ĐỀ THI KẾT THÚC HỌC PHẦN
Học kỳ 2 - Năm học 2024 - 2025
PHƯƠNG PHÁP LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OOP)

Lưu ý: đề bài được soạn lại theo trí nhớ và sự hỗ trợ của AI.

Câu 1

Cho đoạn mã C++ sau:

```cpp
#include <iostream>

struct Fraction {
    friend struct DivisibleFracCounter;
    Fraction(int num, int den) : m_num(num), m_den(den) {
        std::cout << "Create.\n";
    }
    Fraction(const Fraction& f) : m_num(f.m_num), m_den(f.m_den) {
        std::cout << "Copy.\n";
    }
private:
    int m_num, m_den;
};

template <class T> struct Counter {
    void operator()(T num) { m_count += check(num); }
    int operator()() { return m_count; }
    virtual bool check(const T& num) = 0;
protected:
    int m_count = 0;
};

struct DivisibleFracCounter : public Counter<Fraction> {
    bool check(const Fraction& n) { return n.m_num % n.m_den == 0; }
};

int main() {
    Fraction a[] = { {1, 2}, {3, 3}, {4, 2} };
    int n = sizeof(a) / sizeof(*a);
    DivisibleFracCounter count;
    for (int i = 0; i < n; i++) count(a[i]);
    std::cout << count();
    return 0;
}
```

Yêu cầu:

a) Cho biết kết quả in ra màn hình của chương trình trên.

b) Chương trình trên có cách truyền dữ liệu giữa các hàm chưa được hiệu quả, hãy đề xuất cải tiến.

Câu 2

Cài đặt lớp `Fraction` và `PrimeFracIterator` để duyệt một mảng các phân số dưới dạng con trỏ. Với mỗi bước nhảy, con trỏ sẽ trỏ đến phân số tiếp theo trong mảng mà **có cả tử số và mẫu số đều là số nguyên tố**.

Minh hoạ cách sử dụng `PrimeFracIterator`:

```
#include<iostream>
int main() {
    Fraction a[] = { {1, 2}, {2, 3}, {4, 2}, {3, 5}};

    PrimeFracIterator i(a, n);
    while(i){
        std::cout << *i << " ";
        ++i;
    }
} // Ket qua in ra man hinh: 2/3 3/5
```

Ghi chú: Nội dung mô tả cho câu 3 có thể khác với đề gốc do tác giả làm đến bài này thì nước mắt không ngừng rơi làm ướt mất đề.

Câu 3

Bài toán yêu cầu cài đặt một lớp `Extractor`. Lớp này đóng vai trò là thành phần điều phối chính, có nhiệm vụ quản lý một tập hợp các bộ lọc và áp dụng chúng để trích xuất các phần tử từ một mảng dữ liệu đầu vào.

Để thực hiện việc lọc, `Extractor` sử dụng các đối tượng bộ lọc (`Filter`). Quy trình lọc diễn ra như sau: Với mỗi phần tử, nó sẽ được kiểm tra lần lượt với **tất cả** các bộ lọc. Một phần tử chỉ được giữ lại trong mảng kết quả nếu nó thỏa mãn điều kiện của tất cả các bộ lọc đó.

Thiết kế của bộ lọc cần đủ tổng quát để có thể dễ dàng mở rộng và thêm các loại bộ lọc mới với logic khác nhau. Cụ thể trong pham vi bài toán này, cần cài đặt các bộ lọc sau:

- **`DivisibleFilter`:** Lọc ra các phần tử chia hết cho một số nguyên `N` cho trước.
- **`OddIndexFilter`:** Lọc ra các phần tử nằm ở các vị trí có chỉ số lẻ trong mảng ban đầu.
- **`DiffAvgFilter`:** Lọc ra các phần tử có giá trị sai khác không quá `M` % so với giá trị trung bình của toàn bộ mảng ban đầu.

Code minh họa:

```
#include <iostream>
#include <vector>

int main() {
    Extractor<int> ex;
    ex.add(new DivisibleFilter(3));
    ex.add(new OddIndexFilter<int>());
    ex.add(new DiffAvgFilter(0.2));
    std::vector<int> a = {10, 3, 15, 6, 2, 9, 4, 15, 5, 6};
    std::vector<int> b = ex.extract(a);
    for (auto& e : b) {
        std::cout << e << " ";
    } // Ket qua in ra: 6 9 6
    return 0;
}
```

Yêu cầu:

a) Vẽ sơ đồ lớp UML thể hiện mối quan hệ giữa các lớp.

b) Cài đặt các lớp cần thiết.