Thực hành kiến trúc máy tính

Báo cáo giữa kì

IT3280 – GVHD: Ngô Lam Trung

|  |  |
| --- | --- |
| **Họ Tên** | **Lê Thành An** |
| **MSSV** | **20235631** |

1. **Số nguyên – Câu 13**

***Đề bài:*** Nhập 3 số A, B, C nguyên dương từ bàn phím. Kiểm tra A, B, C có phải là 3 cạnh của một tam giác không. Nếu là tam giác thì có phải là tam giác cân không?

* 1. Đoạn mã nguồn

**.data**

**tamgiac: .string "Đây là tam giác không cân\n"**

**tamgiaccan: .string "Đây là tam giác cân\n"**

**khongtamgiac: .string "Đây không phải là tam giác\n"**

**.text**

**main:**

**# Nhập cạnh a**

**li a7, 5**

**ecall**

**add s0, a0, zero # Lưu giá trị a vào thanh ghi s0**

**# Nhập cạnh b**

**li a7, 5**

**ecall**

**add s1, a0, zero # Lưu giá trị b vào thanh ghi s1**

**# Nhập cạnh c**

**li a7, 5**

**ecall**

**add s2, a0, zero # Lưu giá trị c vào thanh ghi s2**

**# Kiểm tra các cạnh > 0**

**ble s0, zero, C # Nhảy đến nhãn C nếu a <= 0**

**ble s1, zero, C # Nhảy đến nhãn C nếu b <= 0**

**ble s2, zero, C # Nhảy đến nhãn C nếu c <= 0**

**# Kiểm tra điều kiện tổng 2 cạnh > cạnh còn lại**

**add t0, s0, s1 # t0 = a + b**

**ble t0, s2, C # Nhảy đến C nếu a + b <= c**

**add t1, s1, s2 # t1 = b + c**

**ble t1, s0, C # Nhảy đến C nếu b + c <= a**

**add t2, s2, s0 # t2 = c + a**

**ble t2, s1, C # Nhảy đến C nếu c + a <= b**

**# Kiểm tra tam giác cân**

**beq s0, s1, A # Nhảy đến A nếu a == b**

**beq s1, s2, A # Nhảy đến A nếu b == c**

**beq s2, s0, A # Nhảy đến A nếu c == a**

**la a0, tamgiac # Địa chỉ thông báo tam giác thường vào a0**

**j print # Nhảy đến phần in kết quả**

**A:**

**la a0, tamgiaccan # Địa chỉ thông báo tam giác cân vào a0**

**j print # Nhảy đến phần in kết quả**

**C:**

**la a0, khongtamgiac # Địa chỉ thông báo không phải tam giác vào a0**

**print:**

**li a7, 4 # in chuỗi**

**ecall**

**li a7, 10 # thoát chương trình**

**ecall**

* 1. Mô tả thuật toán và các chương trình con(nếu có):

**Mô tả thuật toán:**

**1. Nhập liệu**

* Nhập giá trị độ dài 3 cạnh a, b, c từ người dùng
* Sử dụng syscall 5 để đọc số nguyên
* Lưu giá trị vào các thanh ghi s0 (a), s1 (b), s2 (c)

**2. Kiểm tra tính hợp lệ của tam giác**

* Kiểm tra điều kiện các cạnh phải dương (> 0)
* Kiểm tra điều kiện tam giác: tổng 2 cạnh > cạnh còn lại
  + a + b > c
  + b + c > a
  + c + a > b
* Nếu không thỏa mãn điều kiện, nhảy đến nhãn C

**3. Phân loại tam giác**

* Nếu điều kiện tam giác thỏa mãn, kiểm tra xem có phải tam giác cân:
  + Nếu a = b hoặc b = c hoặc c = a, đây là tam giác cân, nhảy đến nhãn A
  + Nếu không, đây là tam giác thường, tiếp tục xuống dưới

**Các chương trình con**

**A**

Chức năng: Xử lý trường hợp tam giác cân

* Nạp địa chỉ của chuỗi "Đây là tam giác cân" vào thanh ghi a0
* Nhảy đến nhãn print để hiển thị kết quả

**C**

Chức năng: Xử lý trường hợp không phải tam giác

* Nạp địa chỉ của chuỗi "Đây không phải là tam giác" vào thanh ghi a0
* Tiếp tục xuống dưới đến nhãn print

**print**

Chức năng: In thông báo kết quả và kết thúc chương trình

* Sử dụng syscall 4 để in chuỗi thông báo trong thanh ghi a0
* Sử dụng syscall 10 để kết thúc chương trình
  1. Ảnh chụp màn hình kết quả thực hiện chương trình

**TH1**: Tam giác không cân - Nhập A = 12, B = 13, C = 14

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

Kết quả:

Ảnh có chứa ảnh chụp màn hình, văn bản, phần mềm, hàng

Nội dung do AI tạo ra có thể không chính xác.

