

BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH GIỮA KỲ

Họ và tên: Hoàng Văn Thắng

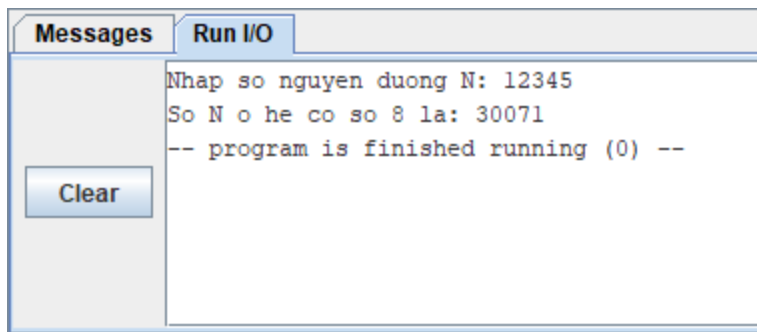
MSSV: 20235828

A. Số nguyên

12. Nhập số nguyên dương N từ bàn phím, in ra màn hình biểu diễn ở hệ cơ số 8 của N.

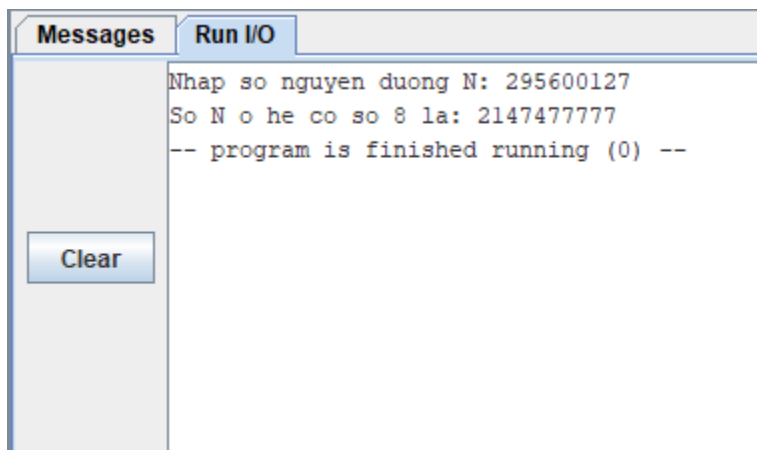
Mô tả thuật toán: Từ số nguyên dương N ban đầu, ta sẽ chia liên tiếp số N cho 8 lấy phần dư và cập nhật N bằng cách chia lấy phần nguyên N cho 8. Ta thêm một biến đáp án, một biến chữ số hàng. Mỗi lần chia lấy phần dư N cho 8 thì ta lấy kết quả nhân với biến chữ số hàng rồi cộng vào biến đáp án, rồi cuối cùng ta nhân biến chữ số hàng cho 10. Ta cứ làm như vậy cho đến khi ta có số N bằng 0 rồi in kết quả của biến đáp án. (Nếu $N \leq 0$ thì ta in lỗi, do không phải là số nguyên dương)

Ảnh chụp màn hình kết quả chạy:



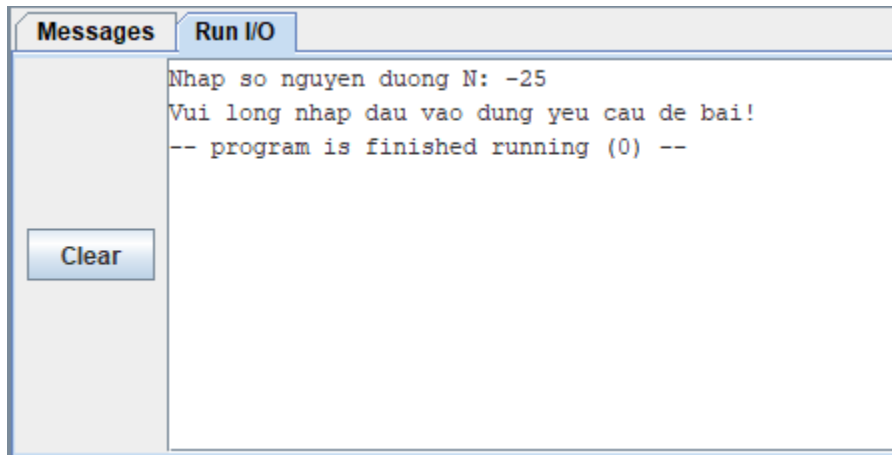
```
Messages Run I/O
Nhap so nguyen duong N: 12345
So N o he co so 8 la: 30071
-- program is finished running (0) --
Clear
```

