1. Change letter case (easy)

Write a program that converts a letter to lowercase if the letter is uppercase, and converts a letter to uppercase if the letter is lowercase. There will be exactly 5 letters in the input.

EXAMPLE INPUT

```
Ab3Cd
```

EXAMPLE OUTPUT

```
aB3cD
```

主程序

```
#include "source"
```

答案程序

```
#include <iostream>
using namespace std;

bool isUppercase(char c) {
    return c >= 'A' and c <= 'Z';
}

bool isLowercase(char c) {
    return c >= 'a' and c <= 'z';
}

char toUppercase(char c) {
    if (isLowercase(c)) {
        return c - ('a' - 'A');
    }
    return c;
}

char toLowercase(char c) {
    if (isUppercase(c)) {
        return c + ('a' - 'A');
    }
    return c;
}


int main() {
    for (int i = 0; i < 5; ++ i) {
        char c;
        cin >> c;
        if (isUppercase(c)) {
            c = toLowercase(c);
        }
        else if (isLowercase(c)) {

            c = toUppercase(c);
        }
        cout << c;
    }
}
```

文本编辑框

保存和测试对错 答案正确

2. Valid triangles (easy)

Write a program that determines if the lengths of the three sides of a triangle are valid.
The lengths of the three sides are valid if none of them is greater than the sum of the other two sides.
The input contains exactly 5 triangles (15 sides).

EXAMPLE INPUT

```
5 2.1 6
4 2.9 9
423 543 234
34 67 987
6542 100 6540
```

EXAMPLE OUTPUT

```
valid
invalid
valid
invalid
valid
```

主程序

```
#include "source"
```

答案程序

```
#include <iostream>
using namespace std;

void testTriangle() {
    double side1, side2, side3;
    cin >> side1 >> side2 >> side3;
    bool valid1 = (side1 + side2 > side3);
    bool valid2 = (side1 + side3 > side2);
    bool valid3 = (side3 + side2 > side1);
    bool valid = valid1 and valid2 and valid3;
    cout << (valid ? "valid" : "invalid") << endl;
}

int main() {
    for (int i = 0; i < 5; ++ i) {
        testTriangle();
    }

}
```

文本编辑框

保存和测试对错 答案正确

3. Display pattern (easy)

Write a program that prints the pattern with a given height as shown in the example.

EXAMPLE INPUT

```
7
```

EXAMPLE OUTPUT

```
+           +
++         ++
+ +       + +
```

```
+  +     +  +
+   +   +   +
+    + +    +
+     +     +
```

主程序

```
#include <iostream>
using namespace std;

#include "source"


int main() {
    int height;
    cin >> height;
    printPattern(height);
}
```

答案程序

```
void printSpaces(int length) {
    for (int i = 0; i < length; ++ i) {
        cout << ' ';
    }
}

void printFirstLine(int height) {
    cout << '+';
    printSpaces(2 * height - 3);
    cout << "+\n";
}

void printLastLine(int height) {
    cout << '+';
    printSpaces(height - 2);
    cout << '+';
    printSpaces(height - 2);
    cout << "+\n";
}

void printPattern(int height) {
    printFirstLine(height);
    for (int i = 0; i < height - 2; ++ i) {
        cout << '+';
        printSpaces(i);
        cout << '+';
        printSpaces(2 * height - 2 * i - 5);
        cout << '+';
        printSpaces(i);
        cout << "+\n";
    }
    printLastLine(height);
}
```

文本编辑框

保存和测试对错 答案正确

4. Sum of digits (easy)

Write a function that returns the sum of the largest and the smallest digits in a positive integer number.

EXAMPLE INPUT

```
1234
5427
29308
1940543
963353
```

EXAMPLE OUTPUT

```
5
9
9
9
12
```

主程序

```
#include <iostream>
using namespace std;

#include "source"

int main() {
    for (int i = 0; i < 5; ++ i) {
        int number;
        cin >> number;
        cout << sumTwoDigits(number) << endl;

    }
}
```

答案程序

```
int sumTwoDigits(int number) {
    int largest = 0;
    int smallest = 10;
    while (number > 0) {
        int digit = number % 10;
        number /= 10;
        largest = (largest > digit ? largest : digit);
        smallest = (smallest < digit ? smallest : digit);
    }
    return largest + smallest;
}
```

文本编辑框

保存和测试对错 答案正确

5. Connect integers (hard)

Write a function that connects two positive integers.

