TỔNG HỢP CÁC BÀI LAB KTLT - HỌC KÌ 232

Quang Nguyễn

2024

LỜI NÓI ĐẦU

"Kĩ thuật lập trình" là một môn học bắt buộc dành cho sinh viên khoa Khoa học và Kĩ thuật Máy tính. Môn học này thường được mở các lớp chính quy vào học kì chính thứ hai, và các lớp dự thính được tổ chức quanh năm. Trong học kì 232, môn Kĩ thuật lập trình (CO1027) bao gồm ba buổi thí nghiệm (lab). Mỗi buổi thí nghiệm được chia thành ba phần: Prelab, Inlab và Postlab. Phần Prelab gồm các câu hỏi từ mức độ dễ đến trung bình, yêu cầu sinh viên hoàn thành trước buổi thí nghiệm. Phần Inlab bao gồm các câu hỏi từ mức độ trung bình đến khó, cần phải hoàn thành trong buổi thí nghiệm hoặc trong vài ngày sau đó. Phần Postlab chứa các câu hỏi khó, yêu cầu hoàn thành sau buổi thí nghiệm trong một khoảng thời gian cố định. Các buổi thí nghiệm này được tính điểm riêng một cột trong môn học này, với điểm các bài lab chiếm 50% tổng số điểm và bài thi cuối kì chiếm 50% còn lại. Quy định về thời gian hoàn thành các phần Prelab, Inlab, Postlab cũng như nội dung đề thi sẽ do giảng viên giảng dạy quyết định, nên không được đề cập chi tiết ở đây.

Tài liệu này tập hợp gần như đầy đủ tất cả các bài tập trong các buổi lab, cùng một số câu hỏi xuất hiện trong các đề thi cuối lab (vì đề thi giữa các lớp là khác nhau). Độc giả có thể sử dụng tài liệu này như một nguồn tham khảo và luyện tập cho môn học, cũng như để nâng cao kỹ năng lập trình của bản thân.

Mặc dù tài liệu đã được rà soát và chỉnh sửa kỹ lưỡng, nhưng không thể tránh khỏi những sai sót. Rất mong nhận được sự thông cảm từ quý độc giả.

Tác giả

Mục lục

Chương I

LAB 1

1 C-String

Câu 1. Viết hàm `void reverse(char str[])` đảo ngược các ký tự trong chuỗi.

- **Mô tả hàm:** Hàm này thực hiện việc đảo ngược chuỗi ký tự truyền vào ngay tại chỗ. Hàm không trả về giá trị và thay đổi chuỗi đầu vào để chuỗi sau khi đảo ngược thay thế chuỗi ban đầu.
- **Đầu vào:** Mảng kí tự chứa chuỗi cần đảo ngược.
- **Đầu ra:** Chuỗi đầu vào được đảo ngược ngay tại chỗ, không cần chuỗi đầu ra `outstr`.
- Các thư viện có thể dùng: `<iostream>`, `<cstring>`.

Lưu ý: Sinh viên không được sử dụng các từ khoá sau trong đoạn code nộp bài, kể cả trong comment: `include`, `string`.

Câu 2. Một đường truyền thông tin gồm các kí tự bị hacker tấn công. Cuộc tấn công chỉ làm ảnh hưởng đến các tín hiệu có giá trị là kí tự trong bảng chữ cái làm cho các chữ thường thành chữ in hoa và ngược lại các chữ in hoa thành chữ thường. Sinh viên được chủ đường truyền nhờ để khôi phục dữ liệu lại về như ban đầu.
Sinh viên viết hàm xử lý `void recover(char signal[])` để biến đổi tất cả chữ cái in hoa thành chữ viết thường và chữ cái thường thành chữ in hoa có trong chuỗi `signal` (lưu ý biến đổi trực tiếp trên signal).

- **Đầu vào:** Chuỗi kí tự cần biến đổi chữ hoa thành chữ thường và chữ thường thành chữ hoa.
- **Đầu ra:** Kết quả biến đổi trực tiếp trên `signal` (lưu ý các kí tự không phải kí tự trong bảng chữ cái thì giữ nguyên).

Câu 3. Viết hàm `int find(char str[], char substr[])` để tìm vị trí của chuỗi con trong một chuỗi cho trước. Hàm trả về vị trí đầu tiên của chuỗi con trong chuỗi ban đầu. Nếu không tìm thấy chuỗi con, hàm trả về `-1`.

- **Mô tả hàm:** Hàm này thực hiện việc tìm kiếm chuỗi con trong chuỗi ban đầu và trả về vị trí đầu tiên của chuỗi con nếu được tìm thấy. Nếu không tìm thấy, hàm trả về -1.
- **Đầu vào:**
  - `char str[]`: Mảng kí tự chứa chuỗi ban đầu.
  - `char substr[]`: Mảng kí tự chứa chuỗi con cần tìm.
- **Đầu ra:** `int`: Vị trí đầu tiên của chuỗi con trong chuỗi ban đầu. Nếu không tìm thấy, trả về `-1`.
- Các thư viện có thể dùng: `<iostream>`, `<cstring>`.

Lưu ý: Sinh viên không được sử dụng các từ khoá sau trong đoạn code nộp bài, kể cả trong comment: `include`, `string`.

Câu 4. Viết hàm `void printFirstRepeatedWord(char str[])` in ra từ đầu tiên trong chuỗi bị lặp lại. Từ bị lặp lại trong một chuỗi là từ mà từ vị trí của nó trở về phía trước có xuất hiện một từ

giống với nó.

- **Đầu vào:** Mảng kí tự chứa chuỗi
- **Đầu ra:** In ra từ đầu tiên trong chuỗi có lặp lại. Nếu không có từ nào lặp lại thì in ra `No Repetition`.
- Các thư viện có thể dùng: `<iostream>`, `<cstring>`.

Lưu ý: Sinh viên không được sử dụng các từ khoá sau trong đoạn code nộp bài, kể cả trong comment: `include`, `string`. Mỗi từ trong chuỗi có không quá 30 ký tự.

Câu 5. Viết hàm `void process(const char str[], char outstr[])` loại bỏ các khoảng trắng thừa trong chuỗi sao cho không còn hai khoảng trắng liền kề nhau và không có khoảng trắng nào ở đầu và cuối chuỗi.

- **Đầu vào:** Mảng kí tự chứa chuỗi cần loại bỏ khoảng trắng
- **Đầu ra:** Kết quả của bài toán được ghi vào chuỗi `outstr`.
- Các thư viện có thể dùng: `<iostream>`, `<cstring>`.

Lưu ý: Sinh viên không được sử dụng các từ khoá sau trong đoạn code nộp bài, kể cả trong comment: `include`, `string`.

2 Class String

Câu 6. Hiện thực hàm `cutString(string s, int index)` để in ra chuỗi con của chuỗi `s` từ vị trí `index` đến hết (`index` tính từ 0). Nếu `index` không hợp lệ cho chuỗi thì không in ra gì cả.

Câu 7. Hiện thực hàm `findAllIndex(string s1, string s2)` để in ra tất cả các vị trí xuất hiện của kí tự đầu tiên của chuỗi `s2` trong chuỗi `s1`. Nếu không tìm thấy in ra `-1`. Các vị trí tìm thấy sẽ cách nhau một khoảng trắng, sau vị trí cuối cùng, không in thêm bất kỳ ký tự nào (kể cả khoảng trắng, dấu xuống hàng).

Câu 8. Hiện thực hàm `void replaceString(string s, string s1, string s2)` để in ra chuỗi `s` sau khi đã thay thế chuỗi con `s1` cuối cùng thành `s2` có trong `s`. Nếu không tìm thấy chuỗi `s1` thì giữ nguyên chuỗi `s` và in ra.

Câu 9. Hiện thực hàm `deleteWord(string s, string s1)` với chức năng in ra chuỗi `s` sau khi xóa tất cả các chuỗi `s1` có trong `s`.

Câu 10. Viết chương trình đọc vào chuỗi `s` từ bàn phím, tìm và trả về chuỗi con có tính chất đối xứng dài nhất có trong `s`.

- **Đầu vào:** Chuỗi `s`.
- **Đầu ra:** Chuỗi con có tính chất đối xứng dài nhất có trong `s`. Trong trường hợp có nhiều chuỗi con dài nhất có cùng độ dài, trả về chuỗi con có vị trí bắt đầu nhỏ nhất.

Câu 11. Viết chương trình tìm và trả về độ dài của chuỗi con dài nhất không có ký tự nào được lặp lại có trong `s`.

- **Đầu vào:** Chuỗi `s`.
- **Đầu ra:** Độ dài của chuỗi con dài nhất không có ký tự nào được lặp lại có trong `s`.

3 Multi-Dimensional Array

Câu 12. Cho mảng 2 chiều chứa các số nguyên, kích thước M x N. Hiện thực hàm `int findMaxColumn (int arr[][1000], int row, int col)` để tìm chỉ số của cột có tổng tất cả các phần tử lớn nhất, trong đó `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng.
Lưu ý: Cột đầu tiên được đánh chỉ số 0. Nếu có nhiều hơn một cột có tổng lớn nhất, ta chọn cột có chỉ số lớn nhất.

Câu 13. Cho mảng 2 chiều chứa các số nguyên, kích thước N x N. Hiện thực hàm `int diagonalProduct (int arr[][1000], int row, int col)` để tìm tích của tất cả các phần tử trong đường chéo chính của mảng, trong đó `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng.

Câu 14. Cho mảng 2 chiều chứa các số nguyên, kích thước N x N. Một ma trận được gọi là đối xứng nếu với mọi `i,j`; giá trị của phần tử ở hàng `i`, cột `j` luôn bằng giá trị của phần tử ở hàng `j`, cột `i`. Hiện thực hàm `bool isSymmetric(int arr[][1000], int row, int col)` để kiểm tra xem mảng này có phải là một ma trận đối xứng hay không, trong đó `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng.