**TH2:** Tam giác cân - Nhập A = 10, B = 15, C = 10

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

Kết quả:

Ảnh có chứa ảnh chụp màn hình, văn bản, hàng, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

**TH3**: Không tạo thành tam giác - Nhập A = 5, B = 6, C = 12Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Nội dung do AI tạo ra có thể không chính xác.

Kết quả:

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng

Nội dung do AI tạo ra có thể không chính xác.

1. **Mảng – Câu 17**

***Đề bài:*** Nhập mảng số nguyên từ bàn phím. In ra màn hình số chẵn lớn nhất nhỏ hơn mọi số lẻ trong mảng.

* 1. Đoạn mã nguồn

**.data**

**error: .asciz "Lỗi: kích thước mảng không hợp lệ\n"**

**result: .asciz "Số chẵn lớn nhất nhỏ hơn mọi số lẻ là: "**

**not\_found\_msg: .asciz "Không tìm được số chẵn thỏa mãn yêu cầu\n"**

**no\_odd\_msg: .asciz "Không có số lẻ nào trong mảng Số chẵn lớn nhất là:"**

**.align 2 # Căn chỉnh dữ liệu theo địa chỉ chia hết cho 4 (tối ưu truy cập bộ nhớ)**

**array: .space 400 # Cấp phát 400 byte**

**.text**

**main:**

**# Nhập kích thước mảng**

**li a7, 5**

**ecall**

**mv s0, a0 # Lưu kích thước mảng vào s0**

**# Kiểm tra kích thước hợp lệ (0 < size <= 100)**

**li t0, 0 # t0 = 0**

**ble s0, t0, Loi\_kich\_thuoc # Nếu size <= 0, báo lỗi**

**li t0, 100 # t0 = 100**

**ble t0, s0, Loi\_kich\_thuoc # Nếu size > 100, báo lỗi**

**# Nhập các phần tử mảng**

**la s1, array # s1 = địa chỉ gốc của mảng**

**li s2, 0 # s2 = biến đếm i = 0**

**read\_loop:**

**beq s2, s0, sort\_array # Nếu i == size, chuyển sang sắp xếp**

**li a7, 1 # In chỉ số phần tử hiện tại (i)**

**mv a0, s2**

**ecall**

**li a7, 5 # Đọc phần tử thứ i từ bàn phím**

**ecall**

**# Lưu phần tử vào mảng: array[i] = a0**

**slli t1, s2, 2 # t1 = i \* 4 (offset phần tử thứ i)**

**add t2, s1, t1 # t2 = địa chỉ array[i]**

**sw a0, 0(t2) # Lưu giá trị vào array[i]**

**addi s2, s2, 1 # Tăng biến đếm i**

**j read\_loop # Lặp lại vòng lặp**

**# Sắp xếp mảng tăng dần bằng Bubble Sort**

**sort\_array:**

**li t0, 0 # t0 = i = 0 (vòng lặp ngoài)**

**outer\_loop:**

**bge t0, s0, find\_result # Nếu i >= size, chuyển sang tìm kết quả**

**li t1, 0 # t1 = j = 0 (vòng lặp trong)**

**inner\_loop:**

**sub t2, s0, t0 # t2 = size - i**

**addi t2, t2, -1 # t2 = size - i - 1 (giới hạn vòng lặp trong)**

**bge t1, t2, next\_outer # Nếu j >= size - i - 1, chuyển vòng ngoài**

**# Load array[j] và array[j+1]**

**slli t3, t1, 2 # t3 = j \* 4**

**add t4, s1, t3 # t4 = &array[j]**

**lw t5, 0(t4) # t5 = array[j]**

**addi t6, t1, 1 # t6 = j + 1**

**slli a1, t6, 2 # a1 = (j + 1) \* 4**

**add a2, s1, a1 # a2 = &array[j+1]**

**lw a3, 0(a2) # a3 = array[j+1]**

**ble t5, a3, skip # Nếu array[j] <= array[j+1], bỏ qua hoán đổi**

**# Hoán đổi array[j] và array[j+1]**

**sw a3, 0(t4) # array[j] = array[j+1]**

**sw t5, 0(a2) # array[j+1] = array[j]**

**skip:**

**addi t1, t1, 1 # j++**

**j inner\_loop**

