复旦大学计算机科学技术学院

2021~2022 学年第二学期期末考试试卷

☑A 卷 ☐B 卷 ☐C 卷

课程名称：面向对象程序设计　　课程代码：COMP130135

开课院系：计算机科学技术学院　　考试形式：线上考试（闭卷）

姓名：＿＿＿＿＿＿　学号：＿＿＿＿＿＿＿＿　专业：＿＿＿＿＿＿＿＿

（本试卷答卷时间为 120 分钟，答案必须写在答题纸上，做在试卷上无效）

提示：请同学们秉持诚实守信宗旨，谨守考试纪律，摒弃考试作弊。学生如有违反学校考试纪律的行为，学校将按《复旦大学学生纪律处分条例》规定予以严肃处理。

题 号	一	二	三	四	总 分
得 分	24	20	24	32	100

（装订线内不要答题）

一、选择题 (每题只选择一个答案; 如选多个, 不计分。每题 3 分, 总分 24 分)。

1. 如果有#include <string>，则以下定义错误的是：(　　)

```
A. const std::string message;
B. const std::string hello("Hello");
C. const std::string exclam(5, "!");
D. const std::string s = "world";
   const std::string msg(s.begin()+1, s.end()-1);
```

2. 下列四行代码中，正确的有几行：(　　)

```
const int buf;
int cnt = 0;
const int sz = cnt;
++cnt; ++sz;
```

A. 1　　B. 2　　C. 3　　D. 4

3. 下列三对函数中，能正确重载的的有：(　　)

```
(a)  int calc(int&, int&);
     int calc(const int&, const int&);
(b)  int get();
     double get();
(c)  int* reset(int*);
     double* reset(double*);
```

A. abc　　B. ac　　C. bc　　D. c

4. 对下面的代码进行编译、链接和运行，以下说法中哪一项是正确的？(　　)

```
#include <iostream>
class A {
```

```
private:
    int value_;
public:
    A(int v) :value_(v) { std::cout << "ctor A "; }
    ~A() { std::cout << "dtor ~A\n"; }
};

int main()
{
    A a[10];
    Return 0;
}
```

A. 编译错误
B. 链接错误
C. 程序可正常运行，输出 ctor A dotr ~A
D. 程序可正常运行，但没有输出

5. 以下选项描述了 vector 类型与 list 类型的相同之处，其中错误的是：(　　)
A. vector 和 list 都可用于保存批量的同类数据
B. vector 对象和 list 对象的大小可以按需增长
C. vector 和 list 都支持 begin(), end()和 push_back()操作
D. vector 对象和 list 对象都可使用标准库算法 sort()进行原地排序

6. 以下伪代码描述了 C++的异常处理机制，选项中错误的是：(　　)

```
try {
    if(cond)  throw exception
    block A
}
catch (exception){
    block B
}
block C
```

A. 可能引发异常的程序代码应该放置于 try 模块中，否则 catch 模块无法捕获异常
B. 如果条件 cond 满足，则先执行语句块 block A，再执行语句块 block B
C. 如果条件 cond 不满足，则执行语句块 block A，语句块 block B 不会执行
D. 无论条件 cond 是否满足，语句块 block C 都会执行

7. 关于虚函数的描述中，哪些说法是正确的？(　　)
①通常虚函数的运行开销要小；
②虚函数意味着额外的空间开销和时间开销；
③虚函数是静态绑定的；
④虚函数是动态绑定的；
A. ①③　　B. ②③　　C. ①④　　D. ②④

8. 下面关于 delete 描述，错误的一项是(　　)
A. delete 必须用于 new 返回的指针
B. 使用 delete 删除对象时，会自动调用析构函数
C. 对一个指针可以多次使用 delete 运算符
D. 删除指针数组时，需要使用“delete []指针”的形式