Câu 15. Cho mảng 2 chiều chứa các số nguyên, kích thước M x N. Tổng của một đường chéo là tổng tất cả các phần tử nằm trên đường chéo đó. Hiện thực hàm `int diagonalDiff(int arr[][1000], int row, int col, int x, int y)` để tìm giá trị tuyệt đối của hiệu giữa hai đường chéo đi qua ô có số hàng `x` và số cột `y`, trong đó `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng, số cột của mảng; `x` và `y` biểu thị ô có số hàng là `x` và số cột là `y` trong mảng đã cho ($0 \leq$ `x` $<$ `row` và $0 \leq$ `y` $<$ `col`).

Câu 16. Cho mảng 2 chiều chứa các số nguyên, kích thước M x N. Một hàng trong mảng được gọi là HN1 nếu trong hàng đó, mỗi phần tử đều có giá trị không lớn hơn các phần tử đứng sau nó. Hiện thực hàm `int ascendingRows(int arr[][1000], int row, int col)` để tìm số hàng HN1 có trong mảng, trong đó `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng.

Câu 17. Cho mảng 2 chiều chứa các số nguyên, kích thước M x N. Một cột của mảng được gọi là HN2 nếu tổng tất cả các phần tử trong cột đó là số nguyên tố. Hiện thực hàm `int primeColumns(int arr[][1000], int row, int col)` để tìm số cột HN2 có trong mảng, trong đó `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng.

Câu 18. Cho mảng 2 chiều chứa các số nguyên, kích thước M x N. Hiện thực hàm `int specialCells (int arr[][1000], int row, int col)`, trong đó; `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng. Một ô trong mảng được gọi là HN3 nếu tổng tất cả các phần tử trong hàng chứa ô đó và tổng tất cả các phần tử trong cột chứa ô đó đều là số nguyên tố. Tìm số ô HN3 trong mảng.
Ghi chú: (Các) thư viện `iostream`, `vector` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

Câu 19. Cho mảng 2 chiều chứa các số nguyên, kích thước M x N. Hiện thực hàm `int subMatrix(int arr[][1000], int row, int col)`, trong đó; `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng. Một mảng con kích thước 2x2 thuộc mảng đã cho được gọi là HN4 nếu tổng tất cả các phần tử trong nó là một số lẻ. Tìm số mảng con HN4 trong mảng đã cho.
Ghi chú: (Các) thư viện `iostream`, `vector` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

4 FileIO

Câu 20. Viết hàm `void calSum(string fileName)` để tính tổng của các số nguyên không âm được đọc vào từ một file đuôi txt (các số nằm trên 1 hàng phân cách với nhau bằng một khoảng trắng).

- **Đầu vào:** Biến `fileName` là tên file dữ liệu chứa các số nguyên không âm (các số nằm trên 1 hàng phân cách với nhau bằng một khoảng trắng).
- **Đầu ra:** Một số nguyên là tổng của các số nguyên đầu vào.

Câu 21. Viết hàm `void uppercase(string output)` để đọc vào chuỗi `S` từ bàn phím, sau đó chuyển tất các ký tự trong chuỗi `S` thành ký tự viết hoa và xuất kết quả ra file `output`.
Chú ý: chỉ thay đổi các chữ cái in thường, các kí tự khác sẽ được giữ nguyên.

- **Đầu vào:** Biến `output` chứa tên file dùng để xuất kết quả.

- **Đầu ra:** Hàm đọc chuỗi `S` từ bàn phím và xử lý chuỗi như mô tả. Sau đó ghi chuỗi đã xử lý vào file có tên được chứa trong biến `output` (hàm không trả về kết quả).

Câu 22. Viết hàm `void process(string fileName)` đọc một file đuôi txt, dòng đầu gồm 2 số N và M cách nhau bởi 1 khoảng trắng, N dòng theo sau, mỗi dòng gồm M số thực cách nhau bởi 1 khoảng trắng; trả về giá trị lớn nhất của mỗi dòng và giá trị lớn nhất trong tất cả các số.

- **Đầu vào:** Biến `fileName` là tên file chứa dữ liệu đầu vào.
- **Đầu ra:** Giá trị lớn nhất của mỗi dòng và giá trị lớn nhất của tất cả các số, cách nhau bởi ký tự khoảng trắng.

Câu 23. Viết hàm `void studendGrading(string fileName)` đọc một file đuôi `txt`, dòng đầu gồm 1 số nguyên dương N, N dòng theo sau mỗi dòng chứa 4 số thực lần lượt là điểm số của các môn NMĐT, KTLT, DSA và PPL của N học sinh.
Điểm trung bình (ĐTB) của sinh viên sẽ là trung bình cộng của 4 cột điểm trên. Sinh viên sẽ được xếp loại dựa trên ĐTB như sau:

- Loại A nếu ĐTB ≥ 8 và không có môn nào dưới 5.
- Loại B nếu $8 >$ ĐTB ≥ 6.5 và không có môn nào dưới 5.
- Loại C nếu $6.5 >$ ĐTB ≥ 5 và không có môn nào dưới 5.
- Loại D cho các trường hợp còn lại.

Xác định số lượng sinh viên mỗi loại và xuất kết quả ra màn hình.

- **Đầu vào:** Biến `fileName` là tên file chứa chứa thông tin về điểm số của sinh viên.
- **Đầu ra:** Số lượng sinh viên mỗi loại được ghi ra màn hình.

Câu 24. Viết hàm `void calMoney(string price, string buy)` đọc vào 2 file `price.txt` và `buy.txt`. Trong đó:

- File `price.txt` chứa thông tin về các sản phẩm, gồm: Dòng đầu tiên chứa số nguyên dương N ($N \leq 20$) là số lượng sản phẩm hiện có trong cửa hàng. N dòng tiếp theo mỗi dòng chứa 2 số nguyên dương là ID và giá của các sản phẩm tương ứng. Ví dụ:

```
3
1 10
2 15
3 12
```

  Có nghĩa là: hiện tại có 3 sản phẩm được bán trong cửa hàng, sản phẩm 1 giá 10 sản phẩm 2 giá 15 và sản phẩm 3 có giá là 12.
- File `buy.txt` chứa thông tin về việc mua hàng hóa của khách hàng. Dòng đầu tiên chứa số nguyên dương M là số lượng khách hàng mua hàng trong 1 ngày. M dòng tiếp theo mỗi dòng chứa thông tin như sau: sẽ có nhiều giá trị nằm trên một hàng, cách nhau bởi một khoảng trắng. Giá trị đầu tiên sẽ là tên khách hàng (dạng `string`), các giá trị còn lại sẽ đi theo cặp với nhau, tương ứng là: `<ID sản phẩm>_<số lượng tương ứng>`. Ví dụ:

```
2
A 1 2 2 3
B 1 3 3 2
```

  Có nghĩa là có 2 khách hàng A và B, A mua 2 sản phẩm loại 1 và 3 sản phẩm loại 2; B mua 3 sản phẩm loại 1 và 2 sản phẩm loại 3.
  Lưu ý: ID của sản phẩm và tên khách hàng là duy nhất (không lặp lại).

Tính số tiền mỗi khách hàng cần trả, sau đó xuất kết quả ra màn hình.

- **Đầu vào:** 2 biến `price` (là tên của file chứa thông tin sản phẩm) và `buy` (là tên của file chứa thông tin mua hàng).
- **Đầu ra:** Số tiền mỗi khách hàng cần trả.

Câu 25. Viết hàm `void manage(string library, string book, string author)` đọc vào 3 file `library.txt`, `book.txt` và `author.txt`. Trong đó:

- File `library.txt` chứa thông tin của các thư viện, gồm: dòng đầu tiên chứa số nguyên dương N là số lượng thư viện được khảo sát. N dòng tiếp theo, mỗi dòng chứa 6 giá trị được phân cách nhau bằng dấu khoảng trắng. Cho mỗi dòng, giá trị đầu tiên là tên của Thư viện (tên Thư viện là duy nhất), 5 giá trị còn lại là 5 số nguyên dương, là ID của 5 quyển sách có trong thư viện đó. Ví dụ:

```
5
A 1 2 3 4 5
LB 5 3 1 2 4
LC 4 1 5 2 3
```

- File `book.txt` chứa thông tin của các quyển sách, gồm: dòng đầu tiên chứa số nguyên dương M là số lượng đầu sách có trong tất cả các thư viện. M dòng tiếp theo mỗi dòng chứa 3 giá trị (phân cách nhau bởi một dấu khoảng trắng) có ý nghĩa như sau: giá trị đầu tiên là một số nguyên dương đại diện cho Mã số sách (ID - ID là duy nhất), giá trị thứ 2 là năm sản xuất và giá trị cuối cùng là thể loại. Ví dụ:

```
5
1 2000 A
2 2001 B
3 1993 D
4 1997 A
5 1995 C
```

- File `author.txt` chứa thông tin của các tác giả, gồm: dòng đầu tiên chứa số nguyên dương P là số lượng tất cả các tác giả của các sách trong các thư viện (giả sử 1 quyển sách chỉ được sáng tác bởi 1 tác giả). P dòng tiếp theo mỗi dòng chứa các giá trị như sau (các giá trị được phân cách với nhau bằng 1 dấu khoảng trắng): giá trị đầu tiên là Tên tác giả, các giá trị còn lại là ID của các quyển sách mà người đó đã sáng tác. Ví dụ:

```
3
David 1 5
John 3
Henry 2 4
```

Xác định xem Thư viện L có chứa tác phẩm nào của Tác giả A hay không, nếu có xuất ra màn hình giá trị `True`, ngược lại xuất ra `False`, với L và A được nhập vào từ bàn phím.

- **Đầu vào:** 3 biến `library`, `book` và `author` lần lượt chứa tên file `library.txt`, `book.txt` và `author.txt`.
- **Đầu ra:** `True` hoặc `False` ứng với đầu vào.

