一、字符串处理与编码

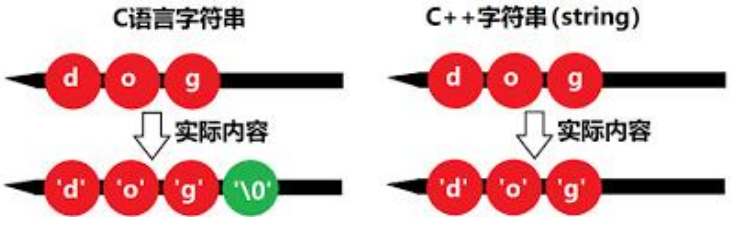
（一）考点分析

分类	重点考点	考察频率	难度
基本概念	字符串定义与存储	高	★
	子串与子序列	高	★★
基本操作	大小写转换	高	★★
	字符统计	中	★★
	逆序操作	中	★★
高级操作	循环移位判断	中	★★★
	字符串映射	中	★★★
	规则展开	低	★★★★
算法应用	最长公共子序列(LCS)	高	★★★★
	字典序处理	中	★★★
特殊处理	字符过滤	中	★★
	周期性问题	低	★★
易错点	边界条件	高	★★
	ASCII 操作	高	★★
	大小写敏感	中	★★

（二）核心知识点

1、基本概念

(1)字符串定义与存储



- **核心知识点：**

  C++ 中字符串可用 **char 数组或 string 类**存储

  char 数组以 **'\0'** 结尾，string 类自动管理内存

- **黄金法则：**使用 string 类更安全便捷，避免手动内存管理
- **高频公式：**

```
// char 数组定义
char str1[20] = "Hello";
// string 类定义
#include <string>
string str2 = "World";
```

- **易错点：**

  char 数组未预留 '\0' 空间导致越界

  混淆 strlen()（不包含 '\0'）和 sizeof()（包含 '\0'）

- **程序模板：**

```
char s1[100];
cin.getline(s1, sizeof(s1));  // 安全输入
string s2;
getline(cin, s2);  // 整行读取
```

(2)子串与子序列

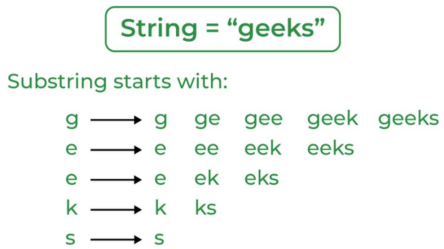
- **核心知识点：**

  子串：**连续**字符序列（如 "abc" 的子串 "bc"）

  子序列：**非连续**但顺序一致（如 "abc" 的子序列 "ac"）

- 黄金法则：非空子串数量公式：**n(n+1)/2**（长度为 n 的字符串）
- **高频公式：**

```
// 获取子串
string s = "abcdef";
string sub = s.substr(1, 3);  // "bcd"
```



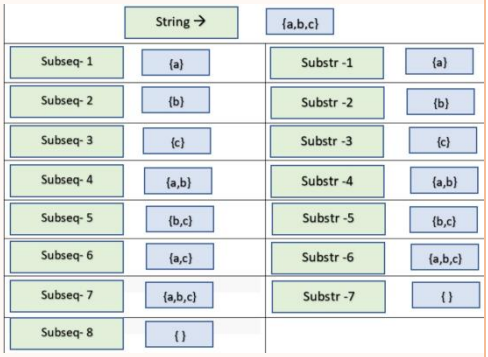
- **易错点：**

  混淆**子串（连续）**和**子序列（非连续）**

  忘记**空串**也是合法子串

- **程序模板：**

```
// 遍历所有子串
for(int i=0; i<s.size(); ++i) {
  for(int j=1; j<=s.size()-i; ++j) {
    cout << s.substr(i, j) << endl;
  }
}
```



2、基本操作

(1)大小写转换

- **核心知识点：**

  ASCII 差值：**'a'-'A'=32**

  库函数 **tolower()/toupper()**

- **黄金法则：**转换前先判断字符类型

- **高频公式：**

```
// 小写转大写
if(ch >= 'a' && ch <= 'z')
    ch -= 32;  // 或 ch = toupper(ch);
```

Dec	Hex	Chr	Dec	Hex	Chr
64	40	@	96	60	`
65	41	A	97	61	a
66	42	B	98	62	b
67	43	C	99	63	c
68	44	D	100	64	d
69	45	E	101	65	e
70	46	F	102	66	f
71	47	G	103	67	g
72	48	H	104	68	h
73	49	I	105	69	i
74	4A	J	106	6A	j
75	4B	K	107	6B	k
76	4C	L	108	6C	l
77	4D	M	109	6D	m
78	4E	N	110	6E	n
79	4F	O	111	6F	o
80	50	P	112	70	p
81	51	Q	113	71	q
82	52	R	114	72	r
83	53	S	115	73	s
84	54	T	116	74	t
85	55	U	117	75	u
86	56	V	118	76	v
87	57	W	119	77	w
88	58	X	120	78	x
89	59	Y	121	79	y
90	5A	Z	122	7A	z
91	5B	[	123	7B	{
92	5C	\	124	7C	|
93	5D	]	125	7D	}
94	5E	^	126	7E	~
95	5F	_	127	7F	DEL

- **易错点：**

  非字母字符错误转换（如数字、标点）

  忘记字符在表达式中自动转整数

- **程序模板：**

```
string s = "Hello123";
for(char &c : s) {
   if(isalpha(c))
        c = tolower(c);
}  // 结果："hello123"
```

(2)字符统计

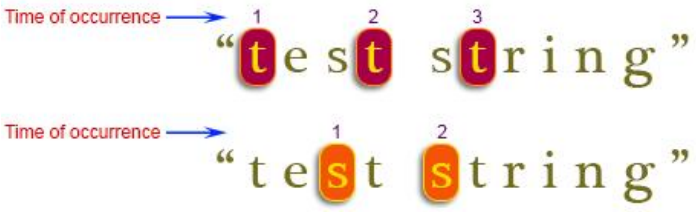
- **核心知识点：**

  使用数组或 map 统计频次

  ASCII 码直接索引（count[ch]++）

- **黄金法则：**

  初始化统计数组为 0



- **高频公式：**

```
int count[256] = {0};  // 初始化
for(char c : str)
    count[c]++;
```