- Với N lớn



```
Messages Run I/O
Nhap so nguyen duong N: 295600127
So N o he co so 8 la: 214747777
-- program is finished running (0) --
Clear
```

- Với số N âm



⇒ Đúng với kết quả

B. Mảng

5. Nhập mảng từ bàn phím. Sắp xếp các phần tử có giá trị dương tăng dần, các phần tử còn lại giữ nguyên vị trí. Ví dụ: Nhập mảng [-1, 150, 190, 170, -2, -3, 160, 180], kết quả sau khi sắp xếp [-1, 150, 160, 170, -2, -3, 180, 190].

Mô tả thuật toán: Từ mảng số nguyên ban đầu, ta sẽ duyệt qua từng phần tử trong mảng. Nếu phần tử đó là số dương thì ta chép nó vào một mảng phụ riêng để chứa các số dương. Sau khi đã tách được tất cả các số dương, ta tiến hành sắp xếp mảng phụ này theo thứ tự tăng dần bằng cách sử dụng thuật toán bubble sort. Sau khi mảng phụ chứa các số dương đã được sắp xếp, ta quay lại duyệt mảng ban đầu. Mỗi khi gặp phần tử dương, ta thay thế nó bằng phần tử tương ứng trong mảng phụ đã được sắp xếp, còn nếu là phần tử âm thì ta giữ nguyên như ban đầu. Cuối cùng, ta in ra toàn bộ mảng sau khi đã xử lý. (Nếu số phần tử trong mảng là 0 thì có thể in lỗi hoặc kết thúc luôn chương trình.)

Ảnh chụp màn hình kết quả chạy:

The screenshot shows a window with two tabs: 'Messages' and 'Run I/O'. The 'Run I/O' tab is active, displaying the following text: 'Nhap so phan tu cua mang: 8', followed by eight lines of input: 'Nhap phan tu thu 0 la: 25', 'Nhap phan tu thu 1 la: -7', 'Nhap phan tu thu 2 la: 15', 'Nhap phan tu thu 3 la: 30', 'Nhap phan tu thu 4 la: -85', 'Nhap phan tu thu 5 la: 12', 'Nhap phan tu thu 6 la: -4', and 'Nhap phan tu thu 7 la: 65'. Below these inputs, the output is shown: '12 -7 15 25 -85 30 -4 65'. At the bottom, it says '-- program is finished running (0) --'. On the left side of the window, there is a 'Clear' button.

```
Nhap so phan tu cua mang: 8
Nhap phan tu thu 0 la: 25
Nhap phan tu thu 1 la: -7
Nhap phan tu thu 2 la: 15
Nhap phan tu thu 3 la: 30
Nhap phan tu thu 4 la: -85
Nhap phan tu thu 5 la: 12
Nhap phan tu thu 6 la: -4
Nhap phan tu thu 7 la: 65
12 -7 15 25 -85 30 -4 65
-- program is finished running (0) --
```

- Nhập số phần tử của mảng là 0

The screenshot shows a window with two tabs: 'Messages' and 'Run I/O'. The 'Run I/O' tab is active, displaying the following text: 'Nhap so phan tu cua mang: 0', 'So phan tu cua mang khong hop le', and '-- program is finished running (0) --'. On the left side of the window, there is a 'Clear' button.

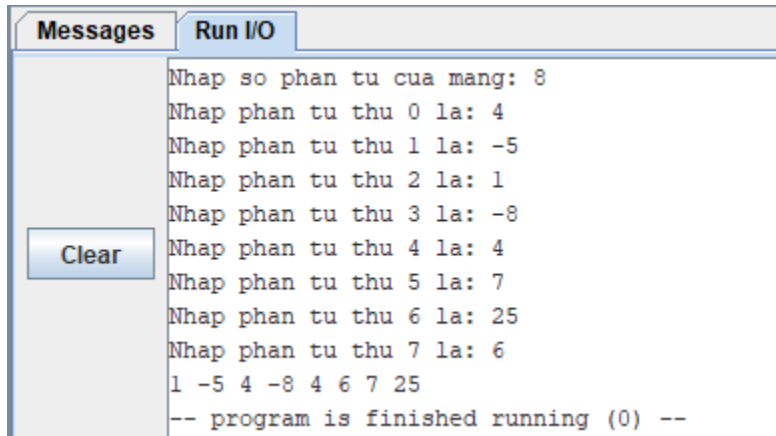
```
Nhap so phan tu cua mang: 0
So phan tu cua mang khong hop le
-- program is finished running (0) --
```