5 Function

Câu 26. Hiện thực hàm tính giai thừa của số N, sau đó gọi hàm vừa hiện thực trong hàm `main` để gán kết quả tính được vào biến `result` theo template sau.

```
1 #include <iostream>
2
3 using namespace std;
4 // implement calculate factorial function in here
5 # TODO
```

```
int main(int narg, char** argv)
{
    int N;
    cin >> N;
    long result;
    // call function calculate factorial in here and assign value to the variable result
    # TODO



    cout << result << endl;
    return 0;
}
```

Câu 27. Viết hàm `sum2` để tính tổng giá trị các phần tử trong mảng số nguyên.
Tham số:

- `int* array`: mảng số nguyên.
- `int size`: số phần tử trong mảng.
- `int& result`: tham số để lưu kết quả cuối cùng sau khi tính toán.

Câu 28. Viết hàm `bool completeNum(int N)` để kiểm tra xem số nguyên dương N có phải là một số hoàn thiện hay không. N là một số hoàn thiện nếu N bằng tổng tất cả ước số nguyên dương (không bao gồm chính nó) của nó.

- **Đầu vào:** `int N`: số nguyên dương N cần kiểm tra.
- **Đầu ra:** trả về `true` nếu N là số hoàn thiện, ngược lại trả về `false`.

Câu 29. Một chuỗi được gọi là palindrome nếu chuỗi đó giống với chuỗi được đảo ngược từ chính nó. Ví dụ: "eye", "noon", "abcba"... Hãy viết hàm kiểm tra xem một chuỗi có là palindrome hay không?

- **Đầu vào:** `const char* str`: chuỗi cần kiểm tra palindrome. `str` chỉ bao gồm chữ cái thường.
- **Đầu ra:** `bool`: `true` nếu chuỗi `str` là palindrome, ngược lại `false`.

Câu 30. Một số tự nhiên n được gọi là đặc biệt khi và chỉ khi n là số nguyên tố và tổng các chữ số của n cũng là số nguyên tố. Viết hàm kiểm tra một số tự nhiên có đặc biệt hay không.

- **Đầu vào:** `int n`: số tự nhiên cần kiểm tra có phải số đặc biệt không.
- **Đầu ra:** `bool`: trả về `true` nếu n là số đặc biệt, ngược lại trả về `false`.

Câu 31. Viết một hàm mã hóa và một hàm giải mã một đoạn `text` theo phương pháp Caesar Cipher. Để mã hoá và giải mã một chuỗi ký tự `text`, ta cần một tham số có giá trị nguyên là `shift`.
Hàm mã hóa (tên `encrypt`) sẽ thay đổi từng chữ cái trong `text` bằng cách dịch chuyển chữ cái đó sang phải `shift` lần trong bảng chữ cái. Ví dụ với `shift = 3`. Khi đó 'a' được mã hoá thành 'd', 'b' được mã hoá thành 'e',... 'z' được mã hoá thành 'c'.
Hàm giải mã (tên `decrypt`) sẽ nhận một chuỗi ký tự `text` và giá trị nguyên `shift` và giải mã chuỗi ký tự này thành chuỗi ban đầu (tức dịch chuyển từng chữ cái sang trái `shift` lần trong bảng chữ cái).

- **Đầu vào:**
  - `char* text`: chuỗi ký tự cần được mã hoá hoặc giải mã, chỉ bao gồm chữ cái thường và hoa
  - `int shift`: giá trị dịch chuyển trong Caesar Cipher
- **Đầu ra:** Hàm không trả về. Chuỗi ký tự truyền vào text sẽ thay đổi trực tiếp trong hàm.

Câu 32. Viết hàm kiểm tra các phần tử trong mảng có duy nhất hay không.

- **Đầu vào:**
  - `int* arr`: mảng số tự nhiên
  - `int n`: số lượng phần tử trong mảng
- **Đầu ra:** `bool`: trả về `true` nếu các phần tử trong mảng là duy nhất, ngược lại trả về `false`.

Câu 33. Sinh viên hiện thực hàm `int numberOfDiffCharac(string str)` để trả về số kí tự phân biệt trong chuỗi `str`.
Lưu ý: Chuỗi `str` chỉ bao gồm các kí tự từ a đến z (các chữ cái lowercase trong bảng chữ cái tiếng Anh).

Chương II

LAB 2

1 Recursion

Câu 34. Cho số nguyên n và số nguyên dương e, trong đó n là số cần tính lũy thừa và e là số mũ. Hãy viết một hàm đệ quy `int calculate_power(int n, int e)` để tính giá trị của n^e.
Lưu ý: Không được sử dụng các từ khóa như `for`, `while`, `goto` (thậm chí là tên biến, comment). Trong bài tập này đã khai báo `#include <iostream>` và `using namespace std;`.

Câu 35. Một hàm tìm ước số chung lớn nhất của 2 số nguyên dương có thể viết thông qua đệ quy và vòng lặp đơn giản. Bạn hãy viết hàm `gcdRecursion` để hiện thực tìm ước chung lớn nhất bằng đệ quy và hàm `gcdIteration` để tìm ước số chung lớn nhất bằng vòng lặp.

- **Đầu vào:** Lần lượt 2 số nguyên p, q ($1 \leq p, q \leq 10^9$).
- **Đầu ra:** Hàm `gcdRecursion` và `gcdIteration` lần lượt trả về giá trị là ước chung lớn nhất của p, q.

Viết theo template sau:

```cpp
#include<iostream>
#include <string>
#include <sstream>
#include <fstream>
#include <vector>
using namespace std;
/* END of library */

int gcdRecursion(int p, int q)
{
    // BEGIN YOUR IMPLEMENTATION [1]
    // TODO

    // END YOUR IMPLEMENTATION [1]
}

int gcdIteration(int p, int q)
{
    // BEGIN YOUR IMPLEMENTATION [2]
    // TODO

    // END YOUR IMPLEMENTATION [2]
    return 0;
}
```

```cpp
int main()
{
    hiddenCheck();
    int p,q;
    cin>>p>>q;
    cout<<gcdRecursion(p,q)<< " "<<gcdIteration(p,q);
    return 0;
}
```

Câu 36. Cho một chuỗi, hiện thực hàm `int strLen(char* str)` để tính độ dài của chuỗi sử dụng đệ quy.
Lưu ý: Không được sử dụng các từ khóa như `for`, `while`, `goto` (thậm chí là tên biến, comment). Trong bài tập này đã khai báo `#include <iostream>` và `using namespace std;`.

Câu 37. Hoàn thành hàm `recursiveSum` sử dụng đệ quy để tính và trả về tổng các phần tử trong một mảng.

- **Đầu vào:**
  - `int arr[]`: Mảng chứa các số nguyên cần tính tổng.
  - `int size`: Số lượng phần tử trong mảng.
- **Đầu ra:** `int` - Kết quả là tổng các phần tử trong mảng cho trước.

Lưu ý: Sinh viên không được sử dụng các từ khoá sau, kể cả trong tên biến và comment: `include`, `for`, `while`, `goto`.

Câu 38. Cho một số thập phân dương, hiện thực hàm `long int decimalToBinary(int decimal_number)` để chuyển đổi số thập phân dương đã cho thành số nhị phân tương đương.
Lưu ý: không được sử dụng từ khóa `for`, `while`, `goto` (ngay cả trong tên biến, comment). Đối với bài tập này, chúng ta có `#include <iostream>` và sử dụng `namespace std;`.

Câu 39. Palindrome là một chuỗi (string) mà nếu đảo ngược thứ tự các ký tự trong chuỗi ta vẫn có được chuỗi cũ. Ví dụ : "ABCBA", "RADAR", "otto", "i am ma i", "C". Hàm palindrome kiểm tra một chuỗi có là Palindrome đã được cho phía dưới.
Viết hàm `palindromeRecursion` sử dụng kỹ thuật gọi đệ quy để kiểm tra một chuỗi có phải là palindrome hay không.

- **Đầu vào:** Các chuỗi ký tự s có độ dài không quá 1000 ký tự
- **Đầu ra:** Mỗi dòng trả về giá trị của hàm `palindrome` và `palindromRecursion`.

Viết theo template sau:

```cpp
#include<iostream>
#include<string>
using namespace std;
/* END of library */
bool palindrome(string strg);
bool palindromeRecursion(string s)
{
    // BEGIN YOUR IMPLEMENTATION [1]
    // TODO

    // END YOUR IMPLEMENTATION [1]
}
int main()
{
    hiddenCheck();
    // BEGIN YOUR IMPLEMENTATION [2]
    // TODO
```

```
    // END YOUR IMPLEMENTATION [2]
    return 0;
}
```

Câu 40. Hiện thực hàm `findOccurrences` với ba đối số: một mảng số nguyên `nums`, độ dài của mảng `size` và một số nguyên `element`. Hàm sẽ trả về số lần xuất hiện của `element` trong `nums`. Ví dụ:

```
Input: nums = {1,2,3}, size = 3, element = 3
Output: 1
```

Lưu ý: Xin lưu ý rằng bạn không thể sử dụng từ khóa `for`, `while`, `goto` (ngay cả trong tên biến, comment). Bạn có thể triển khai các hàm đệ quy khác nếu cần. Đối với bài tập này, chúng ta có `#include <iostream>` và `using namespace std;`.

Câu 41. Cho một số nguyên x, thực hiện hàm `int countWaySumOfSquare(int x)` để tìm số cách biểu thị x thành tổng bình phương của các số nguyên dương duy nhất. Ví dụ:

```
Input : x = 100
Output : 3
```

Giải thích: $100 = 10^2 = 8^2 + 6^2 = 1^2 + 3^2 + 4^2 + 5^2 + 7^2$.
Lưu ý: Xin lưu ý rằng bạn không thể sử dụng từ khóa `for`, `while`, `goto` (ngay cả trong tên biến, comment). Bạn có thể triển khai các hàm đệ quy khác nếu cần. Đối với bài tập này, chúng ta có `#include <iostream>`, `#include <math.h>` và `using namespace std;`.

2 Pointer Basic

Câu 42. Hiện thực hàm `int getValueAtIndex(int *ptr, int k)` trả về giá trị của tại vị trí được chỉ định trong mảng qua con trỏ. Trong đó: `ptr` là con trỏ tới phần tử đầu tiên trong mảng, `k` là vị trí cần truy xuất phần tử (giá trị này không vượt quá độ dài của mảng).