- **易错点：**

  未考虑扩展 ASCII（128-255）

  负数索引导致越界

- **程序模板：**

```
// 找第一个唯一字符
int cnt[256] = {0};
for(char c : s) cnt[c]++;
for(char c : s) {
   if(cnt[c] == 1) {
```

```
            cout << c << endl;
            break;
        }
    }
```

(3)逆序操作

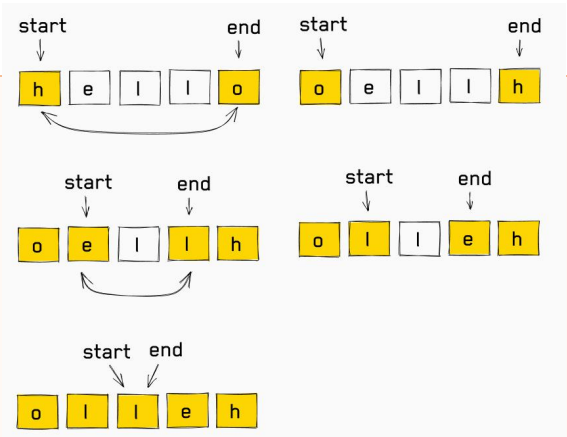
- **核心知识点：**

  双指针交换法

  STL 的 reverse() 函数

- **黄金法则：**终止条件 left < right

- **高频公式：**




```
// 双指针逆序
int l=0, r=s.size()-1;
while(l < r) {
    swap( s[l++] ,  s[r--] );
}
```

- **易错点：**

  偶数长度时中间元素未处理

  忘记字符串以 '\0' 结尾

- **程序模板：**

```
#include <algorithm>
reverse(s.begin(), s.end());   // STL 方法
```

3、高级操作

(1)循环移位判断

- **核心知识点：**y 是 x 的循环旋转 ⇔ y 是 x+x 的子串

- **黄金法则：**先检查长度是否相等
- **高频公式：**

```
bool isRotation(string x, string y) {
    if(x.size() != y.size()) return false;
    return (x + x).find(y) != string::npos;
}//string::npos 是 string 类的一个常量，表示“无匹配”或“查找失败”。
```

- **易错点：**

  未处理空字符串

  混淆子串（连续）和子序列（非连续）

- **程序模板：**

```
string x = "abcde", y = "cdeab";
cout << (x+x).find(y) != string::npos;   // 输出 1
```

(2)字符串映射

- **核心知识点：**

  建立字符映射关系

  大小写敏感处理

- **黄金法则：**统一转换为小写再映射
- **高频公式：**

```
// 电话字母映射
string map = "22233344455566677778889999";
char ch = 'A';
int index = ch - 'A';
char digit = map[index];   // '2'
```

- **易错点：**

映射表索引越界

未处理非字母字符

tel	C	C	F	-	N	O	I	P	-	2	0	1	1
tel[i]-‘A’	2	2	5		13	14	8	15					
map[tel[i]-‘A’]	2	2	3		6	6	4	7					

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
map	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6	7

- **程序模板：**

```
string map = "22233344455566677778889999";
string input = "CCF-NOIP";
for(char c : input) {
  if(isalpha(c))
      cout << map[toupper(c)-'A'];
  else
      cout << c;
}  // 输出"223-NOIP"
```

(3)规则展开

- **核心知识点：**处理连字符展开规则（如 "a-d" → "abcd"）
- **黄金法则**：检查边界：两侧字符类型相同且左 < 右
- **高频公式：**

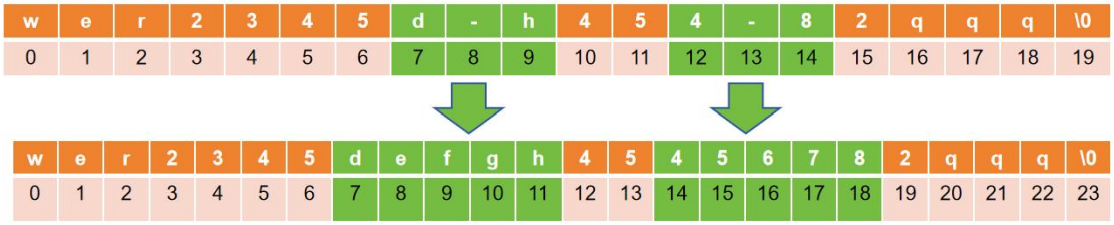
```
if(c=='-' && isalnum(left) && isalnum(right) && left<right) {
  for(char ch=left+1; ch<right; ch++)
      result += ch;
```

```
}
```

- **易错点：**

  未处理开头/结尾的连字符

  大小写字母混合（如 "A-d"）



- **程序模板：**

```
string expand(string s) {
    string res;
    for(int i=0; i<s.size(); ++i) {
        if(s[i]=='-' && i>0 && i+1<s.size()) {
            char a=s[i-1], b=s[i+1];
            if((islower(a)&&islower(b) || isdigit(a)&&isdigit(b)) && a<b) {
                res.pop_back();   // 移除前一个字符
                for(char c=a; c<=b; c++) res += c;
                i++;   // 跳过已处理的字符
                continue;
            }
        }
        res += s[i];
    }
    return res;
```

```
}
```

4、算法应用

(1)最长公共子序列（LCS）

- **核心知识点：**

  动态规划：**dp[i][j] = dp[i-1][j-1] + 1（当 x[i]==y[j]）**

- **黄金法则：**初始化首行首列为 0

a	b	c	d	a	b	c	d

- **高频公式：**

c	d	a	b

```
// LCS 动态规划核心
if(x[i-1] == y[j-1])
    dp[i][j] = dp[i-1][j-1] + 1;
else
    dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
```

	j=1	j=2	j=3	j=4
i=1	0	0	1	1
i=2	0	0	1	2
i=3	1	1	1	2
i=4	1	2	2	2
i=5	1	2	3	3
i=6	1	2	3	4
i=7	1	2	3	4
i=8	1	2	3	**4**

