Lab 2: Ôn tập

1 Struct

1. Viết chương trình nhập vào 2 mốc thời gian. Tính tổng thời gian và in ra màn hình.

```
Input:
    start_hrs: 1
    start_mins: 20
    start_secs: 0
    end_hrs: 0
    end_mins: 50
    end_secs: 0
Output:
    start_time: "01:20:00"
    end_time: "00:50:00"
    total: "02:10:00"
```

```
template <class T>
struct time
{
  T hrs;
  T mins;
  T secs;
};
```

2. Viết chương tình cộng 2 phân số và in ra màn hình.

```
Input:
    Frac1:
    1
    6
    Frac2:
    2
    4
Output:
    Frac1: 1/6
    Frac2: 1/2
    Sum: 2/3
```

```
template <class T>
struct fraction
```

```
        {
            T num; // the numerator of the fraction
            T denom; // the denominator of the fraction
        };
```

2 Danh sách liên kết

Cho một danh sách liên kết đơn được định nghĩa như sau:

```
struct NODE{
    int key;
    NODE* p_next;
};
```

```
struct List{
    NODE* p_head;
    NODE* p_tail;
};
```

Hãy viết hàm thực hiện các yêu cầu sau:

1. Chèn một số nguyên vào vị trí bất kì một `List` cho trước:
    - `bool addPos(List* L, int data, int pos)`
2. Xóa `NODE` ở vị trí bất kì của một `List` cho trước:
    - `void removePos(List* L, int pos)`
3. Cho 2 `List` đã được sắp xếp tăng dần. Ghép 2 `List` lại thành một `List` mới được sắp xếp tăng dần.
    - `List* ConcatList(List* L1, List* L2)`
4. Cho 2 `List` đã được sắp xếp tăng dần. Ghép `List` 2 vào `List` 1 mà vẫn giữ sắp xếp tăng dần.
    - `void MergeList(List* L1, List* L2)`

3 Danh sách liên kết - 2

Cho một danh sách liên kết đơn được định nghĩa như sau:

```
struct NODE2{
    string student_id;
    string name;
    int birth_year;
    NODE* p_next;
};
```

```
struct List{
    NODE* p_head;
    NODE* p_tail;
};
```

Hãy viết hàm thực hiện các yêu cầu sau:

1. Viết hàm khởi tạo Node. Khởi tạo Node có giá trị "123", "Nguyễn Văn A", 2004.
2. Viết hàm in ra giá trị của Node.
3. Viết hàm thêm một node vào danh sách liên kết
4. Viết hàm thay đổi tên của học sinh trong danh sách

4 Stack - Queue

Cho định nghĩa struct của một node trong danh sách liên kết đơn như sau:

```
struct NODE{
    int key;
    NODE* pNext;
};
```

Sử dụng Danh sách liên kết phía trên, định nghĩa cấu trúc dữ liệu cho Ngăn xếp và Hàng đợi, sau đó cài đặt hàm thực hiện các chức năng sau đây:

1. Viết hàm đọc theo một chuỗi ký tự và in chúng theo thứ tự ngược lại. Sử dụng một ngăn xếp.
   - `void ReverseString(char* s)`
2. Viết chương trình đọc trong một chuỗi ký tự và xác định xem dấu ngoặc đơn (), dấu ngoặc nhọn `{}` và dấu ngoặc vuông [ ] của nó có "cân bằng" hay không.

   VD:

   Input: "(ab))"

   Output: false
   - `void CheckBalance(char* s)`
3. Viết chương trình đọc một số nguyên dương và in ra biểu diễn nhị phân của số nguyên đó VD:

   Input: 2

   Output: "0000000000000010"
   - `void DecToBin(int dec)`