SOLUTIONS FOR 27TH OF JULY

ABSURD TRIANGLE 5

```
for(int i = 1; i <= n-2; i++) {
  cout << string(n-i-1, ' ') << '/' << string(2*i-2, ' ') << '\\' << string(n-i-1, ' ') << '\n';
}
cout << '/' << string(2*n-4, '_') << '\\' << '\n';
```

In this kind of problems, it may be convenient to use the `string` on the Standard Template Library.

BUILDING PLAN

First, let's sort the given input an so that $x_1 < x_2 < \cdots < x_n$ holds.

What should be the final distance between two adjacent buildings? It should be $\gcd(x_2 - x_1, x_3 - x_2, \cdots, x_n - x_{n-1})$, because the final distance should be a divisor of all given distances. Also, it is a bad idea to build buildings in the left of x_1 or the right of x_n (obviously)

Therefore, the answer is $\left(\frac{x_n - x_1}{\gcd(x_2 - x_1, x_3 - x_2, \cdots, x_n - x_{n-1})} + 1\right) - n$. The former is the number of final buildings, and the latter is the number of already built buildings.

THE INPUT IS *NOT* GUARANTEED TO BE SORTED

In the examples, the input is sorted. That *doesn't mean that the whole input is sorted!* If the statement said nothing, you should assume that the input can be in any order. This is a trick that some authors like to use.

CALCULATOR WITH TWO BUTTONS

The key point of this problem is that it is always *not optimal* to press buttons like "--" and "+-+", because they are both equal to "" (pressing no buttons) and "-", respectively. (You can check it by doing some examples)

Therefore, the optimal solution always look like this:

(Press button "-" 0 or 1 times) -> (Press button "+" 0 or more times) -> (Press button "-" 0 or 1 times)

So the algorithm goes like this: First, consider all 4 cases about the button "-": whether you press "-" at the beginning (change x to $-x$) or not, and whether you press "-" at the end (change y to $-y$) or not. After changing the signs, if $x' \leq y'$ holds, we can press "+" $y' - x'$ times and achieve the goal.

DELIVERING

Let x as the number of 4kg package to use, and y as the number of 7kg package to use. We need to minimize the value of $x + y$ subject to $4x + 7y = N$. We will find it by some observations.

- It is best to use the 7kg package as much as possible. (quite obvious?) In other words, y should be big as possible.
- If y is fixed, we can calculate x by the formula $x = \frac{N-7y}{4}$.
- However x should be an integer, so $N - 7y \equiv 0 \pmod 4$.
- By the properties of modulo operation, $N - 7y \equiv N + y \equiv 0 \pmod 4 \implies y \equiv -N \pmod 4$.
- Therefore, we can iterate y from $\left\lfloor \frac{N}{7} \right\rfloor$ to 0, and check there exists a valid solution (x, y). Because the necessary condition of the existence of solution is just "$y \equiv -N \pmod 4$", it is guaranteed that we just need to consider at most 4 candidates of y.

EVOLVING CREATURES

No one managed to solve this problem, so we won't reveal the solution.

FORWARD TELEPORTERS

Let the answer for the i-th city as $f(i)$. We can find some recurrence relation like this:

- Base case: $f(n) = 0$.
- $1 \le i \le n - 1$: $f(i) = f(p_i) + 1$.

If you calculate these values by implementing a recursive function or naive iteration, it will take $O(n^2)$ time and will probably get TIME-LIMIT. Actually, we can improve this solution by noticing that $p_i > i$ always holds, so we can iterate *backwards*: Starting from $i = n$ to $i = 1$, we can fill out the values of f. Then, it will take $O(n)$ time.

GIANT BLOG

Just do what the statement says.

```
if(1 < p - k) printf("<< ");
for(int i = p-k; i <= p+k; i++) {
  if(1 <= i && i <= n) printf((i == p) ? "(%d) " : "%d ", i);
}
if(p + k < n) printf(">> ");
printf("\n");
```