Trạng thái	Đã xong
Bắt đầu vào lúc	Chủ Nhật, 30 tháng 3 2025, 11:00 PM
Kết thúc lúc	Chủ Nhật, 30 tháng 3 2025, 11:12 PM
Thời gian thực hiện	11 phút 16 giây

Câu hỏi 1

Đúng

[Tiếng Việt]

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

[English]

A string is a palindrome if it reads the same forward and backward. For example: "eye", "noon", "abcba", ...

Write a function to check if a given string is a palindrome

Input:

- const char* str: the string to be checked. str only contains lowercase letters

Output:

- bool: true if str is a palindrome, false otherwise

For example:

Test	Result
`const char* str = "abba";` `cout << isPalindrome(str);`	1
`const char* str = "axz";` `cout << isPalindrome(str);`	0

Answer: (penalty regime: 0 %)

Reset answer

```
#include <string.h>

/**
 * Kiểm tra xem một chuỗi có phải là palindrome hay không
 *
 * @param str Chuỗi cần kiểm tra, chỉ bao gồm chữ cái thường
 * @return true nếu chuỗi là palindrome, ngược lại false
 */
bool isPalindrome(const char* str) {
    // Nếu chuỗi là NULL, không phải palindrome
    if (str == NULL) {
        return false;
    }

    int length = strlen(str);

    // Duyệt từ hai đầu vào giữa
    for (int i = 0; i < length / 2; i++) {
        // Nếu có bất kỳ cặp ký tự nào không trùng nhau
        if (str[i] != str[length - 1 - i]) {
            return false;
        }
    }

    // Nếu tất cả các cặp ký tự đều trùng nhau
    return true;
}
```

	Test	Expected	Got	
✓	`const char* str = "abba";` `cout << isPalindrome(str);`	1	1	✓
✓	`const char* str = "axz";` `cout << isPalindrome(str);`	0	0	✓

Passed all tests! ✓

Câu hỏi **2**

Đúng

[Tiếng Việt]

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

[English]

A natural number n is special if and only if n is a prime number and the sum of all the digits of n is also a prime number. Write a function that determines if a natural number is a special or not.

Input:

int n: a natural number n. 0 <= n <= 1000

Output:

bool: return true if n is special, return false otherwise

For example:

Test	Input	Result
int n; cin >> n; cout << isSpecialNumber(n);	23	1
int n; cin >> n; cout << isSpecialNumber(n);	7	1

Answer: (penalty regime: 0 %)

Reset answer

```
#include <math.h>
/*
 * Kiểm tra một số có phải là số nguyên tố hay không
 * @param n Số cần kiểm tra
 * @return true nếu n là số nguyên tố, ngược lại false
 */
bool isPrime(int n) {
    // Các trường hợp đặc biệt
    if (n <= 1) {
        return false;  // 0 và 1 không phải là số nguyên tố
    }
    if (n <= 3) {
        return true;   // 2 và 3 là số nguyên tố
    }
    if (n % 2 == 0 || n % 3 == 0) {
        return false;  // Chia hết cho 2 hoặc 3 không phải số nguyên tố
    }

    // Kiểm tra các ước số từ 5 đến căn bậc hai của n
    // Sử dụng phương pháp tối ưu 6k±1
    int sqrtN = (int)sqrt(n);
    for (int i = 5; i <= sqrtN; i += 6) {
        if (n % i == 0 || n % (i + 2) == 0) {
            return false;
        }
    }

    return true;
}

/*
```

```
 * Tính tổng các chữ số của một số
 * @param n Số cần tính tổng chữ số
 * @return Tổng các chữ số của n
 */
int sumOfDigits(int n) {
    int sum = 0;
    while (n > 0) {
        sum += n % 10;  // Lấy chữ số cuối cùng
        n /= 10;        // Bỏ chữ số cuối cùng
    }
    return sum;
}

/*
 * Kiểm tra một số có phải là số đặc biệt hay không
 * Số đặc biệt là số nguyên tố và tổng các chữ số của nó cũng là số nguyên tố
 *
 * @param n Số cần kiểm tra
 * @return true nếu n là số đặc biệt, ngược lại false
 */
bool isSpecialNumber(int n) {
    // Kiểm tra n có phải số nguyên tố không
    if (!isPrime(n)) {
        return false;
    }

    // Tính tổng các chữ số của n
    int sum = sumOfDigits(n);

    // Kiểm tra tổng các chữ số có phải số nguyên tố không
    return isPrime(sum);
```

	Test	Input	Expected	Got	
✓	int n; cin >> n; cout << isSpecialNumber(n);	23	1	1	✓
✓	int n; cin >> n; cout << isSpecialNumber(n);	7	1	1	✓