- Nhập số phần tử của mảng là một số nguyên âm

The screenshot shows a window with two tabs: 'Messages' and 'Run I/O'. The 'Run I/O' tab is active, displaying the following text: 'Nhap so phan tu cua mang: -5', 'So phan tu cua mang khong hop le', and '-- program is finished running (0) --'. On the left side of the window, there is a 'Clear' button.

```
Nhap so phan tu cua mang: -5
So phan tu cua mang khong hop le
-- program is finished running (0) --
```

- Mảng có nhiều phần tử dương bằng nhau



```
Messages Run I/O
Nhap so phan tu cua mang: 8
Nhap phan tu thu 0 la: 4
Nhap phan tu thu 1 la: -5
Nhap phan tu thu 2 la: 1
Nhap phan tu thu 3 la: -8
Nhap phan tu thu 4 la: 4
Nhap phan tu thu 5 la: 7
Nhap phan tu thu 6 la: 25
Nhap phan tu thu 7 la: 6
1 -5 4 -8 4 6 7 25
-- program is finished running (0) --
```

⇒ Đúng với kết quả

C. Xâu ký tự

7. Nhập vào 2 xâu ký tự s1 và s2, kiểm tra xâu s2 có phải là xâu con của s1 hay không?

Mô tả thuật toán: Để kiểm tra xem s2 có phải xâu con của s1 thì đầu tiên ta tìm độ dài của từng chuỗi, sau đó ta kiểm tra xem nếu độ dài s2 lớn hơn s1 thì chuyển đến false_Print, sau đó kiểm tra xem nếu s2 là chuỗi rỗng thì chắc chắn là chuỗi con nên chuyển đến true_Print.

+ Sau khi kiểm tra các trường hợp đặc biệt thì khởi tạo các giá trị để chuyển bị cho vòng lặp duyệt lần lượt các phần tử của chuỗi, khởi tạo các biến đếm cho vòng lặp, khởi tạo điều kiện dừng là $t0 = s1 - s2 + 1$

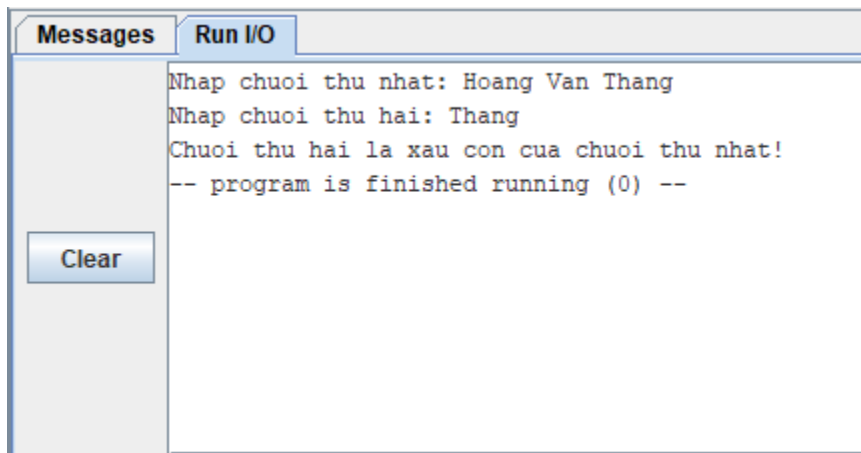
+ Bắt đầu thực hiện vòng lặp 1: Kiểm tra điều kiện dừng nếu biến đếm bằng với điều kiện dừng t0 thì dừng vòng lặp, sau đó khởi tạo giá trị vị trí của chuỗi s1, s2 và biến đếm cho vòng lặp loop2.

