****

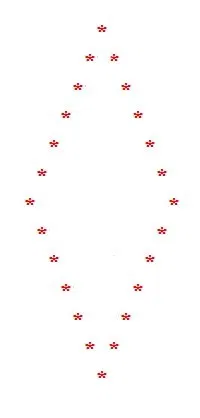
**BÀI TEST GIAI ĐOẠN 1**

**(Max: 100 điểm)**

Lưu ý: tất cả các bài toán lập trình đều lập trình hướng đối tượng. Mỗi ý ít nhất là 1 method.

**Câu 1**: Viết chương trình tính biểu thức sau: S(x,n) = x 0 + x 1 + x 2 + … + xn (**5 điểm**)

**Câu 2**: Tạo class **DrawAnyShape**. Viết chương trình để vẽ hình sau (**5 Điểm**)



Trong class bạn có thể viết method: drawShape(int height). Trong đó height là chiều cao của hình. Ví dụ trên là height=6

**Câu 3**: Tạo đối class **MyString**. Trong class có các method phục vụ các chức năng sau (**15 điểm**):

1. Tính tổng bảng mã ASCII của các ký tự trong chuỗi.
2. Tính tổng bảng mã ASCII của các ký tự in hoa trong chuỗi.
3. Liệt kê các ký tự có trong chuỗi có **cùng số lần xuất hiện** và số lần xuất hiện đó nhiều hơn n lần. n là một parameter của method (phân biệt ký tự hoa và thường)

Ví du:

S = “Hello programers. Im Developer”. **n = 2** Thì in ra như sau:

‘p’, ‘m’: 2

‘l’, ‘o’: 3

‘r’: 4

‘e’: 5

1. Cho hai chuỗi s1 và s2, tìm chuỗi con dài nhất có trong s1 và s2.

**Câu 4:** Tạo class **ArrIntManager.** Cho một mảng phần tử int bất kỳ. Ví dụ:

arrInt = [1,2, 3, 5,4, 1, 3, 4,5, 4, 5,9,7,0,11, 13,10,23, ….]. (**15 điểm**)

1. Tính tổng các số trong mảng
2. Tính tổng các số nguyên tố trong mảng
3. Tìm và in ra bộ 3 liên tiếp nhau sao có: arrInt[i] + arrInt[i+1] = arrInt[i+2]
4. Tìm và in ra dãy con dài nhất có tổng = S.
5. Một dãy số **tăng giảm ổn định** là dãy số nguyên thỏa mãn đặc điểm sau: các phần tử đầu sắp xếp thành một dãy tăng dần đến một phần tử đỉnh sau đó giảm dần. Ví dụ dãy: 1,2, 3, 5,4, 1 là một dãy **tăng giảm ổn định** có độ dài = 6. Hãy chỉ và in ra dãy con **Tăng giảm ổn định** có độ dài dài nhất thuộc **arrInt**. Nếu có 2 dãy dài nhất thì in ra dãy cuối cùng.

**Câu 5**: Tạo class ManagerInt. Trong class này viết phương

public void sumK(int[] m, int n) (**5 điểm**).

Chức năng của phương thức là: Tìm các mảng con (có thể có các phần tử không liên tiếp) trong mảng m có tổng = n và in ra chi tiết các mảng con đã tìm được đó.

Ví dụ: m = [1, 4, 6,3, 2, 2, 8]

n = 10

Kết quả:

* [1, 4, 3, 2]
* [4, 6]
* [6, 2, 2]
* [8, 2]
* [2, 8] (Khác trên vì chỉ số của số 2 ở 2 vị trí khác nhau).