- **易错点：**

  混淆 LCS（子序列）和最长公共子串（需连续）

  索引从 1 开始但字符串从 0 访问

- **程序模板：**

```
int lcs(string x, string y) {
  int m=x.size(), n=y.size();
  vector<vector<int>> dp(m+1, vector<int>(n+1, 0));
  for(int i=1; i<=m; ++i) {
    for(int j=1; j<=n; ++j) {
      if(x[i-1] == y[j-1])
          dp[i][j] = dp[i-1][j-1] + 1;
      else
```

```
                dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
            }
        }
        return dp[m][n];
    }
```

(2)字典序处理

- **核心知识点：**

  字典序比较：str1.compare(str2) < 0 表示 str1 更小

- **黄金法则**：循环位移的最小字典序可用 s+s 构造

i	0	1	2	3	4	5	6	7
原	C	B	B	A	D	A	D	A
现	A	C	B	B	A	D	A	D

- **高频公式：**

```
// 找最小字典序循环位移
string minRotation(string s) {
    string t = s + s;
    int n = s.size(), idx = 0;
    for(int i=1; i<n; ++i) {
        if(t.substr(i, n) < t.substr(idx, n))
            idx = i;
    }
    return t.substr(idx, n);
}
```

- **易错点：**

  未处理全相同字符的情况

  子串截取长度错误

- **程序模板：**

```
string s = "CBBADADA";
string t = s + s;
for(int i=0; i<s.size(); ++i) {
   cout << t.substr(i, s.size()) << endl;
}
// 输出最小："ACBBADAD"
```

5、特殊处理

(1)字符过滤

- **核心知识点：**

  双指针法：快慢指针原地修改

- **黄金法则：**慢指针 j 指向新字符串末尾

- **高频公式：**

```
int j = 0;
for(int i=0; i<s.size(); ++i) {
   if(!isdigit(s[i]))   // 保留非数字
        s[j++] = s[i];
}
s.resize(j);   // 重要！调整大小
```

i	0	1	2	3	4	5
s[]	a	1	b	2	c	3

i	0	1	2
s[]	a	b	c

- **易错点：**

  忘记最后 resize()

未处理字符串结束符 '\0'（char 数组）

- **程序模板：**

```
string filter(string s) {
    string result;
    copy_if(s.begin(), s.end(), back_inserter(result),
            [](char c){ return !isdigit(c); });
    return result;
}
```

(2)周期性问题

- **核心知识点：**周期 T 需满足 **s[i] == s[i%T]**
- **黄金法则：**周期必须是字符串长度的约数
- **高频公式：**

```
// 判断最小周期
int n = s.size();
for(int T=1; T<=n/2; T++) {
    if(n % T != 0) continue;
    bool isPeriod = true;
    for(int i=T; i<n; i++) {
        if(s[i] != s[i%T]) {
            isPeriod = false;
            break;
        }
    }
    if(isPeriod) return T;
}
```

i	0	1	2	3	4	5	6	7	8
s	a	b	c	a	b	c	a	b	c

```
return n;   // 自身为周期
```

- 易错点：

  未检查 n%T==0

  循环变量初始值错误

- 程序模板：

```
string s = "abcabcabc";
int n = s.size();
for(int T=1; T<=n/2; T++) {
  if(n%T) continue;
  bool valid = true;
  for(int i=0; i<n; ++i) {
    if(s[i] != s[i%T]) {
      valid = false;
      break;
    }
  }
  if(valid) cout << "Period: " << T;
}
```

i	0	1	2	3	4	5	6	7	8
s	a	b	c	a	b	c	a	b	c

6、易错点专题

(1)边界条件

- **核心防御：**

  先检查**空字符串**：if(s.empty()) return;

  循环时验证**索引**：if(i+1 < s.size())

- **经典错误：**

```
// 错误：空字符串访问 s[0]
if(s[0] == 'A')  // 可能崩溃
// 正确：先判空
if(!s.empty() && s[0]=='A')
```

(2)ASCII 操作

- **必须掌握：**

  字符转整数：digit = ch - '0'

  字母转索引：index = toupper(ch) - 'A'

- **易错陷阱：**

```
// 错误：忘记减'0'
int num = '5';  // 得到 53（ASCII 值）
// 正确：
int num = '5' - '0';  // 得到 5
```

(3) 大小写敏感

- **处理原则：**

  比较前**统一大小写**

  使用自定义**比较函数**

- **安全比较：**

```
bool equalIgnoreCase(string a, string b) {
  if(a.size() != b.size()) return false;
  for(int i=0; i<a.size(); ++i) {
    if(tolower(a[i]) != tolower(b[i]))
        return false;
  }
```

```
    return true;
}
```

典型错误：

```
// 错误：直接比较大小写混合字符串
if("Hello" == "hello")  // 返回 false
```

小结

- **黄金法则：**

  操作字符串前先判空

  使用 string 类避免内存管理错误

  字符操作注意 ASCII 码特性

- **高频技巧：**

```
// 遍历字符串的三种安全方式
for(int i=0; i<s.size(); ++i)  // 索引访问
for(char c : s)                // 范围循环
for(auto it=s.begin(); it!=s.end(); ++it)  // 迭代器
// 快速统计技巧
vector<int> count(256, 0);
for(char c : s) count[c]++;
```

- **致命陷阱：**

  未初始化字符数组

  越界访问（特别是 char 数组）

  混淆 length()（不含 '\0'）和 sizeof()（含 '\0'）

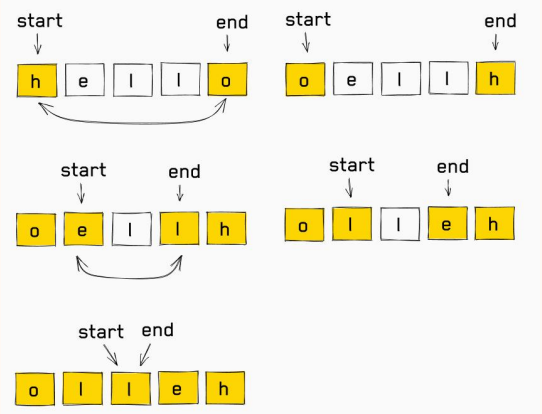
（三）真题强化

1、2007 年第 27 题（字符串逆序）

完善程序：

（求字符的逆序）下面的程序的功能是输入若干行字符串，每输入一行，就按逆序输出该行，最后键入 -1 终止程序。请将程序补充完整。

```
#include <iostream.h>
#include <string.h>
int maxline = 200, kz;
int reverse( char s[] )
{
    int i, j, t;
    for ( i = 0, j = strlen( s ) - 1; i < j; 【①】 , 【②】 )
    {
        t = s[i]; s[i] = s[j]; s[j] = t;
    }
    return(0);
}
int main()
{
    char line[100];
    cout << "continue? -1 for end." <<endl;
    cin >> kz;
    while(【③】)
    {
        cin >> line;
        【④】;
```



```
            cout << line << endl;
            cout << "continue ? -1 for end." << endl;
            cin >> kz;
        }
    }
```

