ĐỀ THI CUỐI KỲ - CA B

Thời gian làm bài: 60 phút

Quy định chung

- Đây là bài thi tại lớp. Sinh viên KHÔNG được sử dụng tài liệu, máy tính cá nhân, điện thoại di động và tham khảo internet, trừ khi có sự đồng ý của Giảng viên.
- Sinh viên không được trao đổi dưới bất kì hình thức nào.
- Sinh viên làm và nộp tất cả các bài trong cùng 1 file <**MSSV**>**.cpp**.
- Sinh viên cần chú thích bài làm rõ ràng.
- Sinh viên không cần kiểm tra tính hợp lệ của input.
- Các bài làm không biên dịch được, hoặc lỗi runtime đều sẽ bị **0 điểm**.
- Vi phạm 1 trong các quy định trên, sẽ bị **0 điểm** bài thi cuối kì.

(Sinh viên sang trang kế tiếp để xem đề bài.)

Nội dung

Lưu ý: con trỏ rỗng được setup mặc định = NULL (không phải nullptr).

Câu 0

Cho file có tên `filename`, thông tin từ file này sẽ là dữ liệu cho các câu **1**, **2** và **3** phía dưới, dữ liệu của mỗi câu cách nhau 1 dòng trắng (xem file `data.txt` để biết thêm chi tiết). Hãy đọc dữ liệu từ file và lưu vào các biến (tham chiếu) của hàm `readfile`.

Lưu ý: Sinh viên cần đọc hết đề để biết từng dạng dữ liệu tương ứng trong file cần được trả về. Phần đọc file của mỗi câu sẽ có điểm riêng, đọc được dữ liệu của câu nào sẽ có một phần điểm tương ứng của câu đó.
Prototype:

```
void readfile(string filename, int &number, int** &matrixA, int &m, int &n, int** &matrixB,
                                            int &p, int &q, List& picture)
```

Câu 1 *(2 điểm)*

Cho số tự nhiên n. Viết hàm phân tích n thành tổng của các số là bội số của luỹ thừa 10 và in kết quả ra màn hình (xem ví dụ).

Yêu cầu: bắt buộc sử dụng đệ quy.

Prototype:

```
void solution_ex1(int number)
```

Ví dụ: $solution\_ex1(1234) \Rightarrow$ sẽ in ra màn hình 1000 200 30 4.

Câu 2 *(4 điểm)*

Một ma trận được gọi là sinh từ ma trận khác khi mỗi vị trí của ma trận đó là bội của vị trí tương ứng trong ma trận còn lại. Cho ma trận A kích thước $m \times n$ và ma trận B kích thước $p \times q$ ($p < m$, $q < n$). Hãy viết hàm tìm ma trận 3 chiều là tập hợp của tất cả các ma trận con A' của A, sao cho A' được sinh từ B.
Prototype:

```
int*** solution_ex2(int** matrixA, int m, int n, int** matrixB, int p, int q,
                                                 int& d1, int& d2, int& d3)
```

Ví dụ:

```
A = {{100, 960, 204}, {693, 847, 805}, {147, 101, 592}}, m = 3, n = 3
B = {{2, 3}, {11, 7}}, p = 2, q = 2
=> solution_ex2(A, 3, 3, B, 2, 2, d1, d2, d3)
    = {{{100, 960}, {693, 847}}, {{960, 204}, {847, 805}}}
```

với `d1 = 2, d2 = 2, d3 = 2`

Câu 3 *(4 điểm)*

Cho định nghĩa `Color`, `Node` và `List` trong danh sách liên kết đơn.

```
struct Color {
    int red;
    int green;
    int blue;
};
```

```
struct Node {
    Color base; // 1 mau rgb duoc
        tao tu red, green, blue
    Node *next;
};
```

```
struct List {
    Node *head;
    Node *tail;
};
```

Mỗi Node tượng trưng cho 1 pixel có màu Color. Xét 1 List cho trước là tập hợp các pixel trong 1 khung hình. Viết hàm xoá màu xuất hiện ít nhất trong khung hình cho trước (giả sử không có 2 màu khác nhau nào có số lần xuất hiện bằng nhau).
Prototype:

```
void solution_ex3(List& picture)
```

Ví dụ:

`picture = {{1, 1, 1}, {1, 2, 3}, {1, 1, 1}, {1, 2, 3}, {1, 2, 3}}`

Màu `1, 1, 1` xuất hiện ít lần nhất.

`picture (new) = {{1, 2, 3}, {1, 2, 3}, {1, 2, 3}}`

Hết