二、程序阅读题（每题 5 分，共 20 分。说明：程序中省略了头文件和 using 声明）

2.1 以下程序的运行结果是 ：_____

```
void f(int& x, int *y, int z) {
    z = x + *y / 2;
    *y = x + z / 2;
    x = *y + z / 2;
}
int main() {
    int a = 5, b = 7, c = 11;
    f(a, &b, c);
    cout << a << ',' << b << ',' << c;
    return 0;
}
```

2.2 以下程序的运行结果是 ：_____

```
class Base {
    char c1, c2;
public:
    Base(char n = 'a') :c1(n), c2(n + 2) {}
    virtual ~Base() {
        cout << c1 << c2 << '\n';
    }
};
class Derived :public Base {
    char c3;
public:
    Derived(char n = 'A') :Base(n + 1), c3(n) {}
    ~Derived() { cout << c3; }
};
int main() {
    Derived* a = new Derived[2];
    delete[] a;
}
```

2.3 以下程序的运行结果是 ：_____

```
class Abc
{
    int value_;
    static const int SPAN = 0x64;
public:
    static int seq;
    Abc() :value_(++seq) {}
    Abc(const Abc& rhs) :value_(rhs.value_ + SPAN) { ++seq; }
    Abc& operator=(const Abc& rhs) {
        seq++;
        value_ = rhs.value_ / 2;
        return *this;
    }
    int value() const { return value_; }
```

```
};

int Abc::seq;

int main()
{
    Abc m, n, p, q(p);
    m = q;

    cout << p.seq << '\n';
    cout << m.value() << '\n';
}
```

2.4 以下程序的运行结果是 ：_____

```
class Value {
    int value_;
public:
    Value(int v = 0) : value_(v % 7) {}
    int value() const { return value_; }
};
bool filter(Value const& v) {
    cout << v.value() << ' ';
    return v.value() % 5 == 0;
}
void output(Value const& v) {
    cout << v.value() << ' ';
}
int main() {
    int a[] = { 20, 25, 30, 35, 40, 45, 50 };
    vector<Value> values(a, a + sizeof a / sizeof a[0]);
    vector<Value> filtered(values.size() / 2);
    copy_if(values.begin(), values.end(), back_inserter(filtered),
filter);
    cout << '\n';
    for (vector<Value>::iterator itr = filtered.begin(); itr !=
filtered.end(); itr++)
        output(*itr);
}
```

三、程序填空题（每空 3 分，共 24 分。说明：程序中省略了头文件和 using 声明）

循环队列就是将队列存储空间的最后一个位置绕到第一个位置，形成逻辑上的环状空间，供队列循环使用。在循环队列结构中，当存储空间的最后一个位置已被使用而要执行进队操作时，只需存储空间的第一个位置空闲，便可将元素加入到第一个位置，即将存储空间的第一个位置作为队尾。循环队列可以更简单地防止伪溢出的发生，但队列大小是固定的。

在循环队列中，当队列为空时，有front=rear，而当所有队列空间全占满时，也有front=rear。为了区别这两种情况，规定循环队列最多只能有MaxSize-1个队列元素，当循环队列中只剩下一个空的存储单元时，就判定队列已满。因此，队列判空的条件是front=rear，而队列判满的条件是front=(rear+1)%MaxSize。

以下程序使用一个大小为(N+1)的vector实现最多能容纳N个元素的循环队列类，并且提供队列的入队、出队等操作。程序的输出为：

2
3
<2><3><1>

```
        1
class LoopQueue {
public:
    typedef typename vector<T>::size_type size_type;
    LoopQueue(int capacity)       2       {
        first = last = 0;
    }
    bool isEmpty()const { return first == last; }
    bool isFull()const { return (last + 1) % data.size() == first; }
    size_type getLength()const {
        if (       3       ) return last - first;
        return last - first + data.size();
    }
    bool dequeue(T& e) {
        if (isEmpty()) return false;
        e = data[       4       ];
        data[       5       ] = T();
        return true;
    }
    bool enqueue(const T& e) {
        if (isFull()) return false;
        data[       6       ] = e;
        last =       7       ;
        return true;
    }
    void print() {
        size_type i;
        for (i = first; i != last; i = (i + 1) % data.size()) {
            cout << data[       8       ] << " ";
        }
        cout << endl;
    }
private:
    vector<T> data;
    size_type first, last;
};

int main()
{
    int a;
    LoopQueue<int> qu(3);
    for (int i = 1; i < 6; i++) {
        qu.enqueue(i);
    }
    qu.dequeue(a);
    cout << qu.getLength() << endl;
    qu.enqueue(a);
    cout << qu.getLength() << endl;
```

```
    qu.print();
    return 0;
}
```

