Back to High School Physics

- **Justification:**
  - It's a direct problem, just asking about the displacement in twice of the giving time and velocity.
- **Input:**
  - **several** test cases, So you need to repeat your code for every test case until the end of the input file. Given '***v***' -> velocity and '***t***' -> time.
- **Output:**
  - Only one number, the displacement in twice of '***t***'.
- **Code:**

```
#include <iostream>
using namespace std;
int main(){
     int v, t, d;
     while(cin >> v >> t){
// returns false by the end of the file.
          d = 2 * t * v;
          cout << d << endl; // print result of every case
     }
}
```

Odd Sum

- **Justification:**
  - Given a range [a, b], you are to find the summation of all the odd integers in this range. For example, the summation of all the odd integers in the range [3, 9] is 3 + 5 + 7 + 9 = 24.
- **Input:**
  - **Fixed** number of test case T **(1** <= T <=**100),** followed by ***T * 2*** numbers. every 2 numbers represents the **start** and the **end** the case.
- **Output:**
  - Only one number, the sum of odd numbers in the range.
- **Code:**

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

int main(){
     int t;
     cin >> t;
     for(int tc = 1, a, b; tc <= t; tc++){
          cin >> a >> b;
          int sum = 0;
          for(int i = a; i <= b; i++){
               if(i % 2 == 1)
                    sum += i;
          }
          cout << "Case " << tc << ": " << sum << endl;
     }
}
```

The 3n + 1 problem

- **Justification:**
  - The problem is clearly defined, given a rang, for every number, find the length of it's cycle.
  - print the maximum length.
  - a lot of tricks here, take care, The integers ***i and j must appear in the output in the same order in which they appeared in the input*** and should be followed by the maximum cycle length (on the same line).
- **Input:**
  - **series** of pairs of integers *i* and *j*, one pair of integers per line. All integers will be less than 1,000,000 and greater than 0.
- **Output:**
  - For each pair of input integers *i* and *j* you should output *i*, *j*, and the maximum cycle length for integers between and including *i* and *j*.
- **Code:**

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

int main ( ) {
     int a, b;
     while (cin >> a >> b) {
          int max = -1;
          cout << a << " " << b << " "; // trick
          if (a > b) { // swapping 2 numbers
               int tmp = a;
               a = b;
               b = tmp;
          }
          for (int j = a; j <= b; j++) {
               int i = j, cycleLength = 1;
               while (i != 1) {
                    if (i % 2 == 0) i /= 2;
                    else if (i % 2 != 0) i = 3 * i + 1;
                    cycleLength++;
               }
               if (cycleLength > max) max = cycleLength;
          }
          cout << max << endl;
     }
     return 0;
}
```

Summing Digits

- **Justification:**
  - You need to repeat the process ***f(n), f(f(n)), f(f(f(n))).*** Stop ***f(n) is a single digit.***
- **Input:**
  - **several** test cases, each line of input contains a single positive integer n at most 2,000,000,000. **Input is terminated by n = 0 which should not be processed.**
- **Output:**
  - Single line with a single digit ***f(n)***

- **Code:**

```
#include <iostream>
using namespace std;
int main ()
{
   int n;
     while(cin >> n && n){
          while(n >= 10){
               int tmp = n, sum = 0;
               while(tmp){
                    sum += tmp % 10;
                    tmp /= 10;
               }
               n = sum;
          }
          cout << n << endl;
     }
   return 0;
}
```

Behold my quadrangle

- **Justification:**
  - We have the length of four sides. You have to determine if they can form a square. If not, determine if they can form a rectangle. If not, determine if they can form a quadrangle, assume the lengths are sorted in increasing order:
    - Square: L1 == L2 && L2 == L3 && L3 == L4
    - Rectangle: L1 == L2 && L3 == L4
    - Quadrangle: L1 + L2 + L3 > L4
- **Input:**
  - **T** number of test cases, following a line with four positive integer numbers, between 0 and 2^30, the lengths.
- **Output:**
  - One of four strings: **‘square’, ‘rectangle’, ‘quadrangle’** or **‘banana’**.