Câu 43. Hiện thực hàm `void swap(int *ptr1, int *ptr2)` thực hiện hoán đổi giá trị tại vị trí của 2 con trỏ trỏ tới. Trong đó: `ptr1`, `ptr2` lần lượt là là các con trỏ trỏ tới vị trí cần hoàn đổi.

Câu 44. Hiện thực hàm `int calcSum(int *ptr, int n)` tính và trả về tổng của các phần tử trong mảng 1 chiều được cho bởi con trỏ. Trong đó: `ptr` là con trỏ tới phần tử đầu tiên trong mảng, `n` là kích thước của mảng.
Lưu ý: Bạn cần phải dùng dereference operator (`*`) để lấy giá trị của các phần tử trong mảng. Không được dùng subscript operator (`[]`).

Câu 45. Hiện thực hàm `void add(int *ptr, int n, int k)` thực hiện thêm phần tử vào cuối của mảng 1 chiều được cho bởi con trỏ. Trong đó: `ptr` là con trỏ tới phần tử đầu tiên trong mảng, `n`, `k` lần lượt là kích thước của mảng và phần tử cần được thêm vào.

Câu 46. Hiện thực hàm `int findMax(int *ptr, int n)` tìm và trả về phần tử lớn nhất trong mảng 1 chiều được cho bởi con trỏ. Trong đó: `ptr` là con trỏ tới phần tử đầu tiên trong mảng, `n` là kích thước của mảng.

Câu 47. Hiện thực hàm `void reverse(int *ptr, int n)` đảo ngược mảng 1 chiều được cho bởi con trỏ. Trong đó: `ptr` là con trỏ tới phần tử đầu tiên trong mảng, `n` là kích thước của mảng 1 chiều.
Lưu ý: Bạn cần phải dùng dereference operator (`*`) để lấy giá trị của các phần tử trong mảng. Không được dùng subscript operator (`[]`).

Câu 48. Hiện thực hàm `bool isSymmetry(int *head, int *tail)` kiểm tra mảng 1 chiều có phải là một mảng đối xứng hay không. Trong đó: `head`, `tail` lần lượt là con trỏ tới phần tử đầu tiên và cuối cùng trong mảng.

3 Pointer 2

Câu 49. Hãy hiện thực hàm `int* zeros(int n)` tạo một mảng có `n` phần tử 0.
- **Đầu vào:** Kích thước mảng `n`.
- **Đầu ra:** Con trỏ trỏ tới mảng vừa được cấp phát.

Lưu ý: Trong trường hợp cấp phát thất bại, hàm sẽ trả về `nullptr`.

Câu 50. Hãy hiện thực hàm `void shallowCopy(int*& newArr, int*& arr)` có chức năng tạo một bản sao của một mảng một chiều.
- **Đầu vào:** Mảng một chiều `arr` cần được sao chép.
- **Đầu ra:** Mảng đích một chiều `newArr` cần sao chép tới.

Lưu ý: Sau thực thi mảng được sao chép và mảng cần sao chép đều sử dụng chung một vùng nhớ.

Câu 51. Hãy hiện thực hàm `int** deepCopy(int** matrix, int r, int c)` trả về một bản sao của `matrix` gồm `r` hàng và `c` cột.
- **Đầu vào:** Con trỏ `matrix` trỏ đến mảng hai chiều có kích thước `r` x `c`.
- **Đầu ra:** Con trỏ trỏ đến mảng hai chiều được sao chép.

Lưu ý: Sau thực thi, con trỏ trả về phải trỏ đến vùng nhớ được cấp phát mới và khi `matrix` truyền vào có kích thước 0, hàm trả về `nullptr`.

Câu 52. Hãy hiện thực hàm `void deleteMatrix(int**& matrix, int r)` thực hiện giải phóng ô nhớ cho một mảng động 2 chiều có `r` hàng. `matrix` được gán bằng giá trị `NULL` sau khi thực hiện hàm.

Đầu vào: Mảng động hai chiều `matrix` có số hàng `r` cần giải phóng ô nhớ.

Câu 53. Cho một mảng động hai chiều `matrix` có kích thước `r` x `c`.
Hiện thực hàm `void insertRow(int**& matrix, int r, int c, int* rowArr, int row)` tiến hành chèn mảng `rowArr` (có kích thước `c`) vào hàng thứ `row` của mảng `matrix`.
- **Đầu vào:** Mảng 2 chiều `matrix` có kích thước `r` x `c`, hàng cần chèn `rowArr` và vị trí chèn `row`.
- **Đầu ra:** Mảng 2 chiều `matrix` sau khi được chèn.

Câu 54. Hãy hiện thực hàm `readArray()` được khai báo như sau: `int** readArray()`. Hàm này sẽ đọc dữ liệu cho một ma trận 2 chiều, mỗi chiều có 10 phần tử. Các phần tử của ma trận sẽ được nhập vào từ bàn phím (từ phần tử `a[0][0]` cho đến `a[9][9]`). Tuy nhiên nếu phần tử `a[i][j]` được nhập là 0 thì tất cả các phần tử còn lại trên hàng (`a[i][k]`, $j < k < 10$) đều được tự động gán là là 0, chương trình sẽ đọc tiếp phần tử `a[i+1][0]` từ bàn phím. Hàm `readArray` sẽ trả về một con trỏ tới mảng 2 chiều đã nhập này.
- **Đầu vào:** Các phần tử có trong mảng 2 chiều, mỗi phần tử là một số nguyên dương có giá trị không vượt quá 1000.
- **Đầu ra:** Con trỏ tới mảng 2 chiều vừa tạo.

Câu 55. Hiện thực hàm `void addElement(int*& arr, int n, int val, int index)` nhận vào một mảng động `arr` có chính xác `n` phần tử và tiến hành chèn giá trị `val` vào vị trí thứ `index`.
- **Đầu vào:** Mảng một chiều `arr` có kích thước `n`, giá trị cần chèn `val` và vị trí cần chèn `index`.
- **Đầu ra:** Mảng `arr` sau khi chèn.

Lưu ý: Việc chèn phần tử vào mảng động phải được thực hiện bằng cách giải phóng mảng cũ có n phần tử và cấp phát mảng mới có $n + 1$ phần tử.

Câu 56. Hiện thực hàm `int* flatten(int** matrix, int r, int c)` trả về một mảng một chiều được "làm phẳng" từ mảng hai chiều có kích thước `r` x `c` (bằng cách nối các hàng của mảng hai chiều lại với nhau).

- **Đầu vào:** Mảng hai chiều có kích thước `r` x `c`.
- **Đầu ra:** Mảng một chiều sau khi được "làm phẳng" từ mảng hai chiều đầu vào.

Câu 57. Hiện thực hàm `char* concatStr(char* str1, char* str2)` trả về một chuỗi là kết quả sau khi nối 2 chuỗi `str1` và `str2` thành một chuỗi duy duy nhất.

- **Đầu vào:** Hai chuỗi `str1` và `str2`.
- **Đầu ra:** Chuỗi được nối từ 2 chuỗi con `str1` và `str2`.

Lưu ý: Không được phép sử dụng các hàm hỗ trợ của thư viện `string` và `string.h` cho bài tập này.

Câu 58. Chuyển vị của một ma trận 2 chiều là một phần quan trọng trong việc tính toán trên ma trận nói riêng và đại số tuyến tính nói chung. Gọi B là ma trận sau khi chuyển vị của ma trận A thì ma trận B có tính chất là $b[i][j] = a[j][i]$. Hãy viết hàm `int** transposeMatrix(int** matrix, int r, int c)` thực hiện phép chuyển vị trên ma trận đã được đề cập bên trên.

- **Đầu vào:**
  - Con trỏ tới mảng 2 chiều. Mỗi phần tử trong mảng 2 chiều có giá trị trong khoảng $(-1000; 1000)$.
  - Kích thước mảng 2 chiều là 1 cặp số dương `r`, `c`. Trong đó: `r` là số hàng của ma trận, `c` là số cột của ma trận. Giá trị `n` không vượt quá 1000.
- **Đầu ra:** Con trỏ trỏ tới mảng hai chiều sau khi được chuyển vị. Trong trường hợp ma trận đầu vào rỗng, trả về con trỏ `null`.

Câu 59. Cho một mảng động hai chiều `matrix` có kích thước `r` x `c`. Hiện thực hàm `int** insertCol(int**& matrix, int r, int c, int* colArr, int col)` tiến hành chèn mảng `rowArr` (có kích thước `r`) vào cột thứ `col` của `matrix`.

- **Đầu vào:** Mảng 2 chiều `matrix` có kích thước `r` x `c`, cột cần chèn `colArr` và vị trí chèn `col`.
- **Đầu ra:** Mảng 2 chiều `matrix` sau khi được chèn, đầu ra phải được điều chỉnh trên biến `matrix` truyền vào.

Câu 60. Cho một mảng động hai chiều `matrix` có kích thước `r` x `c`. Hiện thực hàm `bool deleteRow(int**& matrix, int r, int c, int row)` tiến hành xóa hàng thứ `row` của `matrix`.

- **Đầu vào:** Mảng 2 chiều `matrix` có kích thước `r` x `c` và vị trí hàng cần xóa `row`.
- **Đầu ra:** Mảng 2 chiều `matrix` sau khi xóa hàng và kết quả xóa thành công hay không.

Lưu ý: Nếu mảng hai chiều xóa đi hàng cuối cùng, mà trận được truyền vào hàm sẽ trả về giá trị `nullptr` và đồng thời giải phóng toán bộ vùng nhớ của mảng hai chiều được cấp phát trước đó.

Câu 61. Cho một mảng động hai chiều `matrix` có kích thước `r` x `c`. Hiện thực hàm `bool deleteCol(int** matrix, int r, int c, int col)` tiến hành xóa cột thứ `col` của `matrix`.