答案：

①：i++

②：j--

③：kz != -1

④：reverse(line)

解析：

- reverse 函数：实现字符串反转功能

  i 从字符串开头向后移动，j 从字符串末尾向前移动

  当 i < j 时，交换 s[i] 和 s[j] 的值

  每次交换后，i 需增加 1（i++），j 需减少 1（j--）

- main 函数：控制程序流程

  ③：当输入值 kz 不等于 -1 时（kz != -1），继续循环

  ④：调用 reverse(line) 函数反转输入的字符串

2、2008 年第 9 题（非空字串数量）

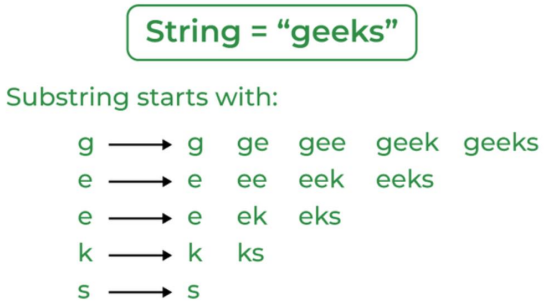
设字符串 S="Olympic"，S 的非空子串的数目是（ ）。

A. 28

B. 29

C. 16

D. 17



答案： A

解析：

- 子串定义：原字符串中任意一段连续的字符序列
- 计算公式：**长度为 n 的字符串有 n(n+1)/2 个非空子串**
- 计算过程：字符串 "Olympic" 长度 n=7，非空子串数量 = 7×8/2=28
- **验证：**
- 长度 1 子串：7 个（O, l, y, m, p, i, c）
- 长度 2 子串：6 个（Ol, ly, ym, mp, pi, ic）
- ...
- 长度 7 子串：1 个（Olympic）
- 总和：7+6+5+4+3+2+1=28

3、2010 年第 25 题（字符串）

阅读程序写结果：

```
#include <iostream>
#include <string>
using namespace std;
int main(){
    string s;
    char m1, m2;
    int i;
    getline(cin, s);
    m1 = ' ';
    m2 = ' ';
    for (i = 0; i < s.length(); i++)
        if (s[i] > m1) {
            m2 = m1;
            m1 = s[i];
```

```
            }
            else if (s[i] > m2)
                m2 = s[i];
        cout << int(m1) << ' ' << int(m2) << endl;
        return 0;
    }
```

Dec	Hex	Chr
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Chr
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	DEL

输入：Expo 2010 Shanghai China，输出：

提示：

字符　空格　'0'　'A'　'a'

ASCII 码　32　48　65　97

答案：　120 112

解析：

- 程序功能：**找出字符串中 ASCII 值最大的两个字符**
- 初始化：m1 和 m2 均为空格（ASCII 32）
- 字符处理流程：
  - 遍历字符串中每个字符
  - 当前字符 > m1 时：m2 继承原 m1，m1 更新为当前字符
  - 当前字符 ≤ m1 但 > m2 时：m2 更新为当前字符
- 输入字符串分析：
  - 字符 'x'：ASCII 120 → 更新 m1=120, m2=32
  - 字符 'p'：ASCII 112 → 更新 m2=112
  - 其他字符（如 'S'=83, 'h'=104）均小于 112
- 输出结果：最大 ASCII 值 120（'x'），次大值 112（'p'）

4、2012 年第 19 题（非空子串数量）

原字符串中任意一段连续的字符所组成的新字符串称为子串。则字符

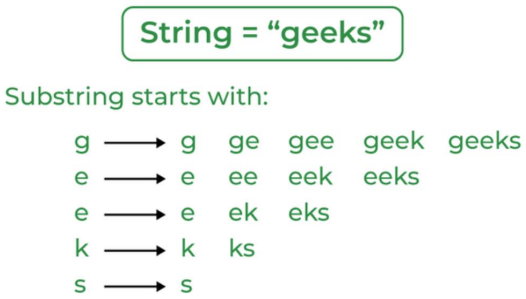
AAABBBCCC 共有（ ）个不同的非空子串。

A. 3

B. 12

C. 36

D. 45



答案： C

解析：

- 子串分类统计：
  - 单一字符段：
    - A 段（长度 3）：子串 "A", "AA", "AAA" → 3 个
    - B 段（长度 3）：子串 "B", "BB", "BBB" → 3 个
    - C 段（长度 3）：子串 "C", "CC", "CCC" → 3 个
  - 跨段子串：
    - A-B 交界：3 起点 × 3 终点 = 9 个（如 "AAB", "ABBB"）
    - B-C 交界：3 起点 × 3 终点 = 9 个（如 "BCC", "BBCC"）
    - A-B-C 交界：3 起点（A）× 3 终点（C）= 9 个（如 "AAABBB", "ABBBCCC"）
- 总计：**3 (A) + 3 (B) + 3 (C) + 9 (AB) + 9 (BC) + 9 (ABC) = 36**
- 对比总子串数：长度 9 的字符串总子串数为 **9×10 / 2 =45**，因有重复子串（如多个 "A"），实际不同子串为 36

5、2014 年第 25 题（字符串）

阅读程序写结果：

```
#include <iostream>
#include <string>
using namespace std;
int main()
```

```
{
    string st;
    int i, len;
    getline(cin, st);
    len = st.size();
    for (i = 0; i < len; i++)
        if (st[i] >= 'a' && st[i] <= 'z')
            st[i] = st[i] - 'a' + 'A';
    cout << st << endl;
    return 0;
}
```

