

BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Lab 5 : Nhập xuất dữ liệu với hàm ECALL, xử lý chuỗi ký tự

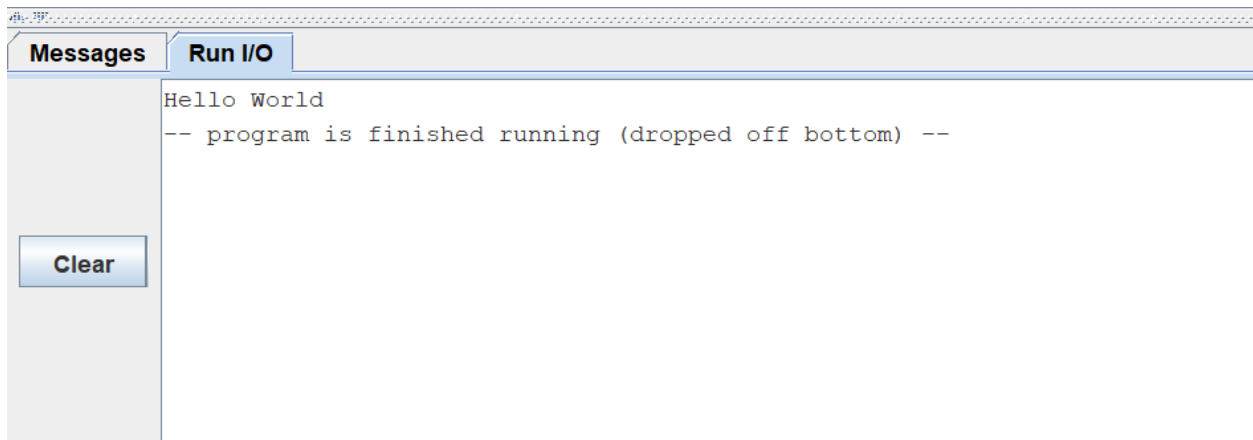
Họ tên	MSSV
Phạm Minh Hiền	20235705

Assignment 1:

Tạo project thực hiện Home Assignment 1. Dịch và nạp chương trình lên trình mô phỏng. Chạy và quan sát kết quả. Chuyển đến Data Segment, kiểm tra cách chuỗi ký tự được lưu trữ trong bộ nhớ.

```
.data
test: .asciz "Hello World"
.text
    li a7, 4
    la a0, test
    ecall
```

- Kết quả chạy :



- Cách chuỗi ký tự được lưu trong Data Segment :

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	l l e H	o W o	\0 d l r	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

+ Chuỗi “Hello World” được lưu trữ dưới dạng các byte ASCII liên tiếp, kết thúc bằng 1 byte null (\0).

+ Ta thấy ở Data Segment, một cột value chứa 4 giá trị tương ứng với các địa chỉ tiếp theo. Ví dụ ở đây, ta có cột Value (+0), trong đó chứa giá trị tại địa chỉ 0x10010000, 0x10010001, 0x10010002, 0x10010003 lần lượt là l l e H.

+ Do mỗi ô thuộc cột value chỉ chứa 4 giá trị cho 4 địa chỉ, nên tại Value (+0) ta chỉ lấy “Hell”, mặt khác các ký tự này được lưu theo thứ tự MSB đến LSB nên ta thấy “Hell” được lưu trữ trong Data Segment là “lleH”.

+ Tương tự với các ký tự còn lại.

+ Ta test thử với các trường hợp khác cũng cho ra kết quả tương tự.

Assignment 2:

Tạo project thực hiện chương trình in tổng của hai toán hạng nằm trong thanh ghi s0 và s1 theo định dạng sau: "The sum of (s0) and (s1) is (result)".