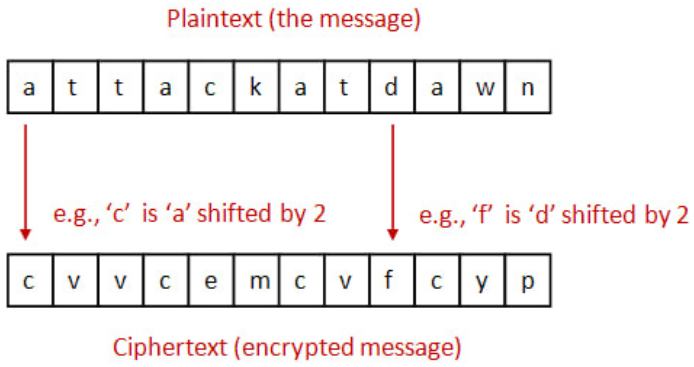
Passed all tests! ✓

Câu hỏi **3**

Đúng

[Tiếng Việt]

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

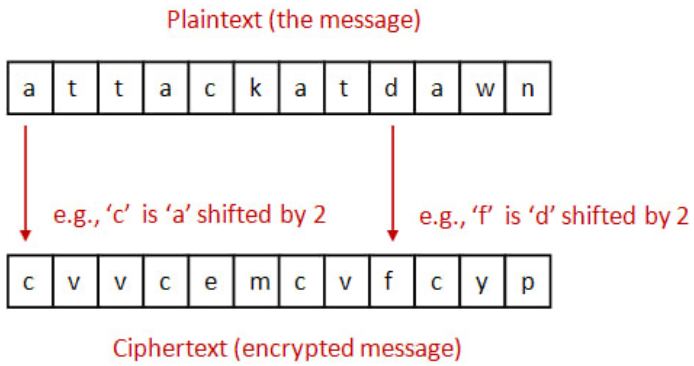
Đầu ra:

- Hàm không trả về.
- Chuỗi ký tự truyền vào **text** sẽ thay đổi trực tiếp trong hàm.

[English]

Write a function to encrypt and a function to decrypt a text string using Caesar Cipher technique.

In this technique, to encrypt a string we need a parameter of type integer called 'shift'.



The **encrypt** method will shift each letter by some fixed number of position (determined by the parameter 'shift') to the right in the alphabet. For example, when 'shift' is 3, 'a' will be replaced by 'd', 'b' will become 'e', ... , 'z' will become 'c'.

The **decrypt** method will receive an encoded string and a shift value and it will decode this string to get the original string, which means shifting each character to the left in the alphabet.

Input:

- **char* text:** the text string that needs to be encrypted or decrypted. text only contains lowercase and uppercase ASCII letters
- **int shift**: the shift value in Caesar Cipher technique

Output:

- The function returns nothing.
- The input parameter **text** will be updated in-place.

For example:

Test	Input	Result

int n, shift;
cin >> n >> shift;
char* text = new char[n+1];
for(int i = 0; i < n; i++) cin >> text[i];
text[n] = 0;
 	
encrypt(text, shift);
cout << text << '\n';
decrypt(text, shift);
cout << text;
 	
delete[] text;
	6 3
	aczDYZ
		dfcGBC
		aczDYZ

int n, shift;
cin >> n >> shift;
char* text = new char[n+1];
for(int i = 0; i < n; i++) cin >> text[i];
text[n] = 0;
 	
encrypt(text, shift);
cout << text << '\n';
decrypt(text, shift);
cout << text;
 	
delete[] text;
	16 25
	programmingisfun
		oqnfqzllhmfhretm
		programmingisfun

Answer: (penalty regime: 0 %)

Reset answer

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

/*
 * Mã hóa một chuỗi ký tự theo phương pháp Caesar Cipher
 * @param text Chuỗi ký tự cần mã hóa (chỉ chữ cái thường và hoa)
 * @param shift Giá trị dịch chuyển (số nguyên)
 */
void encrypt(char* text, int shift) {
    int len = strlen(text);

    // Đảm bảo shift nằm trong khoảng [0, 25]
    shift = shift % 26;
    if (shift < 0) {
        shift += 26;
    }

    for (int i = 0; i < len; i++) {
        // Chỉ xử lý chữ cái
        if (isalpha(text[i])) {
            // Xử lý chữ cái hoa
            if (isupper(text[i])) {
                text[i] = 'A' + (text[i] - 'A' + shift) % 26;
            }
            // Xử lý chữ cái thường
            else {
                text[i] = 'a' + (text[i] - 'a' + shift) % 26;
            }
        }
        // Các ký tự khác giữ nguyên
    }
}

/*
 * Giải mã một chuỗi ký tự đã được mã hóa theo phương pháp Caesar Cipher
 * @param text Chuỗi ký tự cần giải mã (chỉ chữ cái thường và hoa)
 * @param shift Giá trị dịch chuyển (số nguyên)
 */
void decrypt(char* text, int shift) {
    // Giải mã chính là mã hóa với giá trị shift ngược lại
    // Ví dụ: Mã hóa với shift = 3, thì giải mã là mã hóa với shift = -3 hoặc 23
```

