼编码树中，两个频率相同的字符一定具有相同的哈夫曼编码。
6. 宽度优先搜索算法的英文简写是 BFS。
7. 深度优先遍历算法的时间复杂度为$O(N\log N)$，其中N为树的节点数。
8. 任意二叉树都至少有一个结点的度是 2。
9. 将N个数据按照从小到大顺序存放在一个单向链表中。如果采用二分查找，那么查找的平均时间复杂度是$O(\log N)$。
10. 深度优先遍历一般需要借助数据结构栈来实现，广度优先遍历一般需要借助数据结构队列来实现。

三、编程题（每题 25 分，共 50 分）

题号	1	2
答案		

1. 下楼梯

【问题描述】

顽皮的小明发现，下楼梯时每步可以走 1 个台阶、2 个台阶或 3 个台阶。现在一共有N个台阶，你能帮小明算算有多少种方案吗？

【输入描述】

输入一行，包含一个整数N。约定$1 \leq N \leq 60$。

【输出描述】

输出一行，包含一个整数C，表示方案数。

【样例输入 1】

4

【样例输出 1】

7

【样例输入 2】

10

【样例输出 2】

274

【参考程序】

```
#include <iostream>
using namespace std;
long long downstair_record[61];
long long downstair(int n) {
    if (n == 0)
        return 1;
    if (downstair_record[n] > 0)
        return downstair_record[n];
    long long res = 0;
    if (n >= 1)
        res += downstair(n - 1);
    if (n >= 2)
        res += downstair(n - 2);
    if (n >= 3)
        res += downstair(n - 3);
    downstair_record[n] = res;
    return res;
}
int main() {
    for (int i = 0; i <= 60; i++)
        downstair_record[i] = -1;
    int n = 0;
    cin >> n;
    cout << downstair(n) << endl;
    return 0;
}
```

2. 亲朋数

【问题描述】

给定一串长度为L、由数字 0-9 组成的数字串S。容易知道，它的连续子串共有$\frac{L(L+1)}{2}$个。如果某个子串对应的数（允许有前导零）是p的倍数，则称该子串为数字串S对于p的亲朋数。

例如，数字串S为“12342”、p为 2，则在 15 个连续子串中，亲朋数有“12”、“1234”、“12342”、“2”、“234”、“2342”、“34”、“342”、“4”、“42”、“2”等共 11 个。注意其中“2”出现了 2 次，但由于其在S中的位置不同，记为不同的亲朋数。

现在，告诉你数字串S和正整数p，你能计算出有多少个亲朋数吗？

【输入描述】

输入的第一行，包含一个正整数p。约定$2 \leq p \leq 128$。

输入的第二行，包含一个长为L的数字串S。约定$1 \leq L \leq 10^6$。

【输出描述】

输出一行，包含一个正整数C，表示亲朋数的个数。

【样例输入 1】

2

102

【样例输出 1】

5

【样例解释 1】

5 个亲朋数，分别为“10”、“102”、“0”、“02”、“2”。

【样例输入 2】

2

12342

【样例输出 2】

【参考程序】

```
#include <iostream>
using namespace std;
char S[1000001];
long long st_old[128];
long long st_new[128];
int main() {
    int p = 0;
    cin >> p;
    cin >> S;
    for (int i = 0; i < p; i++)
        st_old[i] = 0;
    long long res = 0;
    for (int t = 0; S[t] != '\0'; t++) {
        for (int i = 0; i < p; i++)
            st_new[i] = 0;
        int d = S[t] - '0';
        for (int i = 0; i < p; i++)
            st_new[(i * 10 + d) % p] += st_old[i];
        st_new[d % p]++;
        res += st_new[0];
        for (int i = 0; i < p; i++)
            st_old[i] = st_new[i];
    }
    cout << res << endl;
    return 0;
}
```