Source Code :

```
.data
```

```
message: .asciz "The sum of "
```

```
and_msg: .asciz " and "
```

```
is_msg: .asciz " is "
```

```
.text
```

```
li s0, 10 # Khởi tạo s0
```

```
li s1, 20 # Khởi tạo s1
```

```
add s2, s0, s1 # Tính tổng s2 = s0 + s1
```

```
la a0, message # In "The sum of "
```

```
li a7, 4
```

```
ecall
```

```
add a0, zero, s0 # In giá trị thu 1
```

```
li a7, 1
```

```
ecall
```

```
la a0, and_msg # In " and "
```

```
li a7, 4
```

```
ecall
```

```
add a0, zero, s1 # In giá trị thu 2
```

```
li a7, 1
```

```
ecall
```

```
la a0, is_msg # In " is "
```

```
li a7, 4
```

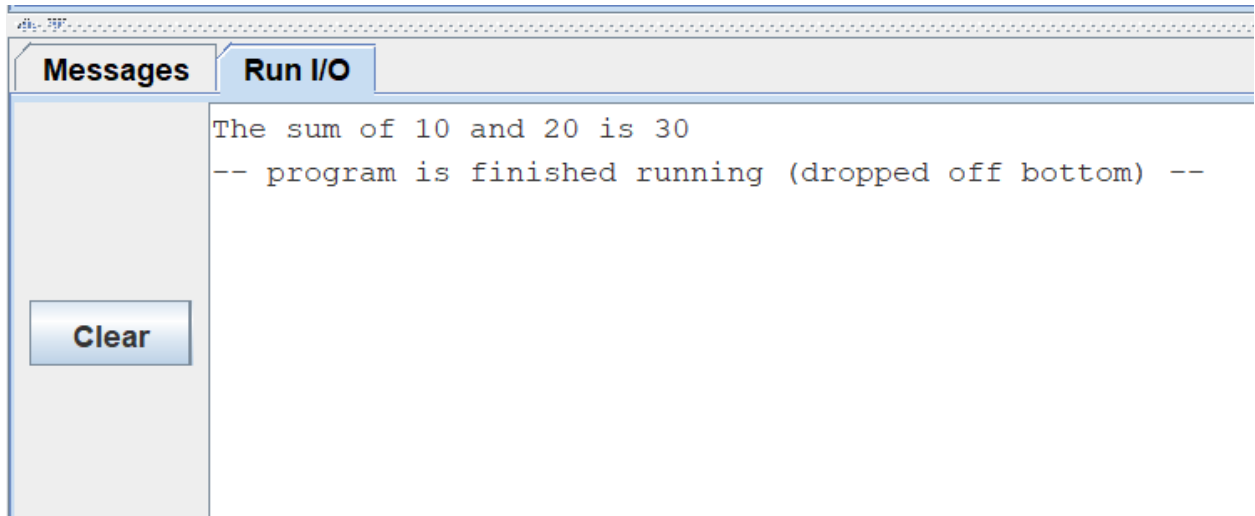
ecall

add a0, zero, s2 # ln tong

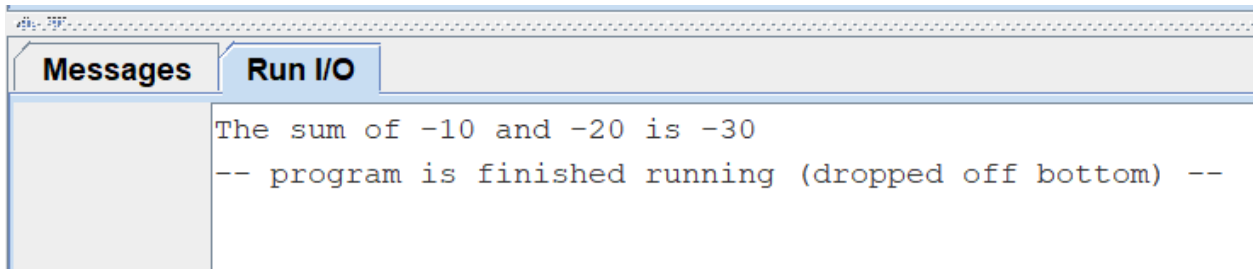
li a7, 1

ecall

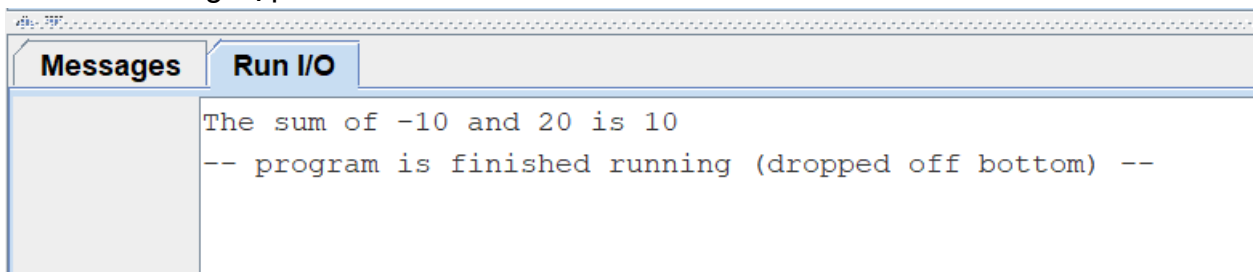
- Kết quả chạy chương trình với trường hợp **s0** = 10 và **s1** = 20:



- Nhận xét : Tổng 2 số 10 và 20 cho ra là 30 và in ra màn hình "The sum of 10 and 20 is 30" đúng cú pháp yêu cầu => **Chương trình hoạt động**.