- **Đầu vào:** Mảng 2 chiều `matrix` có kích thước `r` x `c` và vị trí cột cần xóa `col`.
- **Đầu ra:** Mảng 2 chiều `matrix` sau khi xóa cột.

Lưu ý: Nếu mảng hai chiều xóa đi cột cuối cùng, kết quả trả về là `nullptr` và đồng thời giải phóng toán bộ vùng nhớ của mảng hai chiều được cấp phát trước đó.

4 Struct

Câu 62. Kho lưu trữ của tổ chức SCP chứa hàng loạt các vật thể dị thường. Mỗi vật thể dị thường được lưu trữ dưới struct `SCP` với các thông tin như sau:

- `id`: kiểu `int`, là mã định danh (hay mã vật thể), phân biệt giữa các vật thể với nhau.
- `objClass`: kiểu `int`, là phân loại của vật thể đó.
- `speConProcedures`: kiểu `string`, mô tả quy trình quản thúc đặc biệt của vật thể đó.
- `description`: kiểu `string`, mô tả về các đặc điểm của vật thể đó.
- `addendums`: kiểu `string*`, là một tập hợp của các phụ lục đính kèm, mô tả các thông tin bổ sung cho vật thể đó.

- `numAddendums`: kiểu `int`, là số lượng phụ lục đính kèm.

Hiện thực struct `SCP` với các yêu cầu trên.
Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

Câu 63. Cho struct `SCP` lưu trữ thông tin các vật thể dị thường được mô tả như sau:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

Hiện thực hàm `void addAddendum(SCP &obj, string addendum)`. Hàm thực hiện bổ sung một phụ lục `addendum` vào cuối danh sách phụ lục (`addendums`) của `obj`.
Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

Câu 64. Cho struct `SCP` lưu trữ thông tin các vật thể dị thường được mô tả như sau:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

Hiện thực hàm với prototype sau:

```
SCP createSCP(int id, int objClass, string speConProcedures, string description, string*
    addendums, int numAddendums);
```

Hàm có chức năng khởi tạo `instance` của struct `SCP` với các thông tin đã truyền vào, sau đó trả về `instance` đó.
Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

Câu 65. Cho struct `SCP` lưu trữ thông tin các vật thể dị thường được mô tả như sau:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

Hiện thực hàm `string convertToString(SCP obj)`. Hàm trả về một `string` duy nhất, với format chuẩn được quy định bởi tổ chức:

```
Item #: SCP-<Mã định danh, thêm 0 vào trước nếu chưa đủ 3 chữ số>
```

```
Object Class: <Safe nếu objClass là 0, Euclid nếu là 1, Keter nếu là 2>
Special Containment Procedures: <Quy trình quản thúc đặc biệt>
Description: <Mô tả vật thể>
<Phụ lục 0>
<Phụ lục 1>
...
<Phụ lục n>
```

Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.
Ví dụ:
Test:

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the
    hypothesis that SCP-055 was never formally acquired.";
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE
    was successfully able to enter SCP-055's container and ascertain the appearance.";
SCP obj {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5)
    meter square room.", "All known information about the object is that it is not round.
    ", addendums, 2};
cout << convertToString(obj);
delete [] addendums;
```

Expected:

```
Item #: SCP-055
Object Class: Keter
Special Containment Procedures: Object is kept within a five (5) by five (5) by
↪  two point five (2.5) meter square room.
Description: All known information about the object is that it is not round.
Document #055-1: An Analysis of SCP-055
The author puts forward the hypothesis that SCP-055 was never formally acquired.
Document #055-2: Report of Dr. John Marachek
Survey team #19-055-127BXE was successfully able to enter SCP-055's container and
↪  ascertain the appearance.
```

Câu 66. Cho struct `SCP` lưu trữ thông tin các vật thể dị thường được mô tả như sau:

```
1 struct SCP {
2     int id;
3     int objClass;
4     string speConProcedures;
5     string description;
6     string* addendums;
7     int numAddendums;
8 };
```

Một đối tượng dị thường là SCP-038 có khả năng tạo bản sao của các vật thể dị thường khác, ví dụ như SCP-500 (thuốc vạn ứng). Khi đối tượng gốc bị tiêu thụ/phá hủy, bản sao vẫn tồn tại độc lập. Hiện thực hàm `SCP* cloneSCP(SCP* original)`. Hàm có chức năng tạo một bản sao con trỏ loại SCP chứa các thông tin như đối tượng gốc original, sau đó trả về con trỏ đó. Lưu ý về shallow copy và deep copy.
Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

Câu 67. Cho struct `SCP` lưu trữ thông tin các vật thể dị thường được mô tả như sau:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

Chỉ số `objClass` của mỗi vật thể dị thường đánh giá mức độ khó khăn trong việc quản thúc đối tượng. Chỉ số này càng cao, đối tượng càng khó bị quản thúc, và việc quản thúc cần tiêu tốn nhiều tài nguyên hơn.

Hiện thực hàm `int compareObjectClass(const SCP &objA, const SCP &objB)`. Hàm so sánh mức độ khó khăn trong việc quản thúc của hai đối tượng `objA` và `objB` và trả về `-1` nếu `objA` dễ quản thúc hơn `objB`, 1 nếu `objA` khó quản thúc hơn `objB`, 0 nếu mức độ khó khăn trong việc quản thúc của hai vật thể này tương đương nhau.

Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

Chương III

LAB 3

1 Linked list

Câu 68. Cho chương trình khởi tạo, trong đó:

- `struct node`: đại diện cho một node của linked list.
- Hàm `createLinkedList`:
  - Nhận vào số phần tử (> 0) của linked list
  - Xây dựng một linked list với dữ liệu của các node được nhập từ đầu vào chuẩn (`stdin`), node mới được thêm vào *CUỐI* linked list
  - Trả về con trỏ đến node đầu tiên của linked list.
- Hàm `isEqual` so sánh hai linked list:
  - Nhận vào con trỏ `head1` của linked list thứ nhất, con trỏ của một `head2` của linked list thứ 2
  - Hàm trả về `true` khi và chỉ khi hai linked list có cùng chiều dài và giá trị của node ở vị trí tương ứng luôn bằng nhau. Ngược lại, hảm trả về `false`.
- Hàm `main` đọc vào số phần tử của linked list, gọi hàm `createLinkedList` để khởi tạo linked list, sau đó gọi hàm `print` để in ra các phần tử của linked list.

Bạn hãy hoàn thành hàm `isEqual`.

Đầu vào: Các giá trị nhập vào từ đầu vào chuẩn (`stdin`) đều có giá trị trong khoảng $(0; 5000)$.

```cpp
#include <iostream>
using namespace std;
struct node
{
  int data;
  node *next;
};

node *createLinkedList(int n); // The implementation is provided implicitly

bool isEqual(node *head1, node *head2)
{
   //TODO
}

int main()
{
  int n = 0;
  cin>> n;
  node *head1 = createLinkedList(n);
```

```
  int m = 0;
  cin>> m;
  node *head2 = createLinkedList(m);
  cout << isEqual(head1, head2) << endl;
  return 0;
}
```

Câu 69. Cho chương trình khởi tạo, trong đó:

- `struct node`: đại diện cho một node của linked list
- Hàm `print`: nhận vào con trỏ head của linked list và in ra từng phần tử của linked list
- Hàm `createLinkedList`:
  - Nhận vào số phần tử (> 0) của linked list
  - Xây dựng một linked list với dữ liệu của các node được nhập từ đầu vào chuẩn (`stdin`), node mới được thêm vào *CUỐI* linked list
  - Trả về con trỏ đến node đầu tiên của linked list.
- Hàm `countNode`:
  - Nhận đầu vào là con trỏ head của linked list
  - Trả về số lượng nodes trong linked list
- Hàm `main` đọc vào số phần tử của linked list, gọi hàm `createLinkedList` để khởi tạo linked list, sau đó gọi hàm `print` để in ra các phần tử của linked list.

Bạn hãy hoàn thành hàm `countNode`.

- **Đầu vào:**
  - Số `n` là số lượng phần tử trong linked list ($0 < n < 5000$)
  - n số tiếp theo là giá trị của mỗi node trong linked list, giá trị là một số nguyên có giá trị trong khoảng $(-5000; 5000)$.
- **Đầu ra:** Thỏa yêu cầu bài toán.

```
#include <iostream>
using namespace std;
struct node
{
  int data;
  node *next;
};

node *createLinkedList(int n); // The implementation is provided implicitly

int countNode(node* head)
{
    //TODO
}

void print(node *head)
{
  while (head != nullptr)
  {
    cout << head->data << endl;
    head = head->next;
  }
}
int main()
{
```

```
  int n = 0;
  cin >> n;
  node *head = createLinkedList(n);
  print(head);
  cout<<endl;
  cout<<countNode(head);
  return 0;
}
```

Câu 70. Cho chương trình khởi tạo, trong đó:

- `struct node`: đại diện cho một node của linked list
- Hàm `print`: nhận vào con trỏ head của linked list và in ra từng phần tử của linked list
- Hàm `createLinkedList`:
  - Nhận vào số phần tử (> 0) của linked list
  - Xây dựng một linked list với dữ liệu của các node được nhập từ đầu vào chuẩn (`stdin`), node mới được thêm vào $\hat{DAU}$ linked list
  - Trả về con trỏ đến node đầu tiên của linked list.
- Hàm `countNode`:
  - Nhận đầu vào là con trỏ head của linked list
  - Trả về số lượng nodes trong linked list
- Hàm `main` đọc vào số phần tử của linked list, gọi hàm `createLinkedList` để khởi tạo linked list, sau đó gọi hàm `print` để in ra các phần tử của linked list.

Bạn hãy hoàn thành hàm `createLinkedList`.

- **Đầu vào:**
  - Số `n` là số lượng phần tử trong linked list ($0 < n < 5000$)
  - n số tiếp theo là giá trị của mỗi node trong linked list, giá trị là một số nguyên có giá trị trong khoảng $(-5000; 5000)$.
- **Đầu ra:** Thỏa yêu cầu bài toán.