```
43      encrypt(text, -shift);
44  }
```

	Test	Input	Expected	Got	

✓	int n, shift;
	cin >> n >> shift;
	char* text = new char[n+1];
	for(int i = 0; i < n; i++) cin >> text[i];
	text[n] = 0;
	
	encrypt(text, shift);
	cout << text << '\n';
	decrypt(text, shift);
	cout << text;
	
	delete[] text;
		6 3
		aczDYZ
			dfcGBC
			aczDYZ
				dfcGBC
				aczDYZ
					✓

✓	int n, shift;
	cin >> n >> shift;
	char* text = new char[n+1];
	for(int i = 0; i < n; i++) cin >> text[i];
	text[n] = 0;
	
	encrypt(text, shift);
	cout << text << '\n';
	decrypt(text, shift);
	cout << text;
	
	delete[] text;
		16 25
		programmingisfun
			oqnfqzllhmfhretm
			programmingisfun
				oqnfqzllhmfhretm
				programmingisfun
					✓

Passed all tests! ✓

Câu hỏi **4**

Đúng

[Tiếng Việt]

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

Chú ý: arr[i] nằm trong khoảng từ [0, 1000]

[English]

Write a function that determines if the elements in the given array is unique

Input:

- int* arr: array of integer
- int n: the size of the array

Output:

- bool: return true if the elements in arr is unique, otherwise return false

Note: arr[i] is in the range of [0, 1000]

For example:

Test	Input	Result

int n;
cin >> n;
int* arr = new int[n];
for(int i = 0; i < n; i++) {
  cin >> arr[i];
}
cout << checkElementsUniqueness(arr, n);
delete[] arr;
	5
	2 5 13 5 2
		0

int n;
cin >> n;
int* arr = new int[n];
for(int i = 0; i < n; i++) {
  cin >> arr[i];
}
cout << checkElementsUniqueness(arr, n);
delete[] arr;
	3
	17 10 25
		1

Answer: (penalty regime: 0 %)

Reset answer

```
#include <stdbool.h>

bool checkElementsUniqueness(int* arr, int n) {
    // Sử dụng mảng đánh dấu vì giá trị của arr[i] nằm trong khoảng [0, 1000]
    bool visited[1001] = {false}; // Khởi tạo tất cả phần tử là false

    for (int i = 0; i < n; i++) {
        // Nếu phần tử đã xuất hiện trước đó, mảng không duy nhất
        if (visited[arr[i]]) {
            return false;
        }

        // Đánh dấu phần tử này đã xuất hiện
        visited[arr[i]] = true;
    }

```

```
    // Nếu không có phần tử nào lặp lại, mảng là duy nhất
    return true;
}
```

	Test	Input	Expected	Got	

✓	int n;
	cin >> n;
	int* arr = new int[n];
	for(int i = 0; i < n; i++) {
	    cin >> arr[i];
	}
	cout << checkElementsUniqueness(arr, n);
	delete[] arr;
		5
		2 5 13 5 2
			0	0	✓

✓	int n;
	cin >> n;
	int* arr = new int[n];
	for(int i = 0; i < n; i++) {
	    cin >> arr[i];
	}
	cout << checkElementsUniqueness(arr, n);
	delete[] arr;
		3
		17 10 25
			1	1	✓

Passed all tests! ✓

Câu hỏi **5**

Đúng

[Tiếng Việt]

Cho một số thập phân dương làm đầu vào, chúng ta cần triển khai hàm

```
long int decimalToBinary(int decimal_number){}
```

để chuyển đổi số thập phân dương đã cho thành số nhị phân tương đương.

Xin lưu ý rằng bạn không thể 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;

[English]

Given a positive decimal number as input, we need to implement function

```
long int decimalToBinary(int decimal_number){}
```

to convert the given positive decimal number into equivalent binary number.

Please note that you can't using key work for, while, goto (even in variable names, comment).

For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result
cout << decimalToBinary(20);	10100

Answer:

Reset answer

```
long int decimalToBinary(int decimal_number) {
    // Trường hợp cơ bản: nếu số thập phân là 0 hoặc 1
    if (decimal_number == 0) {
        return 0;
    }
    if (decimal_number == 1) {
        return 1;
    }

    // Công thức đệ quy: chuyển đổi phần còn lại và thêm chữ số cuối cùng
    // Ý tưởng: decimal_number % 2 là bit cuối cùng của số nhị phân
    // decimal_number / 2 là phần còn lại cần chuyển đổi
    return decimalToBinary(decimal_number / 2) * 10 + (decimal_number % 2);
}

```

	Test	Expected	Got	
✓	`cout << decimalToBinary(20);`	10100	10100	✓
✓	`cout << decimalToBinary(192);`	11000000	11000000	✓

Passed all tests! ✓