Dec	Hex	Chr	Dec	Hex	Chr
64	40	@	96	60	`
65	41	A	97	61	a
66	42	B	98	62	b
67	43	C	99	63	c
68	44	D	100	64	d
69	45	E	101	65	e
70	46	F	102	66	f
71	47	G	103	67	g
72	48	H	104	68	h
73	49	I	105	69	i
74	4A	J	106	6A	j
75	4B	K	107	6B	k
76	4C	L	108	6C	l
77	4D	M	109	6D	m
78	4E	N	110	6E	n
79	4F	O	111	6F	o
80	50	P	112	70	p
81	51	Q	113	71	q
82	52	R	114	72	r
83	53	S	115	73	s
84	54	T	116	74	t
85	55	U	117	75	u
86	56	V	118	76	v
87	57	W	119	77	w
88	58	X	120	78	x
89	59	Y	121	79	y
90	5A	Z	122	7A	z
91	5B	[	123	7B	{
92	5C	\	124	7C	|
93	5D	]	125	7D	}
94	5E	^	126	7E	~
95	5F	_	127	7F	DEL

输入：Hello, my name is Lostmonkey.

输出：_____

答案： HELLO, MY NAME IS LOSTMONKEY.

解析：

- 程序功能：**将字符串中所有小写字母转换为大写字母**
- 转换规则：
  - 字符在 'a'~'z' 范围内时：ASCII 值减去 'a' 的差值加上 'A' 的 ASCII 值
  - 例如：'h' → 'H'：104 - 97 + 65 = 72（'H'）
- 保留非字母字符：逗号、空格、句点等保持不变
- 输入输出对比：
  - 输入：Hello, my name is Lostmonkey.
  - 输出：HELLO, MY NAME IS LOSTMONKEY.

6、2016 年第 10 题（字符串）

以下关于字符串的判定语句中正确的是（）。

A. 字符串是一种特殊的线性表

B. 串的长度必须大于零

C. 字符串不可以用数组来表示

D. 空格字符组成的串就是空串

正确答案： A

解析：

- A 正确：字符串本质是字符的线性序列，满足线性表的所有特性（顺序存储、有限元素集合、元素类型相同）。
- B 错误：空串（""）长度为 0，是合法字符串。
- C 错误：字符串常用字符数组表示（如 char s[]="abc"）。
- D 错误：空格串（如 " "）长度≥1，空串长度=0。

7、2016 年第 26 题（字符串）

阅读程序写结果：

```
#include <iostream>
using namespace std;
int main(){
    int i, length1, length2;
    string s1, s2;
    s1 = "I have a dream.";
    s2 = "I Have A Dream.";
    length1 = s1.size();
    length2 = s2.size();
    for (i = 0; i < length1; i++)
        if (s1[i] >= 'a' && s1[i] <= 'z')
            s1[i] -= 'a' - 'A';
    for (i = 0; i < length2; i++)
```

```
            if (s2[i] >= 'a' && s2[i] <= 'z')
                s2[i] -= 'a' - 'A';
        if (s1 == s2)
            cout << "=" << endl;
        else if (s1 > s2)
            cout << ">" << endl;
        else
            cout << "<" << endl;
        return 0;
    }
```

输出：_____

正确答案： =

解析：

- 将 s1 和 s2 中的**所有小写字母转为大写**('a'-'A'=32,减 32 等价于转大写)。
- 比较转换后的字符串。
- 转换结果：

  s1 转换后："I HAVE A DREAM."

  s2 转换后："I HAVE A DREAM."（原串大写字母已符合要求）
- 比较结果：两字符串完全相同，输出 "="。

8、2017 年第 14 题（字符串子串数量）

若串 S=copyright，其子串的个数是（ ）。

A. 72

B. 45

C. 46

D. 36

正确答案： C

解析：

- 子串定义：包括所有连续字符序列（含空串）。
- 计算公式：长度为 n 的字符串有 **n(n+1)/2+1** 个子串（含空串）。
- 计算过程：

  字符串 "copyright" 长度 n=9。

  非空子串数：9×10/2 = 45。

  含空串总数：45 + 1 = 46。

9、2017 年第 23 题（字符串）

阅读程序写结果：

```
#include<iostream>
using namespace std;
int main()
{
    int t[256];
    string s;
    int i;
    cin >> s;
    for (i = 0; i < 256; i++)
        t[i] = 0;
    for (i = 0; i < s.length(); i++)
        t[s[i]]++;
    for (i = 0; i < s.length(); i++)
        if (t[s[i]] == 1)
        {
            cout << s[i] << endl;
```

```
                return 0;
            }
        cout << "no" << endl;
        return 0;
    }
```

输入：xyzxyw

输出：_____

正确答案： z

解析：

- **统计字符串中各字符出现次数**（t[字符] 存储频次）。
- 按原顺序查找第一个频次为 1 的字符并输出。
- 输入分析：
  - 字符频次：x:2, y:2, z:1, w:1。
  - 按顺序检查：
    - x（位置 0）：频次 2 → 跳过
    - y（位置 1）：频次 2 → 跳过
    - z（位置 2）：频次 1 → 输出 z 并终止程序。
- 关键点：w 虽频次为 1，但顺序在 z 之后，不被检查。

10、2018 年第 6 题（字符串）

如果开始时计算机处于小写输入状态，现在有一只小老鼠反复按照 CapsLock、字母键 A、字母键 S、字母键 D、字母键 F 的顺序循环按键，即 CapsLock、A、S、D、F、CapsLock、A、S、D、F、……，屏幕上输出的第 81 个字符是字母 （ ）

A. A

B. S

C. D

D. a

正确答案： A

解析：周期分析

按键序	0（Caps）	1（A）	2（S）	3（D）	4（F）
循环 1	切大写	A	S	D	F
循环 2	切小写	a	s	d	f
循环 3	切大写	A	S	D	F

商为偶数是大写状态，商为奇数是小写状态。

每 4 个字母为一个循环，第 81 个字符是 81 / 4 = 20，偶数，大写。 81 % 4 = 1，即 A。

11、2018 年第 18 题（字符串）

阅读程序写结果:

```
#include <stdio.h>
char st[100];
int main() {
    scanf("%s", st);
    for (int i = 0; st[i]; ++i) {
        if ('A' <= st[i] && st[i] <= 'Z')
            st[i] += 1;
    }
    printf("%s\n", st);
    return 0;
}
```

