

SI@UCF Computer Science Camp
Intro to Competitive Programming in C++
Quiz #1 Solutions
Date: 6/9/2026

1) (10 pts) Consider the following problem: The input consists of two space-separated integers: n the number of servings of a recipe you will make, c , the number of cups of chicken broth needed for the recipe. You can purchase any whole number of gallons of broth. (Note: There are 16 cups in a gallon.) Complete the program below to read the input (without any prompting) and output an integer representing the least number of gallons of broth you can buy to fulfill your recipe needs. (Note: for full credit no floating point numbers are allowed anywhere.)

```
using namespace std;
#include <bits/stdc++.h>

int main() {

    int n, c;                // 1 pt
    cin >> n >> c;          // 2 pts
    cout << (n*c+15)/16 << endl; // 7 pts

    return 0;
}
```

2) (5 pts) Show the output of the following C++ program.

```
using namespace std;
#include <bits/stdc++.h>

int main() {
    vector<int> arr = {9, 16, 13, 2, 8, 17, 14};
    sort(arr.begin()+1, arr.end()-1);
    for (int i=0; i<arr.size(); i++)
        cout << arr[i] << " ";
    cout << endl;
    return 0;
}
```

9 2 8 13 16 17 14

Grading: 5 pts correct list, 3 pts if fully sorted, 0 points otherwise

3) (15 pts) Below is a full prompt from Kattis for a problem (Cold-puter Science). Complete your solution below. (To save space, sample input/output isn't provided, but the input specification is fairly clear.)

We're not going to sugar-coat it: Chicago's winters can be rough. The temperatures sometimes dip to uncomfortable levels and, after last year's "polar vortex", the University of Chicago Weather Service wants to find out exactly how bad the winter was. More specifically, they are interested in knowing the total number of days in which the temperature was below zero degrees Celsius.

Input

The input is composed of two lines. The first line contains a single positive integer, n ($1 \leq n \leq 100$), that specifies the number of temperatures collected by the University of Chicago Weather Service. The second line contains n temperatures, each separated by a single space. Each temperature is represented by an integer t ($-1000000 \leq t \leq 1000000$).

Output

You must print a single integer: the number of temperatures strictly less than zero.

```
using namespace std;
#include <bits/stdc++.h>

int main() {

    int n, tmp, res = 0;                // 1 pt vars, 1 pt set 0
    cin >> n;                          // 2 pts
    for (int i=0; i<n; i++) {           // 2 pts
        cin >> tmp;                    // 2 pts
        if (tmp < 0) res++;             // 5 pts
    }
    cout << res << endl;               // 2 pts

    return 0;
}
```

4) (5 pts) Jasmine enters 50,000 for n in the following code segment. Why does it not print out the answer she intends?

```
cin >> n
cout << n*(n+1)/2 << endl;
```

The intermediate calculation of 50000×50001 is just above 2.5 billion, which overflows integer. It doesn't matter that the final answer of 1,250,025,000 does fit in the int data type.

Grading: Up to grader discretion.

5) (25 pts) We can store a polynomial in vector as follows: Store the coefficient of x^i in index i . For example, the polynomial function $f(x) = 4x^3 - 6x^2 + 2x - 7$ would be stored as $\{-7, 2, -6, 4\}$. Now, imagine evaluating this polynomial for the integers $0, 1, 2, 3, \dots, n-1$. For example if $n = 5$, the values computed would be $-7, -7, 5, 53, 161$. Complete the method below so that it takes in a polynomial as stored above and an integer n , and returns a vector of size n storing $f(0), f(1), \dots, f(n-1)$ in that order. (Thus the function call `compute({-7,2,-5,4}, 5)` would return $[-7, -7, 5, 53, 161]$.)

Assume no overflow errors will occur and ONLY use int for intermediate calculations.

```
vector<int> compute(vector<int> f, int n) {

    vector<int> res;                // 3 pts (can do res(n))
    for (int x=0; x<n; x++) {       // 3 pts

        int pX = 1, sum=0;          // 3 pts
        for (int i=0; i<f.size(); i++) { // 3 pts
            sum += (pX*f[i]);        // 6 pts
            pX *= x;                 // 3 pts
        }

        res.push_back(sum);         // 3 pts
    }

    return res;                     // 1 pt
}
```

6) (10 pts) Many students had trouble solving the problem Baloni from the homework list. (A quick refresher: In this problem you had to determine the minimum number of arrows to shoot down a set of balloons. The balloons were in order from left to right and you could choose any height to start an arrow. Whenever an arrow hit a balloon, it popped it but then had its height lowered by 1 unit.) Summarize HOW we stored the necessary data for this problem and how it aided in an efficient solution to the problem.

The heights of the balloons were no more than a million, so for each height level, we could store the number of arrows currently flying at that height. This way, when we approached a balloon, we could easily see if there was an arrow already flying at the proper height (immediately) that was available to shoot down the balloon. If not, we would need to shoot a new arrow to get this balloon.

Grading: Grader discretion

7) (25 pts) What follows is the prompt for a Kattis problem that uses strings. Complete the solution for it on the space provided.

Drunk Vigenere

One of the most well-known and classic methods of encryption is the Vigenère cipher. Given a message and key of equal length, all you need to do to encrypt the message is shift each character forward by a certain number of characters in the alphabet.

For example, if your string is CALGARY and key is ALBERTA, the number of letters to shift the i^{th} letter in the message is given by the position in the alphabet of the i^{th} letter in the key, using 0-based indexing. So, the first character in our message C is shifted 0 letters forward since the first character in our key is A. Likewise, A is shifted 11 characters since L is the 12th letter in the alphabet. Repeating, for each character, we'll find the encrypted message ends up being CLMKRKY.

Of course since this is such a well-known encryption method, it isn't very safe so your friend Yraglac has come up with a brilliant idea: for each even-indexed character (0-based) in your message, you do the usual encryption as described above, but for the odd-indexed characters, instead of shifting forwards by the key, you actually shift backwards. Thus, our example above would actually be encrypted as CPMCRY Y. That'll throw off any pesky hackers trying to read your secrets!

Of course Yraglac has left the implementation up to you. Given an encrypted message and the key, can you write a program to decrypt it? Don't forget that everything is using 0-based indexing.

Input

Input consists of two lines. The first contains the encrypted message C and the second contains the key K . C and K are always of equal length between 1 and 1000 characters long and consists only of uppercase alphabetic letters.

Output

Output the decrypted message.

Sample Input #1

CPMCRY Y
ALBERTA

Sample Output #1

CALGARY

Sample Input #1

TDLIVA
OILERS

Sample Output #1

FLAMES

```

using namespace std;

#include <bits/stdc++.h>

int main() {

    // Get input.
    string c, k;
    cin >> c >> k;

    int sign = -1; // 1 pt

    for (int i=0; i<c.size(); i++) { // 3 pts
        int num = ((c[i]-'A') + sign*(k[i]-'A') + 26)%26; //16
        cout << (char)(num+'A'); // 3 pts
        sign = -sign; // 2 pts
    }

    cout << endl; // 0 pts

    return 0;
}

```

Grading: 16 point breakdown as follows:

4 pts: (c[i] - 'A')

6 pts: sign*(k[i] - 'A') or equivalent

3 pts: + 26

3 pts: %26

8) (5 pts) The Federation Internationale de Football Association's famous World Cup is starting on Thursday in North America. By what acronym is the organization better known?

FIFA (Give to All)