- Thử với trường hợp 2 số -10 và -20 :



- Thử với trường hợp 2 số -10 và 20 :



Kết luận : Chương trình hoạt động như yêu cầu.

Assignment 3:

Tạo project thực hiện Home Assignment 2. Đọc hiểu mã nguồn, khởi tạo các biến cần thiết cho chương trình, thực hiện hàm strcpy. Dịch và nạp lên mô phỏng, chạy và quan sát kết quả.

Source Code :

.data

x: .space 32

y: .asciz "Hello"

.text

strcpy:

la a0, x

la a1, y

add s0, zero, zero

L1:

add t1, s0, a1

lb t2, 0(t1)

add t3, s0, a0

sb t2, 0(t3)

beq t2, zero, end_of_strcpy

addi s0, s0, 1

j L1

end_of_strcpy:

la a0, x

li a7, 4

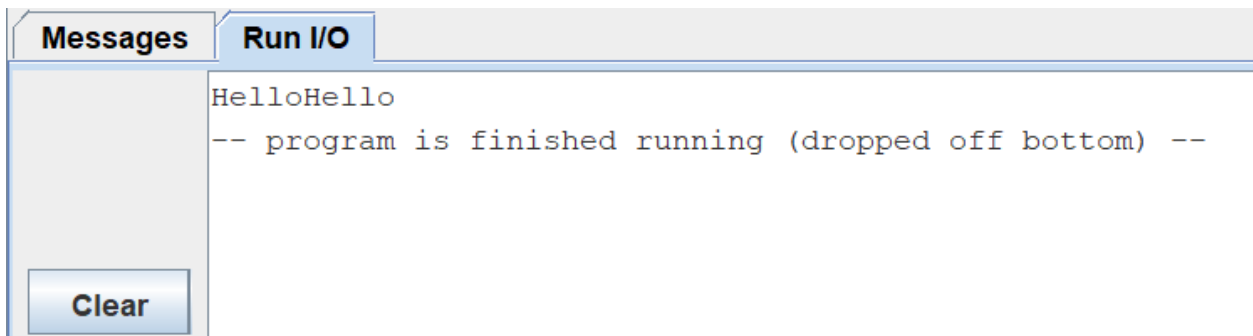
ecall

la a0, y

li a7, 4

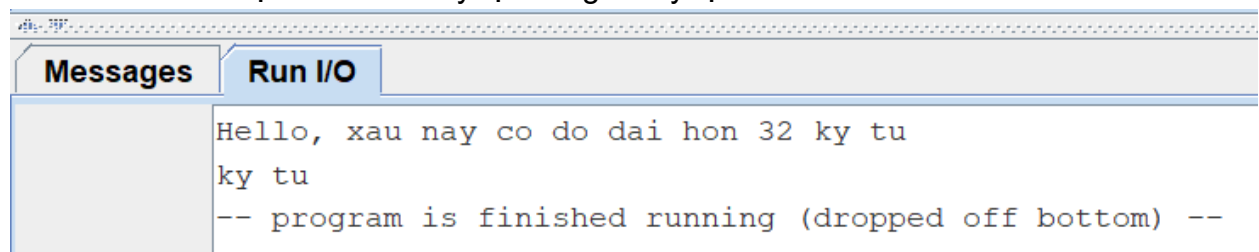
ecall

- Bổ sung thêm các dòng lệnh **la** để khai báo a0 và a1 tương ứng với địa chỉ x và y.
- Bổ sung thêm đoạn lệnh in chuỗi x và y ra màn hình trong nhãn *end_of_strcpy*.
- Kết quả chạy chương trình :



```
Messages Run I/O
HelloHello
-- program is finished running (dropped off bottom) --
Clear
```