**next\_outer:**

**addi t0, t0, 1 # i++**

**j outer\_loop**

**# Tìm số chẵn lớn nhất nhỏ hơn mọi số lẻ**

**find\_result:**

**li t0, 0 # t0 = chỉ số duyệt mảng**

**li t1, -1 # t1 = vị trí số lẻ đầu tiên (khởi tạo -1)**

**li t3, 0 # t3 = flag (0: chưa tìm thấy số lẻ)**

**find\_loop:**

**beq t0, s0, handle\_no\_odd # Nếu duyệt hết mảng, kiểm tra trường hợp không có số lẻ**

**slli t2, t0, 2 # t2 = offset phần tử thứ t0**

**add t4, s1, t2 # t4 = &array[t0]**

**lw t5, 0(t4) # t5 = array[t0]**

**andi t6, t5, 1 # Kiểm tra LSB (0: chẵn, 1: lẻ)**

**beqz t6, next\_find # Nếu chẵn, bỏ qua**

**mv t1, t0 # Lưu vị trí số lẻ đầu tiên**

**li t3, 1 # Đánh dấu đã tìm thấy số lẻ**

**j check\_position # Kiểm tra vị trí số lẻ**

**next\_find:**

**addi t0, t0, 1 # Tăng chỉ số duyệt**

**j find\_loop**

**# Xử lý vị trí số lẻ đầu tiên**

**check\_position:**

**beq t1, zero, not\_found # Nếu số lẻ ở vị trí 0, không có số chẵn phía trước**

**# Lấy phần tử ngay trước số lẻ đầu tiên (chắc chắn là số chẵn lớn nhất)**

**addi t6, t1, -1 # t6 = t1 - 1**

**slli t2, t6, 2 # Tính offset**

**add t4, s1, t2 # t4 = &array[t6]**

**lw a0, 0(t4) # a0 = array[t6] (số chẵn cần tìm)**

**# In kết quả**

**li a7, 4**

**la a0, result**

**ecall**

**lw a0, 0(t4) # Load lại giá trị vào a0 để in ra kết quả**

**li a7, 1**

**ecall**

**j exit # Kết thúc chương trình**

**# Trường hợp không có số lẻ**

**handle\_no\_odd:**

**beqz t3, print\_max # Nếu không tìm thấy số lẻ, in số chẵn lớn nhất**

**# Không tìm thấy số chẵn thỏa điều kiện**

**not\_found:**

**li a7, 4**

**la a0, not\_found\_msg**

**ecall**

**j exit**

**# In số chẵn lớn nhất (khi không có số lẻ)**

**print\_max:**

**li a7, 4**

**la a0, no\_odd\_msg**

**ecall**

**addi t0, s0, -1 # t0 = size - 1 (phần tử cuối mảng)**

**slli t1, t0, 2 # Tính offset**

**add t2, s1, t1 # t2 = &array[size-1]**

**lw a0, 0(t2) # a0 = array[size-1] (số lớn nhất)**

**li a7, 1**

**ecall**

**j exit**

**# Xử lý lỗi kích thước mảng**

**Loi\_kich\_thuoc:**

**li a7, 4**

**la a0, error**

**ecall**

**# Kết thúc chương trình**

**exit:**

**li a7, 10**

**ecall**

* 1. Mô tả thuật toán và các chương trình con(nếu có)

**Mô tả thuật toán**

1. Nhập kích thước mảng từ người dùng
2. Kiểm tra tính hợp lệ của kích thước (phải từ 1 đến 100)
3. Nhập các phần tử của mảng
4. Sắp xếp mảng theo thứ tự tăng dần bằng thuật toán Bubble Sort
5. Tìm số chẵn lớn nhất nhỏ hơn mọi số lẻ trong mảng:
   * Do mảng đã được sắp xếp, số lẻ đầu tiên sẽ là số lẻ nhỏ nhất
   * Số chẵn cần tìm sẽ là số chẵn lớn nhất nhỏ hơn số lẻ nhỏ nhất
   * Đây chính là số mà đứng ngay trước số lẻ đầu tiên trong mảng đã sắp xếp.
6. Kiểm tra xem các trường hợp đặc biệt của mảng:
   * TH1: Mảng chỉ có toàn số chẵn, hiển nhiên số chẵn cần tìm sẽ là phần tử lớn nhất của mảng
   * TH2: Mảng chỉ có toàn số lẻ hoặc phần tử nhỏ nhất của mảng là số lẻ, trường hợp này thì chương trình sẽ in ra không tìm thấy số chẵn thỏa mãn.