```
#include <iostream>
using namespace std;
struct node
{
  int data;
  node *next;
};
node *createLinkedList(int n)
{
    //TODO
}
void print(node *head)
{
  while (head != nullptr)
  {
    cout << head->data << endl;
    head = head->next;
  }
}
int main()
{
  int n = 0;
  cin >> n;
```

```
  if (n > 0)
  {
    node *head = createLinkedList(n);
    print(head);
  }
  else
  {
    cout << "Invalid n" << endl;
  }
  return 0;
}
```

Câu 71. Cho chương trình khởi tạo, trong đó:

- `struct node`: đại diện cho một node của linked list
- Hàm `print`: nhận vào con trỏ head của linked list và in ra từng phần tử của linked list
- Hàm `createLinkedList`:
  - Nhận vào số phần tử (> 0) của linked list
  - Xây dựng một linked list với dữ liệu của các node được nhập từ đầu vào chuẩn (`stdin`), node mới được thêm vào *CUỐI* linked list
  - Trả về con trỏ đến node đầu tiên của linked list.
- Hàm `countNode`:
  - Nhận đầu vào là con trỏ head của linked list
  - Trả về số lượng nodes trong linked list
- Hàm `main` đọc vào số phần tử của linked list, gọi hàm `createLinkedList` để khởi tạo linked list, sau đó gọi hàm `print` để in ra các phần tử của linked list.

Bạn hãy hoàn thành hàm `createLinkedList`.

- **Đầu vào:**
  - Số `n` là số lượng phần tử trong linked list ($0 < n < 5000$)
  - n số tiếp theo là giá trị của mỗi node trong linked list, giá trị là một số nguyên có giá trị trong khoảng $(-5000; 5000)$.
- **Đầu ra:** Thỏa yêu cầu bài toán.

```
#include <iostream>
using namespace std;
struct node
{
  int data;
  node *next;
};
node *createLinkedList(int n)
{
    //TODO
}
void print(node *head)
{
  while (head != nullptr)
  {
    cout << head->data << endl;
    head = head->next;
  }
}
int main()
```

```
{
  int n = 0;
  cin >> n;
  if (n > 0)
  {
    node *head = createLinkedList(n);
    print(head);
  }
  else
  {
    cout << "Invalid n" << endl;
  }
  return 0;
}
```

Câu 72. Cho chương trình khởi tạo, trong đó:

- `struct node`: đại diện cho một node của linked list
- Hàm `print`: nhận vào con trỏ head của linked list và in ra từng phần tử của linked list
- Hàm `createLinkedList`:
  - Nhận vào số phần tử (> 0) của linked list
  - Xây dựng một linked list với dữ liệu của các node được nhập từ đầu vào chuẩn (`stdin`), node mới được thêm vào *CUỐI* linked list
  - Trả về con trỏ đến node đầu tiên của linked list.
- Hàm `insertNode`:
  - Nhận vào con trỏ `head` của linked list, con trỏ của một node mới, và vị trí `position`.
  - Hàm sẽ chèn node mới này vào vị trí `position` (lấy vị trí của node `head` là 1). Nếu `position<=0`, hàm không làm gì cả. Nếu `position` lớn hơn số phần tử hiện tại của linked list thì node mới được chèn vào cuối của linked list.
- Hàm `main` đọc vào số phần tử của linked list, gọi hàm `createLinkedList` để khởi tạo linked list, sau đó gọi hàm `print` để in ra các phần tử của linked list.

Bạn hãy hoàn thành hàm `insertNode`.

Đầu vào: Các giá trị nhập vào từ đầu vào chuẩn (`stdin`) đều có giá trị trong khoảng $(0; 5000)$.

```
#include <iostream>
using namespace std;
struct node
{
  int data;
  node *next;
};

node *createLinkedList(int n); // The implementation is provided implicitly

void replace(node* head, int position, int value)
{
  //TODO
}
void print(node *head)
{
  while (head != nullptr)
  {
    cout << head->data << " ";
    head = head->next;
```

```
  }
  cout<<endl;
}
int main()
{
  int n = 0;
  cin >> n;
  node *head = createLinkedList(n);
  print(head);
  int pos, val;
  cin>>pos>>val;
  replace(head, pos, val);
  print(head);
  return 0;
}
```

Câu 73. Cho chương trình khởi tạo, trong đó:

- `struct node`: đại diện cho một node của linked list
- Hàm `print`: nhận vào con trỏ head của linked list và in ra từng phần tử của linked list
- Hàm `createLinkedList`:
  - Nhận vào số phần tử (> 0) của linked list
  - Xây dựng một linked list với dữ liệu của các node được nhập từ đầu vào chuẩn (`stdin`), node mới được thêm vào *CUỐI* linked list
  - Trả về con trỏ đến node đầu tiên của linked list.
- Hàm `searchLinkedList`:
  - Nhận vào con trỏ `head` của linked list, giá trị cần tìm.
  - Nếu tìm thấy thì trả về vị trí đầu tiên của nó trong Linked List (`index` từ 0), không thì trả về `-1`.
- Hàm `main` đọc vào số phần tử của linked list, gọi hàm `createLinkedList` để khởi tạo linked list, sau đó gọi hàm `print` để in ra các phần tử của linked list.

Bạn hãy hoàn thành hàm `searchLinkedList`.

```
#include <iostream>
using namespace std;
struct node
{
  int data;
  node *next;
};

node *createLinkedList(int n); // The implementation is provided implicitly

int searchLinkedList(node* head, int key)
{
  // TODO
}
void print(node *head)
{
  while (head != nullptr)
  {
    cout << head->data << endl;
    head = head->next;
  }
```

```
}
int main()
{
  int n = 0;
  cin >> n;
  node *head = createLinkedList(n);
  print(head);
  int m;
  cin>>m;
  cout<<searchLinkedList(head,m);
  return 0;
}
```

2 OOP

Câu 74. Hãy thiết kế một class `Integer` với một biến private `val` dạng `int`. Class gồm các phương thức sau:

- constructor nhận biến `val` ban đầu.
- `setValue()` để thay đổi giá trị biến `val`.
- `getValue()` để lấy giá trị biến `val`.

Câu 75. Trong thân hàm `main()`, hãy viết chương trình chính tạo ra một mảng các đối tượng thuộc class `Course` có `n` phần tử (`n` được nhập từ người sử dụng – sử dụng kỹ thuật tạo mảng động bằng lệnh `new`). Sau đó nhập và hiển thị dữ liệu cho `n` đối tượng này bằng hai phương thức `getinfo()` và `disinfo()` đã được hiện thực sẵn.

```
#include<iostream>

using namespace std;

class Course {
    private:
        int ID;
        int numOfStudent;
        int numOfTeacher;
        int numOfTA;
    public:
        void getinfo();
        void disinfo();
};

void Course::getinfo() {
   cout << "ID: ";
   cin >> ID;
   cout << "Number of Students: ";
   cin >> numOfStudent;
   cout << "Number of Teachers: ";
   cin >> numOfTeacher;
   cout << "Number of TAs: ";
   cin >> numOfTA;
}

void Course::disinfo()
```

```
{
   cout<<endl;
   cout<< "CourseID = "<< ID << endl;
   cout<< "Number of student = " << numOfStudent << endl;
   cout<< "Number of teacher = " << numOfTeacher << endl;
   cout<< "Number of TA = " << numOfTA<< endl;
}


int main() {
    //TODO
}
```

Câu 76. Cho class template `Array` như bên dưới dùng để chứa một mảng (1 chiều) được khai báo trong vùng nhớ `Heap`. Trong class `Array` có khai báo một số phương thức (hàm) để thao tác với `Array`.

```
template <typename T>
class Array {
public:
    Array(int size, T initValue);
    ~Array();
    void print();
   void setAt(int idx, const T & value);
   T getAt(int idx);
   T& operator[](int idx);
private:
    int size;
    T * p;
};
```

Trong class `Array` có khai báo các thuộc tính sau:

- Thuộc tính `p` là con trỏ trỏ đến vùng nhớ `Heap` được cấp phát.
- Thuộc tính `size` của `Array` chứa số lượng phần tử của mảng.

SV hiện thực phương thức `operator[](int idx)`: quá tải toán tử `[]` để thực hiện được cả 2 việc: đọc (lấy) giá trị của phần tử ở vị trí `idx` và ghi (gán) một giá trị mới vào phần tử ở vị trí `idx`. Ví dụ: đọc: `x = a[2]`, ghi: `a[2] = 5` với a là một đối tượng của class Array. Nếu `idx` có giá trị không hợp lệ (`idx < 0` hoặc `idx >= size`) thì throw `-1`.

Câu 77. Hãy thiết kế một class `Room` với constructor gồm 3 biến `length`, `breadth`, `height` (theo thứ tự). Hiện thực các phương thức sau cho class `Room`:

- Constructor: đầu vào lần lượt là `length`, `breadth`, `height`.
- `calculateArea`: tính diện tích của căn phòng.
- `calculateVolume`: tính thể tích của căn phòng.

Câu 78. Hãy xem xét câu lệnh: `ClockType myClock (5, 12, 40);`. Câu lệnh này khai báo một đối tượng `myClock` thuộc class `ClockType`. Ở đây, chúng ta đang truyền ba giá trị kiểu `int`, giá trị này khớp với kiểu của các tham số chính thức của hàm tạo với một tham số.

- Xem xét đoạn code cho trước trong phần trả lời, chú ý đến hàm khởi tạo có 3 tham số. Hãy hiện thực hàm này để sau khi gọi câu lệnh khai báo trên, 3 biến thành viên được lần lượt đặt thành 5, 12, 40.
- Hiện thực hàm khởi tạo với không tham số, hàm sẽ gán giá trị 0 cho 3 biến thành viên.

Lưu ý: `hr`, `min`, `sec` cần thoả mãn các điều kiện sau: `0 <= hr < 24`; `0 <= min < 60`; `0 <= sec < 60`. Nếu tham số đầu vào không thoả mãn điều kiện thì ta gán giá trị 0 cho biến thành viên tương ứng.