+ Trong loop2 kiểm tra nếu biến đếm t2 bằng với độ dài của chuỗi s2 thì chuyển đến true_print, điều này xảy ra khi các ký tự $s1[i+j] = s2[j]$, nếu $s1[i+j] \neq s2[j]$ thì kết thúc loop2 sau đó tăng biến đếm loop1, nếu điều kiện đúng thì tăng biến đếm cho loop2 và tiếp tục kiểm tra

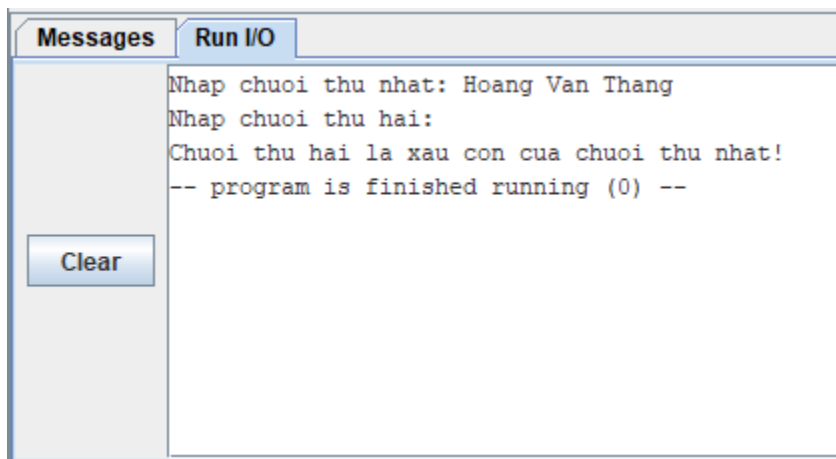
+ Sau khi vòng lặp kết thúc thì in kết quả ra màn hình thông qua false_Print và true_Print rồi kết thúc chương trình

Ảnh chụp màn hình:

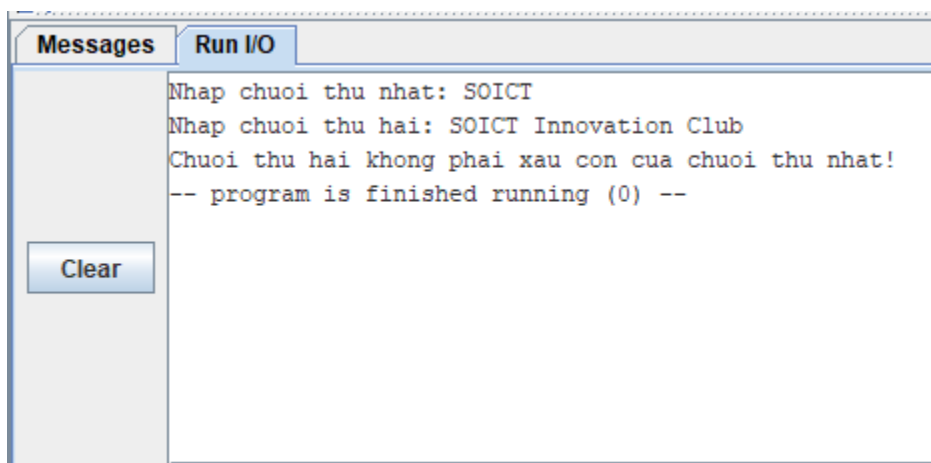
- Với s2 là xâu con của s1



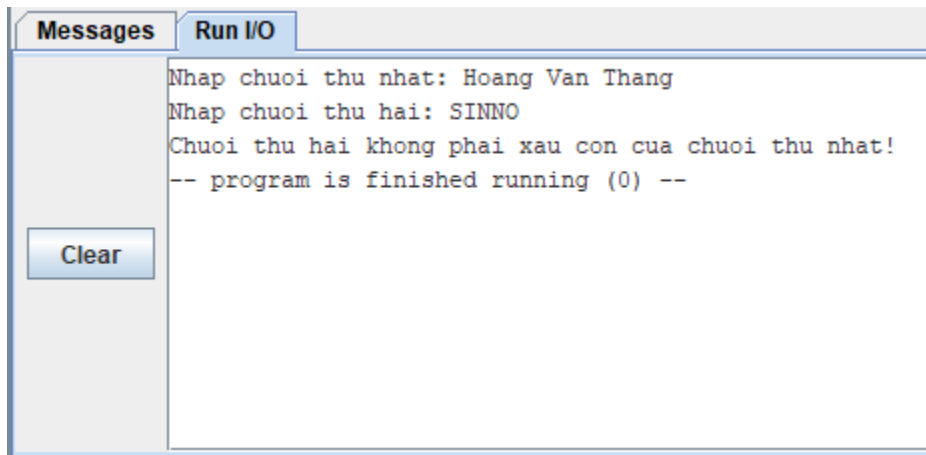
- Với s2 là chuỗi rỗng



- Xâu s2 có độ dài lớn hơn xâu s1



- Với s2 không phải là xâu con của s1



⇒ Đúng với kết quả