**Các chương trình con**

* + main: Hàm chính; điểm khởi đầu của chương trình, thực hiện nhập kích thước, kiểm tra, đọc mảng, sắp xếp và sau đó xử lý tìm kiếm.
  + read\_loop: Vòng lặp nhập các phần tử của mảng từ bàn phím và lưu trữ chúng vào bộ nhớ.
  + sort\_array: Nhãn khởi đầu cho quá trình sắp xếp mảng.
  + outer\_loop: Vòng lặp bao bọc của thuật toán Bubble Sort, điều chỉnh số lần “bong bóng” cần thực hiện.
  + inner\_loop: Vòng lặp so sánh cặp phần tử liền kề, thực hiện hoán đổi nếu cần.
  + find\_result: Bắt đầu quá trình tìm kiếm số chẵn lớn nhất thỏa mãn điều kiện.
  + find\_loop: Duyệt mảng theo thứ tự tăng dần để tìm số lẻ đầu tiên.
  + check\_position: Sau khi tìm được số lẻ đầu tiên, kiểm tra xem có số chẵn đứng trước nó hay không (nếu số lẻ nằm ở vị trí 0 thì không có số chẵn hợp lệ).
  + handle\_no\_odd: Xử lý trường hợp mảng không chứa số lẻ nào.
  + print\_max: Trường hợp mảng toàn số chẵn, in ra số chẵn lớn nhất (là phần tử cuối của mảng đã được sắp xếp).
  + not\_found: Xử lý trường hợp số lẻ đầu tiên xuất hiện ở vị trí 0, khi đó không tồn tại số chẵn nào nhỏ hơn so với số lẻ.
  + Loi\_kich\_thuoc: Xử lý lỗi đầu vào khi kích thước mảng không thuộc khoảng (0, 100].
  + exit: Nhãn kết thúc chương trình; gọi syscall thoát.
  1. Ảnh chụp màn hình kết quả thực hiện chương trình

**TH1**: mảng không có số chẵn

Nhập mảng: arr[3] = {1,7,9}

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

Kết quả:

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

**TH2:** Mảng không có số lẻ - arr[3]={2,4,6}

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

Kết quả:

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

TH3: arr[4]={5,7,4,6}

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

Kết quả:

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

1. **Xâu kí tự – Câu 12**

***Đề bài:*** Nhập vào xâu ký tự. Đổi các ký tự chữ hoa thành chữ thường và chữ thường thành chữ hoa. Các ký tự khác giữ nguyên. Ví dụ: nhập vào xâu “xIn chAO 2023” => kết quả “XiN Chao 2023”.