EXAMPLE INPUT

```
123 456
1230 456
123 4560
```

EXAMPLE OUTPUT

```
123456
1230456
1234560
```

主程序

```
#include <iostream>
using namespace std;
```

```
using namespace std;

#include "source"

int main() {
    for (int i = 0; i < 3; ++ i) {
        int number1;
        int number2;
        cin >> number1 >> number2;
        cout << connect(number1, number2) << endl;
    }
}
```

答案程序

```
int length(int number) {
    int len = 0;

    while (number > 0) {
        number /= 10;
        ++ len;
    }
    return len;
}

int connect(int number1, int number2) {
    int len = length(number2);
    for (int i = 0; i < len; ++ i) {
        number1 *= 10;
    }
    return number1 + number2;
}
```

文本编辑框

保存和测试对错 答案正确

6. Sum of three primes (hard)

Write a function that prints the three primes sum up equal to given number. Print the smallest possible prime first and the largest possible prime last.

EXAMPLE INPUT

```
6
7
8
9
10
100
1000
10000
100000
1000000
```

EXAMPLE OUTPUT

```
6 = 2 + 2 + 2
7 = 2 + 2 + 3
8 = 2 + 3 + 3
9 = 2 + 2 + 5
10 = 2 + 3 + 5
100 = 2 + 19 + 79
1000 = 2 + 7 + 991
10000 = 2 + 31 + 9967
100000 = 2 + 7 + 99991
1000000 = 2 + 19 + 999979
```

主程序

主程序

```
#include <iostream>
using namespace std;

#include "source"

int main() {
    for (int i = 0; i < 10; ++ i) {
        int number;
        cin >> number;
        printEquation(number);
    }
}
```

答案程序

```
int isPrime(int number) {
    for (int i = 2; i <= number / 2; ++ i) {
        if (number % i == 0) return false;
    }
    return true;
}

void printEquation(int sum) {
    for (int i = 2; i < sum / 2; ++ i) {
        if (! isPrime(i)) continue;
        for (int j = sum - i * 2; j >= i; -- j) {
            int k = sum - (i + j);
            if (k < i || k > j) break;
            if (! isPrime(j)) continue;
            if (! isPrime(k)) continue;
            cout << sum << " = " << i << " + " << k << " + " << j << endl;
            return;
        }
    }
}
```

文本编辑框

保存和测试对错 答案正确

7. Connect integers without overlapping (hard++)

Connect two positive integers, with overlapping digits removed.

EXAMPLE INPUT

```
123456 456789
120 120456
123450 450
123450 6789
123450 2345
```

EXAMPLE OUTPUT

```
123456789
120456
123450
1234506789
1234502345
```

主程序

```
#include <iostream>
using namespace std;

#include "source"

```

```
int main() {
    for (int i = 0; i < 5; ++ i) {
        int number1;
        int number2;
        cin >> number1 >> number2;
        int connected = connectWithoutOverlapping(number1, number2);
        cout << connected << endl;
    }
}
```

答案程序

```
int length(int number) {
    int len = 0;
    while (number > 0) {
        number /= 10;
        ++ len;
    }
    return len;
}

int getFirstDigits(int number, int count) {
    int len = length(number);
    int removeDigitCount = len - count;
    for (int i = 0; i < removeDigitCount; ++ i) {
        number /= 10;
    }
    return number;
}

int getLastDigits(int number, int count) {
    int base = 1;
    int result = 0;
    for (int i = 0; i < count; ++ i) {
        result += base * (number % 10);
        number /= 10;
        base *= 10;
    }
    return result;
}

int numberOfOverlapping(int number1, int number2) {
    int len1 = length(number1);
    int len2 = length(number2);
    int maxLen = (len1 > len2 ? len1 : len2);
    for (int i = maxLen; i >= 0; -- i) {
        int lastDigits = getLastDigits(number1, i);
        int firstDigits = getFirstDigits(number2, i);
        if (lastDigits == firstDigits) return i;
    }
}

int connect(int number1, int number2) {
    int len = length(number2);
    for (int i = 0; i < len; ++ i) {
        number1 *= 10;
    }
    return number1 + number2;
}

int connectWithoutOverlapping(int number1, int number2) {
    int overlapping = numberOfOverlapping(number1, number2);
    int len1 = length(number1);
    number1 = getFirstDigits(number1, len1 - overlapping);
```

```
53         return connect(number1, number2);
54     }
```

文本编辑框

保存和测试对错 答案正确

保存于 14:14.11