- Ta thấy, chuỗi “Hello” trong y đã được copy sang x thông qua vòng lặp L1 :
 - + Duyệt từng ký tự trong y thông qua **t1**, **t2** và lưu ký tự vào x.
 - + Kiểm tra điều kiện khi gặp ký tự null thì *end_of_strcpy*.
 - + Tiếp tục vòng lặp.
- Sau khi chạy chương trình chuỗi “Hello” trong y không hề mất đi, chứng minh rằng chương trình chỉ copy “Hello” sang x.
- Thử với xâu có độ dài hơn 32 ký tự bao gồm ký tự null :



```
Messages Run I/O
Hello, xau nay co do dai hon 32 ky tu
ky tu
-- program is finished running (dropped off bottom) --
```

- + Ta thấy xâu này có 38 ký tự tính cả null, do tràn bộ nhớ nên chương trình hoạt động không chính xác.
- + Do địa chỉ của x nằm trước y 32 byte nên khi chương trình thực hiện sẽ ghi đè.

Assignment 4:

Tạo project thực hiện Home Assignment 3, sử dụng ecall để nhập chuỗi ký tự cần đếm, và in kết quả ra màn hình.

Source Code :

```
.data
```

```
string: .space 50
```

```
message1: .asciz "Nhap xau: "
```

```
message2: .asciz "Do dai xau la: "
```

```
.text
```

```
main:
```

```
get_string:
```

```
    la a0, message1 # ln "Nhap xau: "
```

```
    li a7, 4
```

```
    ecall
```

```
    la a0, string # Nhap xau mong muon
```

```
    li a1, 50
```

```
    li a7, 8
```

```
    ecall
```

```
get_length:
```

```
    la a0, string
```

```
    li t0, 0
```

```
check_char:
```

```
    add t1, a0, t0
```

```
    lb t2, 0(t1)
```

```
    beq t2, zero, end_of_str
```