- **Code:**

```cpp
#include <iostream>
using namespace std;
int main () {
	int cases;
	int s1, s2, s3, s4;
	cin >> cases;

	while (cases--) {
		cin >> s1 >> s2 >> s3 >> s4;

		// sort the numbers first.

			if (s1 > s2) {
				int t = s1; s1 = s2; s2 = t;
			}
			if (s1 > s3) {
				int t = s1; s1 = s3; s3 = t;
			}
			if (s1 > s4) {
				int t = s1; s1 = s4; s4 = t;
			}
			if (s2 > s3) {
				int t = s2; s2 = s3; s3 = t;
			}
			if (s2 > s4) {
				int t = s2; s2 = s4; s4 = t;
			}
			if (s3 > s4) {
				int t = s3; s3 = s4; s4 = t;
			}

		if (s1 == s2 && s2 == s3 && s3 == s4) {
			cout << "square" << endl;
		} else if (s1 == s2 && s3 == s4) {
			cout << "rectangle" << endl;
		} else if (s1 + s2 + s3 > s4) {
			cout << "quadrangle" << endl;
		} else {
			cout << "banana" << endl;
		}
	}

	return 0;
}
```

Automatic Answer

- **Justification:**
  - Just apply the arithmetic operation on the given number, we are interested in the digit in the tens column, remember?. So, you will print only a single positive digit.
- **Input:**
  - **Fixed number of test cases *T*,** each of the following ***T*** contains a single number.
- **Output:**
  - The digit in the tens column.
- **Code:**

```
#include <iostream>
using namespace std;
int main ( ) {
     int n, t;
     cin >> t;
     for (int i = 0; i < t; i++) {
          cin >> n;
          n = (((n * 567 / 9) + 7492) * 235 / 47) - 498;
          n = (n / 10) % 10;
          if (n < 0) n *= -1;
          cout << n << endl;
     }
     return 0;
}
```

Jumping Mario

- **Justification:**
  - Given a sequence of numbers, just count up and down moves.
- **Input:**
  - **Fixed** number of test case ***T*** **,** Each case starts with an integer N (0 < N < 50) that determines the number of walls. The next line gives the height of the N walls from left to right. Each height is a positive integer not exceeding 10.
- **Output:**
  - Output the case number followed by 2 integers, total up jumps and total down jumps, respectively.

- **Code:**

```cpp
#include <iostream>
using namespace std;
int main ( ) {
	int n, t;
	cin >> t;
	for (int tc = 1; tc <= t; tc++) {
		cin >> n;
		int prv, cur, uc = 0, dc = 0;
		for (int i = 0; i < n; i++) {
			cin >> cur;
			if (i == 0) {
				prv = cur;
				continue;
			}
			if (prv < cur) uc++;
			if (prv > cur) dc++;
			prv = cur;
		}
	cout << "Case " << tc << ": " << uc << " " << dc << endl;
	}
	return 0;
}
```

Omar

- **Justification:**
  - Given 2 number, print the sum.
- **Input:**
  - **Fixed number of test cases *T*,** each of the following ***T*** contains 2 numbers.
- **Output:**
  - The Sum.
- **Code:**

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

int main ( ) {
	int u, v, t;
	cin >> t;
	while (t--) {
		cin >> u >> v;
		cout << u + v << endl;
	}
}
```

Kids Love Candies

- **Justification:**
  - a kid must eat ***K*** candies of same kind to be happy, calculate the number of happy kids.
- **Input:**
  - **Fixed number of test cases *T,*** each of the following ***T*** cases starts with ***N*** the number of candy kinds, then a line containing N numbers, the amount of each kind.
- **Output:**
  - The number of happy kids.
- **Code:**

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

int main ( ) {
     int t, n, k, m;
     cin >> t;
     while (t--) {
          cin >> n >> k;
          int ans = 0;
          while (n--) {
               cin >> m;
               ans += m / k;
          }
          cout << ans << endl;
     }
}
```