* 1. Đoạn mã nguồn

**.data**

**buffer: .space 100 # Khai báo vùng nhớ 100 byte để lưu chuỗi nhập vào**

**.text**

**main:**

**# Đọc xâu ký tự từ người dùng**

**li a7, 8**

**la a0, buffer # Địa chỉ buffer để lưu chuỗi nhập**

**li a1, 100 # Số ký tự tối đa có thể đọc**

**ecall # Gọi hệ thống**

**# Bắt đầu quá trình đổi chữ hoa-thường**

**la t0, buffer # t0 = con trỏ trỏ tới đầu chuỗi**

**loop:**

**lb t1, 0(t0) # Load byte hiện tại từ chuỗi vào t1**

**beq t1, zero, done # Nếu gặp null thì nhảy đến done**

**# Kiểm tra xem ký tự có phải là chữ thường (a-z)**

**li t2, 97 # ASCII code của 'a'**

**li t3, 122 # ASCII code của 'z'**

**blt t1, t2, check\_upper # Nếu < 'a' thì kiểm tra chữ hoa**

**blt t3, t1, check\_upper # Nếu > 'z' thì kiểm tra chữ hoa**

**# Xử lý chữ thường -> chữ hoa**

**li t4, 32 # Hiệu số giữa chữ hoa và thường trong ASCII**

**sub t1, t1, t4 # Chuyển thành chữ hoa bằng cách trừ 32**

**sb t1, 0(t0) # Lưu ký tự đã chuyển đổi trở lại bộ nhớ**

**j next\_char # Nhảy đến ký tự tiếp theo**

**check\_upper:**

**# Kiểm tra xem ký tự có phải là chữ hoa (A-Z)**

**li t2, 65 # ASCII code của 'A'**

**li t3, 90 # ASCII code của 'Z'**

**blt t1, t2, next\_char # Nếu < 'A' thì bỏ qua**

**blt t3, t1, next\_char # Nếu > 'Z' thì bỏ qua**

**# Xử lý chữ hoa -> chữ thường**

**li t4, 32 # Hiệu số giữa chữ hoa và thường trongASCII**

**add t1, t1, t4 # Chuyển thành chữ thường bằng cách cộng 32**

**sb t1, 0(t0) # Lưu ký tự đã chuyển đổi trở lại bộ nhớ**

**next\_char:**

**addi t0, t0, 1 # Di chuyển con trỏ đến ký tự tiếp theo trong chuỗi**

**j loop # Lặp lại vòng lặp**

**done:**

**# In ra chuỗi đã được chuyển đổi**

**li a7, 4**

**la a0, buffer**

**ecall**

**# Kết thúc chương trình**

**li a7, 10**

**ecall**

* 1. Mô tả thuật toán và các chương trình con(nếu có)

**Mô tả thuật toán:**

1. Nhập một chuỗi từ người dùng (tối đa 99 ký tự + ký tự null)
2. Duyệt từng ký tự trong chuỗi cho đến khi gặp ký tự null (kết thúc chuỗi)
3. Với mỗi ký tự:
   * Nếu là chữ thường (a-z), chuyển thành chữ hoa tương ứng
   * Nếu là chữ hoa (A-Z), chuyển thành chữ thường tương ứng
   * Nếu không phải chữ cái, giữ nguyên
4. In ra chuỗi đã được chuyển đổi

**Các chương trình con:**

**1. main:**

* Sử dụng syscall 8 để đọc chuỗi từ người dùng
* Lưu chuỗi vào vùng nhớ buffer
* Giới hạn số ký tự đọc là 100 (bao gồm ký tự null)

**2. loop:**

Chức năng: Vòng lặp chính để duyệt từng ký tự trong chuỗi

* Duyệt từng ký tự trong chuỗi
* Load byte hiện tại từ vị trí con trỏ (t0) vào thanh ghi t1
* Kiểm tra nếu gặp ký tự null (t1 = 0), kết thúc vòng lặp
* Kiểm tra xem ký tự hiện tại có phải chữ thường không
  + Nếu là chữ thường, chuyển thành chữ hoa
  + Nếu không phải, nhảy đến check\_upper để kiểm tra chữ hoa

**Kiểm tra và chuyển đổi chữ thường thành chữ hoa**

* Kiểm tra nếu ký tự nằm trong khoảng ASCII từ 'a' (97) đến 'z' (122)
* Nếu đúng, trừ 32 để có mã ASCII của chữ hoa tương ứng
* Lưu ký tự đã chuyển đổi trở lại vị trí ban đầu trong buffer

**3. check\_upper:**

Chức năng: Kiểm tra và chuyển đổi chữ hoa thành chữ thường

* Kiểm tra nếu ký tự nằm trong khoảng ASCII từ 'A' (65) đến 'Z' (90)
* Nếu đúng, cộng 32 để có mã ASCII của chữ thường tương ứng
* Lưu ký tự đã chuyển đổi trở lại vị trí ban đầu trong buffer

**4. next\_char:**

Chức năng: Di chuyển đến ký tự tiếp theo trong chuỗi

* Tăng con trỏ chuỗi lên 1 byte
* Quay lại vòng lặp chính (loop)

**5. done:**

Chức năng: Hiển thị kết quả và kết thúc chương trình

* In chuỗi đã được chuyển đổi bằng syscall 4
* Kết thúc chương trình bằng syscall 10
  1. Ảnh chụp màn hình kết quả thực hiện chương trình

Nhập xâu: xinChao 2023

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

Kết quả:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, hàng

Nội dung do AI tạo ra có thể không chính xác.

Nhập xâu: Le Thanh An

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

Kết quả:

Ảnh có chứa ảnh chụp màn hình, văn bản, hàng

Nội dung do AI tạo ra có thể không chính xác.