输入：QuanGuoLianSai

正确答案： RuanHuoMianTai

解析：

- 遍历字符串，若字符是大写字母（'A'-'Z'），则 ASCII 值 +1。
- 输入处理：

  Q→R（ASCII 81→82）

  G→H（71→72）

  L→M（76→77）

  S→T（83→84）
- 小写字母（u,a,n,o,i,a）不满足条件，保持不变。

输出结果：

原串：QuanGuoLianSai

新串：RuanHuoMianTai

i	0	1	2	3	4	5	...	13
st[i]	Q	u	a	n	G	u		i
st[i]+1	R	u	a	n	H	u		i

12、2022 年第 14 题（字符串）

一个字符串中任意个连续的字符组成的子序列称为该字符串的子串，则字符串 abcab 有（ ）个内容互不相同的子串。

A. 12

B. 13

C. 14

D. 15

正确答案： B

解析：

- 枚举所有不同子串：

- 长度 1：a, b, c → 3 个
- 长度 2：ab, bc, ca, cb → 4 个（ab 重复出现）
  - 位置 0-1："ab"
  - 位置 1-2："bc"
  - 位置 2-3："ca"
  - 位置 3-4："ab"（重复）
- 长度 3：abc, bca, cab → 3 个
  - 位置 0-2："abc"
  - 位置 1-3："bca"
  - 位置 2-4："cab"
- 长度 4：abca, bcab → 2 个
  - 位置 0-3："abca"
  - 位置 1-4："bcab"
- 长度 5：abcab → 1 个
- 总计不同子串：3 + 4 + 3 + 2 + 1 = 13 个

验证公式：

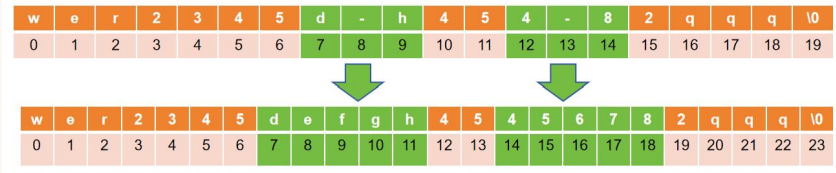
- 总可能子串数：5×6/2=15
- 重复子串："a"（位置 0 和 3）、"b"（位置 1 和 4）、"ab"（位置 0-1 和 3-4）→ 15-2=13。

13. 2007 年第 26 题（字符串展开）

看程序写结果：

```
#include "ctype.h"
#include "stdio.h"
void expand( char s1[], char s2[] )
{
 int i, j, a, b, c;
 j = 0;
```

```
for ( i = 0; (c = s1[i]) != '\0'; i++ )
    if ( c == '-' )
{
    a = s1[i - 1]; b = s1[i + 1];
    if ( isalpha( a ) && isalpha( b ) || isdigit( a ) && isdigit( b ) )
        /*函数 isalpha(a) 用于判断字符 a 是否为字母，
        isdigit(b) 用于判断字符 b 是否为数字，
        如果是,返回 1，否则返回 0 */
    {
        j--;
        do
            s2[j++] = a++;
        while ( tolower( a ) < tolower( s1[i + 1] ) );
    }
    /*函数 tolower(a) 的功能是当字符 a 是大写字母,改为小写,其余情
况不变*/
    else s2[j++] = c;
}else s2[j++] = c;
s2[j] = '\0';
}
int main()
{
 char s1[100], s2[300];
 printf( "input s1:" );
 gets( s1 );
 expand( s1, s2 );
```



```
    printf( "%s\n", s2 );
}
```

输入：wer2345d-h454-82qqq

答案：

wer2345defgh45456782qqq

解析：

- 函数功能：**expand 函数将 s1 中的连字符 - 按规则展开**：
  - 若 - 两侧字符同为字母或数字（如 d-h 或 4-8），则展开为连续序列（defgh 或 45678）。
  - 否则直接复制 -（如 - 在开头/结尾，或两侧类型不同）。
- 关键步骤：
  - j--：回退指针，覆盖前一个字符（因 - 前的字符已写入，需重新开始展开）。
  - do...while 循环：从 a 递增写入字符，直到 a 等于 b（tolower 确保大小写不敏感）。
- 输入处理：
  - wer2345d-h → d-h 展开为 defgh → wer2345defgh
  - 454-8 → 4-8 展开为 45678 → 4545678
  - 82qqq 直接复制
- 最终输出：wer2345defgh45456782qqq

14. 2008 年第 27 题（字符串替换）

完善程序：

（字符串替换）给定一个字符串（S）((S) 仅包含大小写字母），下面的程序将（S）中的每个字母用规定的字母替换，并输出（S）经过替换后的结果。程序的输入是两个字符串，第一个字符串是给定的字符串（S），第二个字符串（S'）由 26 个字母组成，它是（a~ z）的任一排列，大小写不定，（S'）规定了每个字母对应的替换字母：（S'）中的第一个字母是字母（A）和（a）的替换字母，即（S）中的（A）用该字母的大写替换，(S) 中的（a）用该字母的小写替换；（S'）中的第二个字母是字母（B）和（b）的替换字母，

即（S）中的（B）用该字母的大写替换，(S）中的（b）用该字母的小写替换；……以此类推。

```
#include <iostream>
#include <string.h>
char change[26], str[5000];
using namespace std;
void CheckChangeRule()
{
    int i;
    for (i = 0; i < 26; i++)
    {
        if (【①】)
            change[i] -= 'A' - 'a';
    }
}
void ChangeString()
{
    int i;
    for (i = 0; i < strlen(str); i++)
    {
        if (【②】)
            str[i] = change[str[i] - 'A'] - 'a' + 'A';
        else
            【③】;
    }
}
```

```
int main()
{
    int i;
    cin >> str;
    cin >> change;
    CheckChangeRule();
    【④】;
    cout << str << endl;
    return 0;
}
```

答案：

① change[i] >= 'A' && change[i] <= 'Z'

② str[i] >= 'A' && str[i] <= 'Z'

③ str[i] = change[str[i] - 'a'];

④ ChangeString();