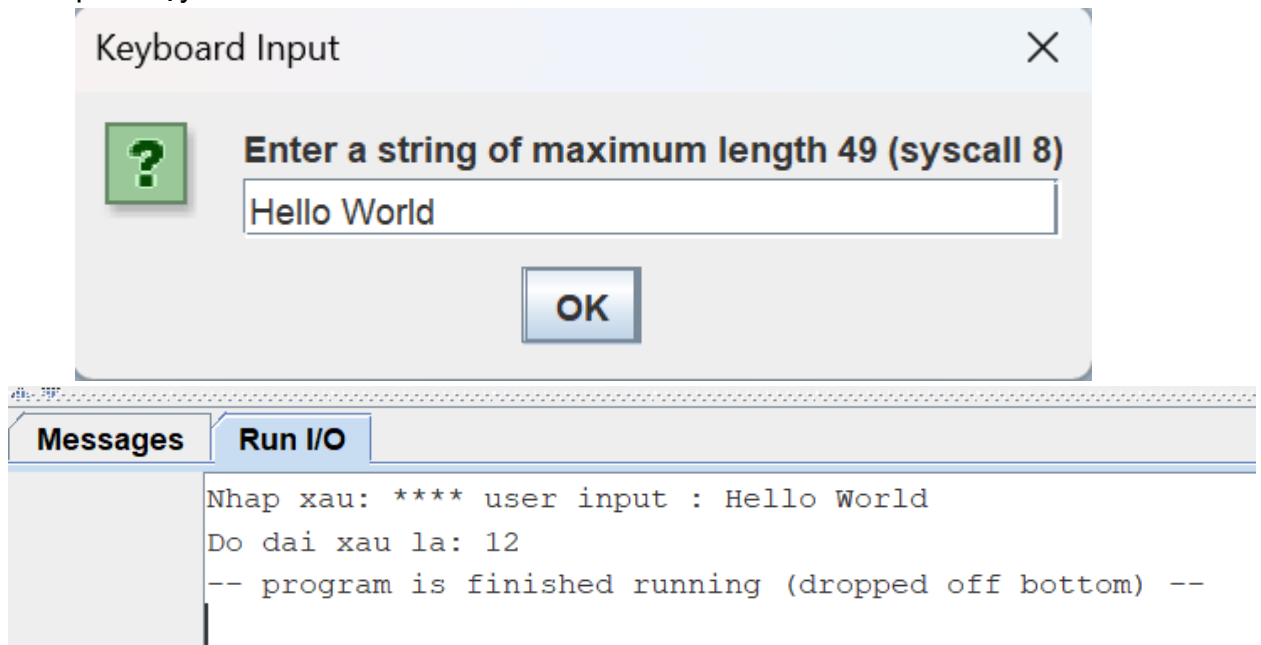
```
    addi t0, t0, 1
```

```

        j check_char
end_of_str:
end_of_length:
print_length:
    la a0, message2 # In "Do dai xau la: "
    li a7, 4
    ecall
    addi a0, t0, 0 # In ra do dai cua xau
    li a7, 1
    ecall

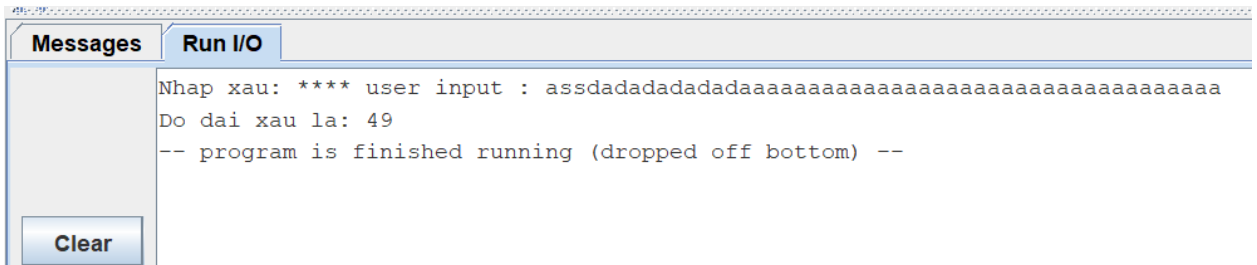
```

- Đã bổ sung đoạn code trong *get_string* để lấy xâu mong muốn và bổ sung code trong *print_length* để in ra độ dài của xâu.
- Kết quả chạy với xâu “Hello World” :



+ Ta thấy chương trình chạy chính xác khi đếm số ký tự của xâu “Hello World” là 12 bao gồm cả ký tự ‘ ‘ và ‘\0’.

- Kết quả khi chạy thử với xâu có độ dài lớn hơn 49 (bao gồm cả ký tự null) :



- Ta thấy chương trình cho ra kết quả là 49 ký tự, trong khi chuỗi này có 50 ký tự (bao gồm cả ký tự null).
- Như vậy chương trình đếm tối đa được 49 ký tự, với các xâu dài hơn thì kết quả cho ra là 49, điều này xảy ra là do ta chỉ khai báo **string** với độ dài 50 mà thôi.

Assignment 5:

Viết chương trình cho phép người dùng nhập chuỗi ký tự bằng cách nhập từng ký tự từ bàn phím. Việc nhập sẽ kết thúc khi người dùng nhấn Enter hoặc khi độ dài chuỗi ký tự vượt quá 20. In chuỗi đã nhập theo chiều ngược lại.

