概念题

1. 请简述 C++中异常处理的两种策略。
   就地处理：在发现异常错误的地方处理异常
   异地处理：在其它地方（非异常发现地）处理异常

2. C++异常处理机制中 try，throw 和 catch 语句的作用分别是什么？
   try 语句的作用是启动异常处理机制
   throw 语句用于在发现异常情况时抛掷（产生）异常对象
   catch 语句用于捕获 throw 抛掷的异常对象并处理相应的异常

3. 请简述 C++中断言（assertion）的概念和作用。
   断言（assertion）是一个描述了程序执行到断言处应满足的条件的逻辑表达式
   如果条件满足则程序继续执行下去，否则程序异常终止
   在程序开发阶段，断言可以用来帮助开发者发现程序的错误和用于错误定位

编程题

1. 请完成如下异常测试（ExceptionTest）类的若干接口。
   要求：实现接口功能；按注释进行异常就地处理（输出异常原因并终止程序执行）。

```
#include <iostream>
#include <cstdlib>
#include <cmath>
#include <fstream>
using namespace std;

class ExceptionTest{
private:
    int prime[100]; //存前100个素数（质数）
public:
    //求分数，分子分母为a和b；分母为零异常
    double fraction(double a, double b) {
        if (b == 0) {
            cout << "分母不能为零！" << endl;
            exit(-1);
        }
        else
            return a / b;
    }
    //求底数为10的对数，真数为a；真数为负异常
    double logarithm(double a) {
        if (a < 0) {
            cout << "真数不能为负！" << endl;
```

```
        exit(-1);
    }
    else
        return log10(a);
}
//求算出前100个素数，放在prime中，并写入文件；文件打开失败异常
void calPrime(const char* address) {
    ofstream out_file(address, ios::out);
    if (!out_file) {
        cout << "文件打开失败！" << endl;
        exit(-1);
    }
    else {
        int count = 0;
        for (int i = 2; ; i++) {
            bool is_prime = true;
            for (int j = 2; j * j <= i; j++) {
                if (i % j == 0) {
                    is_prime = false;
                    break;
                }
            }
            if (is_prime) {
                prime[count] = i;
                count += 1;
            }
            if (count == 100)
                break;
        }
        for (int k = 0; k < 100; k++) {
            out_file << prime[k] << ' ';
            if (k % 10 == 9)
                out_file << '\n';
        }
        out_file.close();
    }
}
//从prime中获取第i个素数；数组下标越界异常
int getPrime(int i) {
    if (i > 100 || i < 1) {
        cout << "数组下标越界！" << endl;
        exit(-1);
    }
    else
```

```
            return prime[i - 1];
    }
};
```

2. “全国青少年征文大赛”启动，以下是官网（Web）的报名流程和可能存在的异常情况：
要求：完成官网（Web）类；设计程序展示异常操作，并最终完成报名。

```
#include <iostream>
#include <fstream>
using namespace std;
#pragma warning (disable:4996)

class Web {
    char temp_name[30];
    int temp_age;
    char temp_phone[20];
    char temp_file[40];
public:
    void inputName() {
        cin >> temp_name;
    }
    void inputAge() {
        int age;
        cin >> age;
        if (age > 18 || age < 11)
            throw age;
        temp_age = age;
    }
    void inputPhone() {
        char phone[20];
        cin >> phone;
        for (int i = 0; phone[i] != '\0'; i++) {
            if (!(phone[i] >= 48 && phone[i] <= 57) && phone[i] != '-')
                throw phone;
        }
        strcpy(temp_phone, phone);
    }
    void uploadFile() {
        char file_name[50];
        cin >> file_name;
        ifstream in_file(file_name, ios::in);
        if (!in_file)
            throw file_name;
        char temp[50];
```

```
    strcpy(temp, temp_name);
    strcat(temp, ".txt");
    strcpy(temp_file, temp);
    ofstream out_file(temp_file, ios::out);
    if (!out_file)
        throw temp_file;
    out_file << "姓名：" << temp_name << endl;
    out_file << "年龄：" << temp_age << endl;
    out_file << "电话：" << temp_phone << "\n" << endl;
    char buffer[101];
    while (!in_file.eof()) {
        in_file.getline(buffer, 99);
        out_file << buffer << endl;
    }
    in_file.close();
    out_file.close();
}
void enroll() {
    cout << "请输入姓名：";
    inputName();
    cout << "请输入年龄：";
    try { inputAge(); }
    catch (int age) {
        if (age < 11)
            cout << "需要年满11周岁方能报名！" << endl;
        else if (age > 18)
            cout << "超过18周岁不能报名！" << endl;
        exit(0);
    }
    cout << "请输入电话：";
    try { inputPhone(); }
    catch (char* phone) {
        while (1) {
            cout << "电话号码" << phone << "包含非法字符，请重新输入：";
            try { inputPhone(); }
            catch (char* phone) { continue; }
            break;
        }
    }
    cout << "请输入作文文件名：";
    try { uploadFile(); }
    catch (char* file) {
        if (strcmp(file, temp_file)) {
            while (1) {
```

```
                    cout << "无法打开文件" << file << "，请重新输入文件名：";
                    try { uploadFile(); }
                    catch (char* file) { continue; }
                    break;
                }
            }
            else {
                cout << "无法创建报名文件！" << endl;
                exit(-1);
            }
        }
    }
};

int main() {
    Web WEB;
    WEB.enroll();
    return 0;
}
```