```
class ClockType
{
   public:
      ClockType(int, int, int); //constructor with parameters
      ClockType(); //default constructor

      void printTime() const;

   private:
      int hr;
      int min;
      int sec;
};

void ClockType::printTime() const {
    if (hr < 10)
        cout << "0";
    cout << hr << ":";
    if (min < 10)
        cout << "0";
    cout << min << ":";
    if (sec < 10)
        cout << "0";
    cout << sec;
}

//TODO
```

Câu 79. Cho định nghĩa class:

```
class ClockType
{
   public:
      ClockType();
      void setTime(int, int, int);
      void printTime() const;
   private:
      int hr;
      int min;
      int sec;
};
```

Cho câu lệnh: `myClock.setTime(5, 2, 30);`. Trong câu lệnh này, phương thức `setTime` được thực thi. Các giá trị 5, 2 và 30 được chuyển dưới dạng tham số cho hàm `setTime` và hàm sử dụng các giá trị này để đặt giá trị của ba biến thành viên `hr`, `min` và `sec` của đối tượng `myClock` thành 5, 2 và 30, tương ứng. SV hiện thực phương thức `setTime` để nó thực hiện được mô tả trên.
Lưu ý: `hr`, `min`, `sec` cần thoả mãn các điều kiện sau: `0 <= hr < 24`; `0 <= min < 60`; `0 <= sec < 60`. Nếu tham số đầu vào không thoả mãn điều kiện thì ta gán giá trị 0 cho biến thành viên tương ứng.

```
class ClockType
{
public:
    ClockType();
```

```
    void setTime(int, int, int);
    void printTime() const;

private:
    int hr;
    int min;
    int sec;
};

void ClockType::printTime() const
{
   if (hr < 10)
      cout << "0";
   cout << hr << ":";
   if (min < 10)
      cout << "0";
   cout << min << ":";
   if (sec < 10)
      cout << "0";
   cout << sec;
}

// TODO
```

Câu 80. Cho định nghĩa class:

```
class ClockType
{
   public:
      void setTime(int, int, int);
      void getTime(int&, int&, int&) const;
      void printTime() const;
      clockType(int, int, int); //constructor with parameters
      clockType(); //default constructor
   private:
      int hr;
      int min;
      int sec;

};
```

Cho đoạn code sau:

```
clockType myClock;
int hours;
int minutes;
int seconds;
myClock.getTime(hours, minutes, seconds);
cout << "hours = " << hours << ", minutes = " << minutes << ", seconds = " << seconds <<
    endl;
```

Trong câu lệnh thứ 5, hàm thành viên `getTime` được thực thi. Các giá trị `hr`, `min` và `sec` của `myClock` được hàm sử dụng để cài đặt giá trị của ba biến `hours`, `minutes`, `seconds` tương ứng. SV hiện thực phương thức `getTime` để thực hiện được mô tả như trên.

Lưu ý: `hr`, `min`, `sec` cần thoả mãn các điều kiện sau: `0 <= hr < 24`; `0 <= min < 60`; `0 <= sec < 60`. Nếu tham số đầu vào không thoả mãn điều kiện thì ta gán giá trị 0 cho biến thành viên tương ứng.

Câu 81. Hãy xem xét định nghĩa sau:

```
template < class T >
class Cell {
      protected:
          T info;
      public:
          void set(T x){ info = x; }
          T get() { return info; }
   };
enum Color {White, Yellow, Black, Red, Blue};
```

Định nghĩa một lớp con `ColoredCell` của `Cell` với:

1. Với một biến color.
2. Hàm `setColor(Color)` để cài đặt màu cho ô
3. Hàm `getColor()` để lấy màu của ô (trả về kiểu dữ liệu của màu).
4. Cập nhật method `get` để trả về con trỏ đến `info` nếu ô này không trắng, ngược lại trả về `NULL`.

Câu 82. Thực hiện các yêu cầu sau:

- Xây dựng một class `Integer` gồm một thành phần private tên là `val` là một số nguyên kiểu `int`.
- Hiện thực constructor: `Integer(int)`.
- Hiện thực constructor: `Integer(Integer*)`.
- Quá tải toán tử `operator +` để phép tính `Integer(2) + Integer(3)` trả về `Integer(5)`.
- Quá tải toán tử `operator +` để phép tính `Integer(3) + 2` trả về `Integer(5)`.

Chú ý: khai báo phương thức `print()` trong định nghĩa của class và không cần hiện thực.

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

// TODO

void Integer::print() {
    cout << this->val << endl;
}

int main() {
    int x, y, z;
    cin >> x >> y >> z;
    Integer a(x);
    Integer b(y);
    Integer* t = new Integer(z);
    Integer c(t);

    a.print(); b.print();
    (a + b + c + 4).print();

    delete t;
    return 0;
}
```

Câu 83. Cho class `ClockType` như sau:

```
class ClockType
{
   public:
      void printTime() const;
      void incrementSeconds();
      void incrementMinutes();
      void incrementHours();
      clockType(int, int, int);
      clockType();
   private:
      int hr;
      int min;
      int sec;
};
```

Hãy xem xét câu lệnh: `myClock.incrementSeconds();`. Câu lệnh này tăng biến `sec` thêm 1 đơn vị. Hãy hiện thực 3 phương thức:

- `incrementHours`: tăng `hr` thêm 1 đơn vị. Sau khi tăng, nếu `hr = 24` thì ta đặt lại `hr = 0`.
- `incrementMinutes`: tăng `min` thêm 1 đơn vị. Sau khi tăng, nếu `min = 60` thì ta đặt lại `min = 0` và tăng hr thêm 1 đơn vị bằng cách phù hợp.
- `incrementSeconds`: tăng sec thêm 1 đơn vị. Sau khi tăng, nếu `sec = 60` thì ta đặt lại `sec = 0` và tăng `min` thêm 1 đơn vị bằng cách phù hợp.

Câu 84. Cho class template `Array` như bên dưới dùng để chứa một mảng (1 chiều) được khai báo trong vùng nhớ Heap. Trong class `Array` có khai báo một số phương thức (hàm) để thao tác với `Array`.

```
template <typename T>
class Array {
public:
    Array(int size, T initValue);
    ~Array();
    Array(const Array<T> & other); // Copy constructor
    Array<T> & operator=(const Array<T> & other); // Copy assignment operator
private:
    int size;
    T * p;
};
```

Trong class `Array` có khai báo các thuộc tính sau:

- Thuộc tính `p` là con trỏ trỏ đến vùng nhớ Heap được cấp phát.
- Thuộc tính `size` của `Array` chứa số lượng phần tử của mảng.

Yêu cầu: Hiện thực hàm Copy Constructor và Copy Assignment operator:

- Hàm `Array(const Array<T> & other)`: copy constructor, khởi tạo đối tượng mới dựa trên dữ liệu của đối tượng được cung cấp (other). Đồng thời, in ra thông báo: `"Call copy constructor"`.
- Hàm `operator=`: copy assignment operator, gán giá trị của đối tượng hiện tại bằng giá trị của đối tượng được cung cấp. Đồng thời, in ra thông báo: `"Call assignment operator"`.

Câu 85. Cho class template `Array` như bên dưới dùng để chứa mảng 1 chiều được khai báo trong vùng nhớ `Heap`.

```
template <typename T>
class Array {
```

```
3 public:
4     Array(int size, T initValue);
5     ~Array();
6 private:
7     int size;
8     T * p;
9 };
```

Trong class `Array` có khai báo các thuộc tính sau:

- Thuộc tính `p` là con trỏ trỏ đến vùng nhớ `Heap` được cấp phát.
- Thuộc tính `size` của `Array` chứa số lượng phần tử của mảng.

Yêu cầu:

- SV hiện thực 2 phương thức được mô tả như sau:
  1. Hàm `Array(int size, T initValue)`: hàm khởi tạo (constructor), gán size vào số lượng phần tử của mảng; khởi tạo mảng 1 chiều có kích thước là `size` trong vùng nhớ `Heap` và lưu địa chỉ phần tử đầu tiên của mảng vào biến `p`.
  2. Hàm `~Array()`: hàm huỷ, thu hồi vùng nhớ Heap đã cấp phát.
- SV thực hiện việc khai báo phương thức `print` (không định nghĩa) cho class `Array`.

Câu 86. Cho class template `Array` như bên dưới dùng để chứa mảng 1 chiều được khai báo trong vùng nhớ `Heap`.

```
1  template <typename T>
2  class Array {
3  public:
4      Array(int size, T initValue);
5      ~Array();
6      Array(const Array<T>& other);
7  private:
8      int size;
9      T * p;
10 };
```

Trong class Array có khai báo các thuộc tính sau:

- Thuộc tính `p` là con trỏ trỏ đến vùng nhớ Heap được cấp phát.
- Thuộc tính `size` của `Array` chứa số lượng phần tử của mảng.

Yêu cầu: SV hiện thực phương thức Copy Constructor (dòng thứ 6), phương thức này khởi tạo một đối tượng Array mới dựa trên một đối tượng Array khác.

Câu 87. Cho định nghĩa class:

```
1  class clockType
2  {
3     public:
4        void setTime(int, int, int);
5        bool equalTime(const clockType&) const;
6        clockType(int, int, int);
7        clockType();
8     private:
9        int hr;
10       int min;
11       int sec;
12 };
```

Hãy xem xét câu lệnh:

```
if (myClock.equalTime(yourClock))
    cout << "Both times are equal." << endl;
else
    cout << "The two times are not equal." << endl;
```

Biểu thức kiểm tra điều kiện if-else so sánh `myClock` và `yourClock`. Nếu thời gian của 2 clock gồm (hr, min, sec) là giống nhau thì trả về `true`, ngược lại sẽ trả về `false`.
SV hiện thực phương thức `equalTime` để thực hiện như mô tả trên.

Câu 88. Cho class như sau:

```
class Book {
public:
    Book(const char*);
    ~Book();
    void display();
private:
    char* name;
};
```

Hiện thực constructor và destructor của `Book` sao cho chương trình sau không bị lỗi khi chạy:

```
Book * pBook = new Book("To Kill a Mockingbird");
pBook->display();
delete pBook;
```

Kết quả in ra: `Book: To Kill a Mockingbird`.