解析：

- ①：将 change 中的大写字母转为小写（统一替换规则）。
- ②：检测 str 中的大写字母，计算其在字母表中的位置（str[i]-'A'），从 change 取出替换字母后转为大写。
- ③：处理小写字母，直接取 change 中对应位置的小写字母。
- ④：调用函数执行替换。
- **示例：**

输入：

HelloWorld

qwertyuiopasdfghjklzxcvbnm

输出：

ItssgVgksr

	0	1	2	3	4	5	6	7	8	9
S	H	e	l	l	o	W	o	r	l	d

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
S’	q	w	e	r	t	y	u	i	o	p	a	s	d	f	g	h	j	k	l	z	x	c	v	b	n	m

完整版程序

```
#include <iostream>
#include <string.h>
char change[26], str[5000];
using namespace std;
void CheckChangeRule()
{
    int i;
    for (i = 0; i < 26; i++)
    {
        if (change[i] >= 'A' && change[i] <= 'Z') // ①
            change[i] -= 'A' - 'a';
    }
}
void ChangeString()
{
    int i;
    for (i = 0; i < strlen(str); i++)
    {
        if (str[i] >= 'A' && str[i] <= 'Z') // ②
```

```
                str[i] = change[str[i] - 'A'] - 'a' + 'A';
            else
                str[i] = change[str[i] - 'a']; // ③
        }
    }
    int main()
    {
        int i;
        cin >> str;
        cin >> change;
        CheckChangeRule();
        ChangeString(); // ④
        cout << str << endl;
        return 0;
    }
```

15. 2011 年第 24 题（字符转换）

阅读程序写结果：

```
#include<iostream>
#include<string>
using namespace std;
int main(){
    string map= "22233344455566677778889999";
    string tel;
    int i;
    cin>>tel;
```

```
        for(i=0;i<tel.length();i++)
            if((tel[i]>='0') && (tel[i]<='9') )
                cout<<tel[i];
            else if( (tel[i]>='A') && (tel[i]<='Z'))
                cout<<map[tel[i]-'A'];
        cout<<endl;
        return 0;
    }
```

输入：CCF-NOIP-2011

答案：

22366472011

解析：

- **映射规则：map 字符串按字母顺序存储替换数字**（A→2, B→2, C→2, D→3, ..., Z→9）。
- 输入处理：
  - CCF → C=2, C=2, F=3 → 223
  - - 被忽略（非数字/字母）
  - NOIP → N=6, O=6, I=4, P=7 → 6647
  - 2011 直接保留
- 最终输出：22366472011

tel	C	C	F	-	N	O	I	P	-	2	0	1	1
tel[i]-‘A’	2	2	5		13	14	8	15					
map[tel[i]-‘A’]	2	2	3		6	6	4	7					

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
map	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6	7

16. 2012 年第 26 题（循环位移）

阅读程序写结果：

```
#include <iostream>
#include <string>
using namespace std;
int n, i, j, ans;
string s;
char get(int i)
{
    if (i < n) return s[i];
    else return s[i - n];
}
int main()
{
    cin >> s;
    n = s.size();
    ans = 0;
    for (i = 1; i <= n - 1; i++)
    {
        for (j = 0; j <= n - 1; j++)
            if (get(i + j) < get(ans + j))
            {
                ans = i;
                break;
            }
```

```
            else if (get(i + j) > get(ans + j)) break;
        }
        for (j = 0; j <= n - 1; j++) cout << get(ans + j);
        cout << endl;
        return 0;
    }
```

输入：CBBADADA

答案：

ACBBADAD

解析：

- 函数功能：get(i) **实现循环索引（字符串视为环形）**。
- 核心算法：
  - 遍历所有循环位移起点 i（从 1 到 n-1）。
  - 逐字符比较当前位移（i）与已知最小位移（ans）：
    - 若更小，更新 ans 并跳出内层循环。
    - 若更大，直接终止内层循环。
- 输入分析：
  - 字符串 CBBADADA 的循环位移有 8 种。
  - 最小字典序位移：ACBBADAD（以第 7 位 A 开头）。
- 输出：字典序最小的循环位移 ACBBADAD。

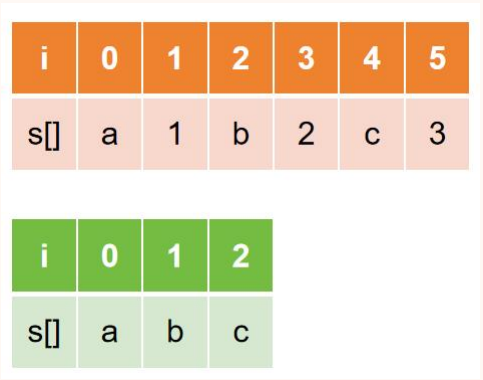
i	0	1	2	3	4	5	6	7
原	C	B	B	A	D	A	D	A
现	A	C	B	B	A	D	A	D

17. 2014 年第 27 题（删除数字字符）

完善程序：

(数字删除) 下面程序的功能是将字符串中的**数字字符删除后输出**。请填空。(每空 3 分，共 12 分)

```
#include <iostream>
using namespace std;
int delnum( char *s )
{
 int i, j;
 j = 0;
 for ( i = 0; s[i] != '\0'; i++ )
      if ( s[i] < '0' 【①】 s[i] > '9' )
      {
           s[j] = s[i];
           【②】;
      }
 Return【 ③】);
}
const int SIZE = 30;
int main()
{
 char     s[SIZE];
 int len, i;
 cin.getline( s, sizeof(s) );
 len = delnum( s );
 for ( i = 0; i < len; i++ )
```



i	0	1	2	3	4	5
s[]	a	1	b	2	c	3

i	0	1	2
s[]	a	b	c

```
        cout << 【④】;
    cout << endl;
    return(0);
}
```

答案:

① s[i] < '0' || s[i] > '9'

② j++

③ j

④ s[i]

解析:

①：筛选非数字字符（ASCII 范围外）。

②：有效字符写入新位置后，指针 j 后移。

③：返回新字符串长度（j 为最终索引，即长度）。

④：按新长度输出过滤后的字符串。

示例：输入 a1b2c3 → 输出 abc。

完整版程序

```
#include <iostream>
using namespace std;
int delnum( char *s )
{
 int i, j;
 j = 0;
 for ( i = 0; s[i] != '\0'; i++ )
     if ( s[i] < '0' || s[i] > '9' )  // ①
     {
```

i	0	1	2	3	4	5
s[]	a	1	b	2	c	3

i	0	1	2
s[]	a	b	c

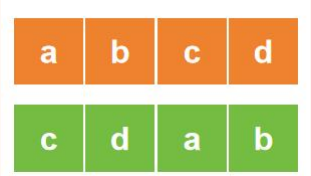
```
            s[j] = s[i];
            j++;   // ②
        }
    return(j);   // ③
}
const int SIZE = 30;
int main()
{
    char      s[SIZE];
    int len, i;
    cin.getline( s, sizeof(s) );
    len = delnum( s );
    for ( i = 0; i < len; i++ )
        cout << s[i];   // ④
    cout << endl;
    return(0);
}
```

18. 2023 年第 17 题（字符串匹配）

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int f(string x, string y) {
    int m = x.size();
```

```
    int n = y.size();
    vector<vector<int>> v(m + 1, vector<int>(n + 1, 0));
    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (x[i - 1] == y[j - 1]) {
                v[i][j] = v[i - 1][j - 1] + 1;
            } else {
                v[i][j] = max(v[i - 1][j], v[i][j - 1]);
            }
        }
    }
    return v[m][n];
}
bool g(string x, string y) {
    if (x.size() != y.size()) {
        return false;
    }
    return f(x + x, y) == y.size();
}
int main() {
    string x, y;
    cin >> x >> y;
    cout << g(x, y) << endl;
    return 0;
}
```

	j=1	j=2	j=3	j=4
i=1	0	0	1	1
i=2	0	0	1	2
i=3	1	1	1	2
i=4	1	2	2	2
i=5	1	2	3	3
i=6	1	2	3	4
i=7	1	2	3	4
i=8	1	2	3	**4**



判断题

1、f 函数的返回值小于等于 (min(n, m))。 ()

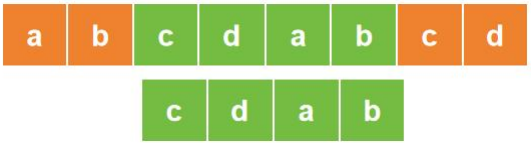
答案： A. 正确

解析：f 函数返回的是两个字符串的最长公共子序列的长度，最长不会超过较短字符串的长度。

2、f 函数的返回值等于两个输入字符串的最长公共子串的长度。 ()

答案：B. 错误

解析：f 函数计算的是最长公共子序列的长度，而不是最长公共子串的长度。



3、当输入两个完全相同的字符串时，g 函数的返回值总是 true。 ()

答案：A. 正确

解析：如果两个字符串相同，将其中一个与自己拼接，检查其是否包含另一个，结果总是 true。

单选题

4. 将第 19 行中的 v[m][n] 替换为 v[n][m]，那么该程序（）。

答案：D. 可能非正常退出

解析：如果 m != n，那么访问 v[n][m] 会越界，导致程序可能非正常退出。

	j=1	j=2	j=3	j=4
i=1	0	0	1	1
i=2	0	0	1	2
i=3	1	1	1	2
i=4	1	2	2	2
i=5	1	2	3	3
i=6	1	2	3	4
i=7	1	2	3	4
i=8	1	2	3	**4**

5. 当输入为 csp-j p-jcs 时，输出为（）。

答案：B. 1

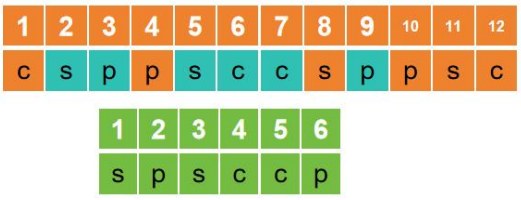
解析：字符串 "csp-j" 与 "p-jcs" 互为循环移位，因此返回 1。

6. 当输入为 csppsc spsccp 时，输出为（）。

答案：D. 1

解析：字符串 "csppsc" 与 "spsccp" 互为循环移位，因此返回 1。

	j=1	j=2	j=3	j=4	j=5	j=6
i=1	0	0	0	1	1	1
i=2	1	1	1	1	1	1
i=3	1	2	2	2	2	2
i=4	1	2	2	2	2	3
i=5	1	2	3	3	3	3
i=6	1	2	3	4	4	4
i=7	1	2	3	4	5	5
i=8	1	2	3	4	5	5
i=9	1	2	3	4	5	6
i=10	1	2	3	4	5	6
i=11	1	2	3	4	5	6
i=12	1	2	3	4	5	6



程序功能详解

这个程序用于判断字符串 y 是否是字符串 x 的循环旋转（cyclic rotation）。具体来说，如果 y 可以通过将 x 的前若干字符移到末尾得到，则返回 1（真），否则返回 0（假）。

核心函数解析：

- **f(x, y) 函数：**计算两个字符串的最长公共子序列（LCS）长度
  - 使用动态规划，创建二维数组 v
  - v[i][j] 表示 x[0..i-1] 和 y[0..j-1] 的 LCS 长度
  - 状态转移：

- if (x[i-1] == y[j-1])
- v[i][j] = v[i-1][j-1] + 1; // 字符匹配，长度+1
- else
- v[i][j] = max(v[i-1][j], v[i][j-1]); // 取左侧或上方的最大值

- **g(x, y) 函数：判断循环旋转**
  - 关键操作：f(x + x, y)
  - 原理：将 x 复制一份拼接（如 "abc" → "abcabc"），如果 y 是 x 的循环旋转，那么 y 必定是 x+x 的子序列
  - 验证条件：f(x+x, y) == y.size()

执行流程：

- f 函数：动态规划计算最长公共子序列（LCS），非子串（子串需连续）。
- g 函数：检测 y 是否为 x 的循环移位（通过判断 x+x 是否包含 y）。
- 判断题：
  - LCS 长度 ≤ min(m,n)（正确）。
  - LCS ≠ 最长公共子串（错误）。
  - 相同字符串互为循环移位（正确）。
- 单选题：
  - 4. m≠n 时访问 v[n][m] 会越界（可能崩溃）。
  - 5. csp-j 与 p-jcs 是循环移位 → 输出 1（true）。
  - 6. csppsc 与 spsccp 是循环移位 → 输出 1。