3. void *memcpy(void *dst, void *src, unsigned count)是 C++的一个库函数。
请重写该函数，使得函数能抛出异常。在 main 中调用该函数，处理异常情况并运行。
（提示：异常情况包括指针为空，两指针指向内存有重叠等）

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

void* memcpy(void* dst, void* src, unsigned count) {
    if (dst == nullptr)
        throw 1;
    if (src == nullptr)
        throw 2;
    memmove(dst, src, count);
}

int main()
{
    char *a = nullptr;
    char *b = nullptr;
    char temp[] = "ERROR";
    try { memcpy(a, b, 6); }
    catch (int error) {
        if (error == 1) {
            a = new char[6];
```

```
            try { memcpy(a, b, 6); }
            catch (int error) {
                b = temp;
                memcpy(a, b, 6);
            }
        }
        else if (error == 2) {
            b = temp;
            memcpy(a, b, 6);
        }
    }
    cout << a << endl;
    char str[] = "memmove can be very useful.....";
    memcpy(str + 20, str + 15, 11);
    puts(str);
    return 0;
}
```

4. 在所有课件的程序例子中，举出两个鲁棒性不高的程序。
   说明可能引发异常的情况并改进程序，处理异常。

以二进制方式读写文件

```
int x = 658763, y;
ofstream out_file("d:\\myfile.dat", ios::out | ios::binary);
out_file.write((char*)&x, sizeof(x));
out_file.close();

ifstream in_file("d:\\myfile.dat", ios::in | ios::binary);
in_file.read((char*)&y, sizeof(y)); //y=658763
in_file.close();
```

(658763)10　= (0x000a0d4b) 16　0a = '\n'　0d = '\r'

可能引发异常的情况：打开文件失败
改进程序，处理异常：

```
if (!out_file) exit(-1);
if (!in_file) exit(-1);
```

利用 STL 的容器 list 和迭代器实现求解约瑟夫问题

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

```
int main()
{
    int m, n; //m用于存储要报的数，n用于存储小孩个数
    cout << "请输入小孩的个数和要报的数：";
    cin >> n >> m;
    //构建圈子
    list<int> children; //children是用于存储小孩编号的容器
    for (int i = 0; i < n; i++) //循环创建容器元素
        children.push_back(i); //把i（小孩的编号）从容器
                               //尾部放入容器
  //开始报数
    list<int>::iterator current; //current为指向容器元素的迭代器
    current = children.begin(); //current初始化成指向容器的第一个元素
    while (children.size() > 1) //只要容器元素个数大于1，就执行循环
    {
        for (int count = 1; count < m; count++)  //报数，循环m-1次
        {
            current++; //current指向下一个元素
            //如果current指向的是容器末尾，current指向第一个元素
            if (current == children.end()) current = children.begin();
        }
        //循环结束时，current指向将要离开圈子的小孩
        current = children.erase(current);  //小孩离开圈子，current指向
                                //下一个元素
        //如果current指向的是容器末尾，current指向第一个元素
        if (current == children.end()) current = children.begin();
    } //循环结束时，current指向容器中剩下的唯一的一个元素，即胜利者
    //输出胜利者的编号
    cout << "The winner is No." << *current << "\n";
    return 0;
}
```

可能引发异常的情况：输入小孩的数和要报的数小于等于零
改进程序，处理异常：

```
while (n <= 0 || m <= 0) {
    cout << "小孩个数和要报的数必须为正数，请重新输入：";
    cin >> n >> m;
}
```