Chương IV

MỘT SỐ CÂU TRONG CÁC ĐỀ FINAL LAB

Câu 89. Cho mảng động hai chiều `matrix` có kích thước `r` x `c`.
Hiện thực hàm `bool deleteCol(int** matrix, int r, int c, int col)` tiến hành xoá cột thứ `col` của `matrix`.

- **Đầu vào:** Mảng 2 chiều `matrix` có kích thước `r` x `c` và vị trí cột cần xoá `col`.
- **Đầu ra:** Mảng 2 chiều `matrix` sau khi xoá cột.

Lưu ý: Nếu mảng hai chiều xoá đi cột cuối cùng, kết quả trả về là `nullptr` và đồng thời giải phóng toàn bộ vùng nhớ của mảng hai chiều được cấp phát trước đó.

Câu 90. Cho định nghĩa `struct Node` và `struct LList` như sau.

```
struct Node{
    int data;
    Node* next;
};
struct LList{
    Node* head;
};
```

Hiện thực hai hàm `void greatestOfThree(const LList& list, int& first, int& second, int& third);` và `int isSublist(const LList& list1, const LList& list2);` với `list`, `list1`, và `list2` là các danh sách liên kết đơn.

a) Hàm `greatestOfThree` tìm bộ ba số liên tiếp trong một danh sách có tổng cao nhất trong các bộ ba số liên tiếp. Ba số này lần lượt được gán vào các tham số `first`, `second`, `third`. Nếu có nhiều bộ cùng thoả mãn, hàm trả về bộ số xuất hiện cuối cùng. Nếu không có bộ nào thoả mãn, hàm trả về `-1`.

b) Hàm `isSublist` kiểm tra xem một danh sách có phải là con của danh sách còn lại hay không, tức là, các phần tử trong danh sách này có nằm trong danh sách kia không.
Ví dụ:
- Nếu `list1={1,2,3,1,2,4}` và `list2={2,3,1}`, thì `list2` là danh sách con của `list1`.
- Nếu `list1={4,2,1}` và `list2={1,2,3,1,2,4}`, thì `list1` là danh sách con của `list2`.

Hàm trả về độ dài của danh sách con. Nếu không có danh sách con, hàm trả về `-1`.

Chú ý:

- Không được sử dụng mảng và toán tử `[]`.
- Một danh sách rỗng là danh sách con của mọi danh sách.

Câu 91. Class `LLNode` dùng để lưu trữ một node trong một danh sách liên kết đơn. Class `LLNode` được định nghĩa như sau, với `val` là giá trị của node, `next` là con trỏ trỏ đến node tiếp theo:

```
class LLNode{
    public:
        int val;
        LLNode* next;
        LLNode();
        LLNode(int val, LLNode* next);
}
```

Hiện thực hàm `LLNode* addLinkedList(LLNode* l0, LLNode* l1);`, với `l0`, `l1` là hai danh sách liên kết đơn biểu diễn các số nguyên dương, mỗi node là một chữ số, với phần tử đầu là chữ số hàng đơn vị (giá trị của mỗi node nằm trong đoạn $[0; 9]$, độ dài của mỗi danh sách liên kết nằm trong đoạn $[0; 100000]$). Hàm này trả về danh sách liên kết biểu diễn tổng của hai số nguyên.
Ví dụ: Cho `l0=[2,3]` (biểu diễn số 32), và `l1=[1,8]` (biểu diễn số 81). Kết quả là `l0=[3,1,1]` (biểu diễn $32 + 81 = 113$).

Chú ý: Ở bài tập này, các thư viện `iostream`, `string`, `cstring`, `climits`, `utility`, `vector`, `list`, `stack`, `queue`, `map`, `unordered_map`, `set`, `unordered_set`, `functional`, `algorithm` đã được thêm vào, và `namespace std` đã được sử dụng. Sinh viên có thể, nhưng không khuyến khích, thêm vào và sử dụng các thư viện khác được cho phép.

Câu 92. Kỉ nguyên Đại Hải Tặc

Giàu sang, danh vọng, và quyền lực. Gold Roger, vua của các hải tặc, đã đạt được tất cả những điều này và mọi thứ khác mà thế giới có thể mang lại. Những lời nói của ông trước khi chết đã truyền cảm hứng cho toàn thế giới lên đường ra khơi. "Kho báu của ta ư? Nếu các ngươi muốn nó, hãy cứ lấy! Mọi thứ ta để lại đều nằm ở trên biển đấy!". Vì thế, những nhà thám hiểm và những kẻ tham vọng đã đổ xô ra biển để tìm kiếm nó. Kỷ nguyên Đại Hải Tặc chính thức bắt đầu!

Mỗi cướp biển được biểu diễn bởi một đối tượng trong class `Pirate`. Class này gồm các thuộc tính sau:

- `name` (kiểu dữ liệu `string`): tên của cướp biển.
- `bounty` (kiểu dữ liệu `int`): tiền thưởng của cướp biển (được tính bằng triệu Beri).

Những cướp biển thường tụ tập thành băng, biểu diễn bởi một đối tượng trong class `Crew`. Class này gồm các thuộc tính sau:

- `name` (kiểu dữ liệu `string`): tên của băng cướp biển.
- `crewmate` (kiểu dữ liệu `Private*`): chứa các cướp biển trong băng, biểu diễn bằng một mảng động.
- `size` (kiểu dữ liệu `int`): số lượng cướp biển trong băng.
- `capacity` (kiểu dữ liệu `int`): sức chứa của thuyền. Kích thước của mảng cấp phát động bằng sức chứa.

Trong thế giới cướp biển, những băng cướp biển chỉ có thể hợp lại với nhau để tạo thành một băng cướp biển mạnh hơn. Trong trường hợp đó, tên của băng cướp biển ở dạng `<crew1Name>-<crew2Name> Alliance` (xem phần ví dụ để hiểu rõ hơn).
Hiện thực các phương thức và hàm sau:

1. Phương thức `void append(string name, int bounty)` trong class `Crew` để thêm một cướp biển ***ở cuối*** danh sách cướp biển trong băng. Nếu sức chứa của băng không đổi, nhân đôi sức chứa của băng.
2. Phương thức `void prepend(string name, int bounty)` trong class `Crew` để thêm một cướp biển ***ở đầu*** danh sách cướp biển trong băng. Nếu sức chứa của băng không đổi, nhân đôi sức chứa của băng.
3. Hàm `Crew* formAlliance(Crew* crew1, Crew* crew2)` biểu diễn sự hợp nhất của hai băng cướp biển. Ở trường hợp này, danh sách các thành viên trong liên minh sẽ do danh sách các thành viên trong từng băng gộp lại, và giữ nguyên thứ tự của từng danh sách. Nếu hai băng

cướp biển có cùng tên, băng có tiền thưởng cao hơn sẽ được giữ lại. Nếu một trong hai băng cướp biển không có thành viên nào, hàm trả về băng còn lại. Nếu cả hai băng không có thành viên, in ra `Can't form an Alliance!` và trả về `nullptr`.

```
class Pirate{
public:
    string name;
    int bounty;

    Pirate(){}
    Pirate(string name, int bounty);
    friend ostream & operator<<(ostream &os, const Pirate& pirate);
    friend class Crew;
    friend Crew* formAlliance(Crew* crew1, Crew* crew2);
};

class Crew{
public:
    string name;
    Pirate* crewmate;
    int size;
    int capacity;

    Crew(string name, int capacity=2);
    ~Crew();
    void printCrew();

    void append(string name, int bounty); //TODO
    void prepend(string name, int bounty); //TODO
    friend Crew* formAlliance(Crew* crew1, Crew* crew2); //TODO
}
```

Ví dụ:

Test	Result

`Pirate* luffy = new Pirate("Luffy", 3000);`
`Pirate* law = new Pirate("Law", 3000);`
 	
`cout << *law << endl;`
`cout << *luffy << endl;`
	`Pirate: Law, Bounty: 3000 million Berri`
	`Pirate: Luffy, Bounty: 3000 million Berri`

`Crew* StrawhatPirate = new Crew("Strawhat Pirates");`
`StrawhatPirate->append("Zoro", 1150);`
`StrawhatPirate->append("Sanji", 1032);`
`StrawhatPirate->printCrew();`
	`Crew: Strawhat Pirates`
	`Pirate: Zoro, Bounty: 1150 million Berri`
	`Pirate: Sanji, Bounty: 1032 million Berri`

```
Crew* StrawhatPirate = new Crew("Strawhat Pirates
    ");
Crew* HeartPirate = new Crew("Heart Pirates");

StrawhatPirate->prepend("Luffy", 3000);
StrawhatPirate->prepend("Zoro", 1150);
StrawhatPirate->prepend("Sanji", 1032);
HeartPirate->prepend("Law", 2000);
HeartPirate->prepend("Bepo", 100);
HeartPirate->prepend("Sanji", 4000);

StrawhatPirate->printCrew();
HeartPirate->printCrew();

cout << "-----------Form Alliance-----------\n";

Crew* alliance = formAlliance(StrawhatPirate,
    HeartPirate);
if(alliance){
    alliance->printCrew();
}
```

```
Crew: Strawhat Pirates
Pirate: Sanji, Bounty: 1032 million
    Berri
Pirate: Zoro, Bounty: 1150 million
    Berri
Pirate: Luffy, Bounty: 3000 million
    Berri
Crew: Heart Pirates
Pirate: Sanji, Bounty: 4000 million
    Berri
Pirate: Bepo, Bounty: 100 million Berri
Pirate: Law, Bounty: 2000 million Berri
-----------Form Alliance-----------
Crew: Alliance
Pirate: Sanji, Bounty: 4000 million
    Berri
Pirate: Zoro, Bounty: 1150 million
    Berri
Pirate: Luffy, Bounty: 3000 million
    Berri
Pirate: Bepo, Bounty: 100 million Berri
Pirate: Law, Bounty: 2000 million Berri
```