**Câu 6 (10 điểm)**:

Cho 2 mảng số nguyên đã được sắp xếp. Tìm phần tử ở giữa của dãy đã được sắp xếp từ dãy được trộn từ 2 dãy trên sao cho **độ phức tạp của thuật toán là thấp nhất. Gới ý: min(log(m), log(n)).**

Ví ạ: ar1 [] = {1, 2, 5, 7, 8}; => m = 5

ar2[] = {3, 4, 6}; => n = 3

Dãy được gộp là: merge[] = {1, 2, 3, 4, 5, 6, 7, 8}. Số ở giữa là số: 4.

**Câu 7**: Tạo một class **Student** gồm có các thuộc tính sau: name, age, math, physical, chemistry, avg(float), rank(String). Tạo một class ManagerStudent, có thuộc tính: List<Student> students (**15 Điểm**).

1. Tạo 8 học sinh gồm, mỗi học sinh gồm: name, age, math, physical, chemistry.
2. Tính điểm trung bình(avg), và xếp hạng (rank) cho mỗi Student. Rank gồm có: GIOI(>=8.0), KHA(>=6.5 && < 8.0), TB(>= 5.0 && < 6.5), YEU(< 5).
3. Viết phương thức tìm kiếm 1 student có name là parameter của method đó.
4. Viết phương thức lấy ra danh sách Student có rank là GIOI.
5. Sắp xếp students tăng dần theo AlphaB. Nếu tên giống nhau thì sắp xếp tăng dần theo điểm TB.
6. Viết phương thức List<Student> smartSearch(String content). Search theo tên.

* Search giống. Có nghĩa content truyền vào trùng với name của Student. Cách này sẽ có rank cao nhất là = 10.
* Search gần đúng: Các chữ cái có trong content sẽ có trong name của Student. Cách này sẽ có rank = 9.
* Áp dụng quy tắc sửa lỗi chính tả. có rank = 8.

….

* Các bạn có thể apply một số quy tắc search khác mà bạn biết.

**Câu 8: Viết các câu lệnh SQL (postgresql) (30 điểm).**

1. Viết câu lệnh SQL tạo các bảng sau:

* **user**(id: int - primary key, username: varchar – not null, password: varchar – not null, fullname: varchar – not null, avatar: varchar – can null, birthday: timestamp – can null, created\_time: timestamp – not null).
* **friend** (id: int – primary key, sender\_id: int – foreign key not null, receiver\_id: int - foreign key -not null, status (pending, accepted, rejected), created\_time). friend liên kết với **user** thông qua foreign key: sender\_id, receiver\_id
* message(id, sender\_id: int – foreign key- not null, receiver\_id: int - foreign key – not null, type(text, image, video, file): varchar - not null, content: varchar - can null, status(sent, pending\_read, read): varchar, created\_time: datetime). message liên kết với **user** thông qua foreign key: sender\_id, receiver\_id

1. Viết câu lệnh SQL insert vào trong 3 table trên mỗi table ít nhất 4 records.
2. Viết câu lệnh SQL query:

* Lấy id, username, fullname, avatar: của các user có id = 2, 3.
* Lấy các bạn bè (gồm thông tin sau: id, username, fullname, avatar) của user có id = 2.
* Lấy tin nhắn của user có id = 2 với một bàn bè nào đó (ví dụ: id bạn bè = 3). Các trường lấy ra gồm: message\_id, sender\_id, receiver\_id, type, status, content, created\_time.
* Lấy tin nhắn cuối cùng (last\_message) với tất cả bạn bè của user có id = 2. Các trường lấy ra gồm: friend\_id, mesage\_id, type, status, content, sender\_id, created\_time.
* Lấy danh sách 10 user (id, username, avatar, birthday) có số lượng bạn bè nhiều nhất trong hệ thống.

FE  
Tạo form login:  
a. Khóa nút đăng nhập khi chưa nhập user/pass ;   
b. Icon eye để ẩn hiện mật khẩu ;   
c. Check required input javascript, Focus username….  
d. sau khi login thành công hiển 1 popup mới có dòng chữ: xin chào: {user name}  
e. Lưu thông tin đăng nhập vào local storage