Source Code :

```
.data
message: .asciz "Nhap xau: "
message1: .asciz "Chuoi dao nguoc: "
counter: .space 21

.text
main:
    la a0, message # Hien thi thong bao nhap
    li a7, 4
    ecall

    la t0, counter # Khoi tao bo dem
    li t1, 0

loop:
    li a7, 12 # Doc 1 ky tu
    ecall

    li t2, 10 # Kiem tra neu la 10 ('\n') thi ket thuc
    beq a0, t2, end_loop
    sb a0, 0(t0) # Luu ky tu
    addi t0, t0, 1 # Tang dia chi luu
    addi t1, t1, 1 # Tang bo dem
    li t3, 20 # Kiem tra 20 ky tu
    bge t1, t3, end_loop
```

```

        j loop
end_loop:
    sb zero, 0(t0) # Them ky tu '\n'
    li a7, 4 # Hien thi thong bao xuat
    la a0, message1
    ecall

    la t0, counter # In nguoc chuoi
    add t0 ,t0 ,t1 # Tro den ky tu cuoi cung

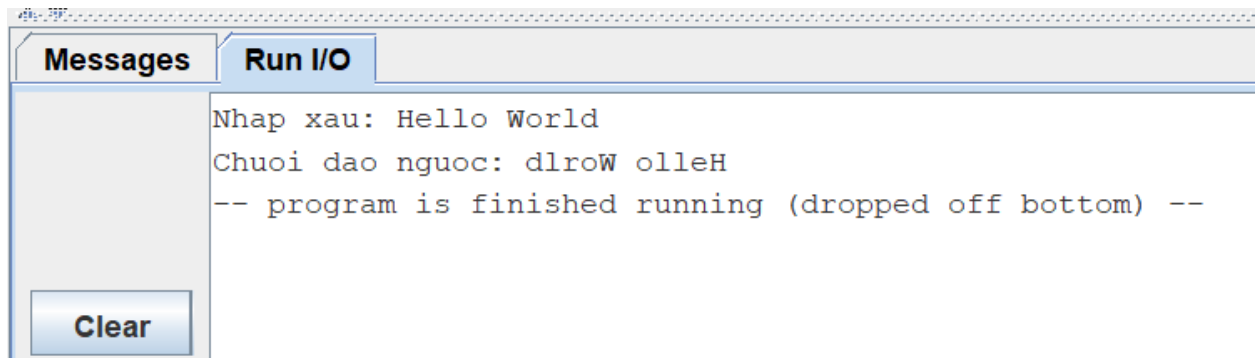
reverse:
    addi t0, t0, -1 # Lui ve ky tu truoc
    lb a0, 0(t0) # Lay ky tu
    li a7, 11 # In ky tu
    ecall

    addi t1, t1, -1 # Giam bo dem
    bgt t1, zero, reverse

end:

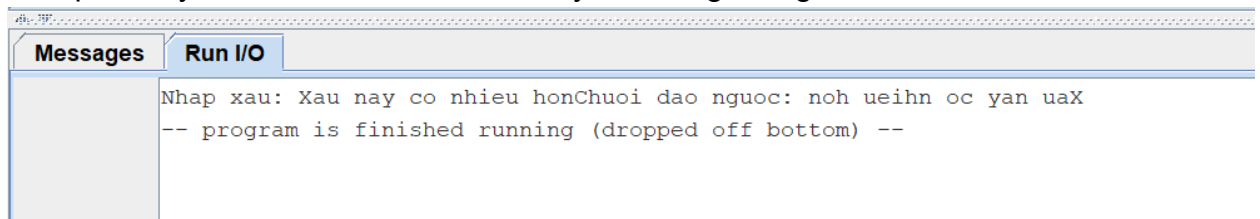
```

- Nguyên lý hoạt động :
 - + Chương trình yêu cầu nhập 1 xâu không quá 20 ký tự (không bao gồm ký tự null).
 - + Chương trình sẽ duyệt từng ký tự thông qua vòng lặp *loop* để kiểm tra xem ký tự đó có phải 10 (\n) hay không. Nếu là ký tự '\n' thì *end_loop*.
 - + Sau khi kiểm tra xâu ký tự, chương trình thực hiện đảo ngược xâu thông qua giảm bộ đếm t1 tại vị trí ký tự cuối cùng cho đến khi bộ đếm t1 = 0.
- Kết quả chạy thử với xâu "Hello World" :



+ Ta thấy kết quả chạy thử chính xác, có vẻ chương trình hoạt động theo đúng yêu cầu.

- Kết quả chạy thử với xâu có độ dài 20 ký tự không bao gồm null :



+ “Xau nay co nhieu hon” có 20 ký tự chưa tính null, khi ta nhập đến ký tự ‘n’ trong ‘hon’ thì chương trình ngay lập tức thực hiện và không cho phép ta nhập thêm.

+ Chương trình vẫn hoạt động chính xác và đưa ra xâu đảo ngược.

⇒ Vậy chương trình hoạt động đúng kỳ vọng.