四、编程题（32分）

1.（12 分）编写模板函数 Rotate(begin, mid, end)。

模板函数 Rotate 循环左移区间[begin, end)中的每个元素，使 mid 指向的元素成为区间的第一个元素，移出的元素循环放至区间末尾。其中，begin, mid 和 end 为前向迭代器。

以下为测试程序（头文件和 using 声明省略）：

```
int main() {
    vector<int> v;

    for (int i = 1; i < 10; ++i) v.push_back(i);
    Rotate(v.begin(), v.begin() + 3, v.end());

    cout << "v contains:";
    vector<int>::const_iterator it;
    for (it = v.begin(); it != v.end(); ++it)
        cout << ' ' << *it;
    cout << endl;

    return 0;
}
```

该测试程序的输出是：

```
v contains: 4 5 6 7 8 9 1 2 3
```

2.（20 分）夏令营选拔。对报名人员进行笔试，分数达到面试分数线的人员进入面试。面试分数线根据计划录取人数的 150%划定，即如果计划录取 m 名，则面试分数线为排名第 m×150%（向下取整）名选手的分数，不低于面试分数线的所有人员进入面试。编写程序划定面试分数线，并输出所有进入面试人员的报名号和笔试成绩。

以下是程序运行示例：

输入	输出
Enter candidates' info: 1000 90 3239 88 2390 95 7231 84 1005 95 1001 88 ^Z Enter the number of summer camps to be recruited: 3	The minimum passing score: 88 Number of interviewees: 5 Info of the interviewees: 1005 95 2390 95 1000 90 1001 88 3239 88

【示例说明】

m×150%=3×150%=4.5，向下取整后为 4。保证 4 个人进入面试的分数线为 88，但因为 88 有重分，根据题意，分数大于等于 88 的选手都可以进入面试，故最终有 5 个人进入面试。

输入数据：报名人员信息每行包括两个数据，中间用一个空格隔开，分别是报名号 （字符串类型且各不相同）和笔试成绩(0≤成绩≤100)。输入的计划招募数 m 保证 m×150%向下取整后小于等于报名总人数。

输出数据：按照笔试成绩从高到低输出，如果成绩相同，则按报名号从小到大输出。

要求：定义 candidate 结构保存单个报名人员的信息，定义 candidates 类保存所有报名人员的信息。candidates 类支持如下操作：

1）成员函数 readinfo()读入所有报名人员信息；

2）成员函数 sort_by_score_ID()对所有人员信息先按照成绩从高到低排序，成绩相同的按照报名号从小到大排序；

3）成员函数 calculate()计算面试分数线及入围人数；

4）成员函数 output()输出所有入围人员信息；

5）其它成员函数请根据测试代码要求定义。

请给出 candidates.h 和 candidates.cpp 的完整代码。

测试代码如下（保存在 main.cpp 中）：

```
#include "candidates.h"

using std::cin;       using std::cout;
using std::endl;

int main()
{
    candidates cs;
    cout << "Enter candidates' info: " << endl;
    cs.readinfo(cin);
    cs.sort_by_score_ID();

    cout << "Enter the number of summer camps to be recruited: ";
    int n;
    cin >> n;
    cs.calculate(n);
    cout << "The minimum passing score: " << cs.get_mps() << endl;
    cout << "Number of interviewees: " << cs.get_num() << endl;
    cout << "Info of the interviewees: " << endl;
    cs.output(cout);

    return 0;
}
```