SOLUTIONS FOR 26TH OF JULY

ABSURD TRIANGLE 4

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

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

BOYS AND GIRLS

If we do not send anybody to the internship, we can form $\min\left(\lfloor\frac{m}{2}\rfloor, n\right)$ teams. This is the upper bound of the answer, and let the value p (for simplicity).

Then, we can send $(m-2\cdot p)+(n-p)$ people to the internship without decreasing the number of teams. (Because we only need $2\cdot p$ girls and p boys to form p teams) So, if $k \le m+n-3\cdot p$, the answer is obviously p.

What if $k > m+n-3\cdot p$? Send $m+n-3\cdot p$ people to the internship, and let's try to solve the same problem with $m' = 2\cdot p,\ n' = p$ and $k' = k-(m+n-3\cdot p)$. How to solve this new problem? First, it is best to send $\left\lfloor\frac{k'}{3}\right\rfloor\times 2$ girls and $\left\lfloor\frac{k'}{3}\right\rfloor$ boys (because they make exactly $\left\lfloor\frac{k'}{3}\right\rfloor$ teams without surplus) After that, we may need to send 0, 1 or 2 people more. If we have to send 1 or 2 people more, we can send that number of girls and then at most one team will be broken. Therefore, the answer for the original problem is $p-\left\lfloor\frac{k'}{3}\right\rfloor$.

Alternatively, we can use a binary search on the answer, because if we can form q teams, it is obvious that we can also form $(q-1)$ teams. To check whether we can form q teams is simple: Check whether $m \ge 2\cdot q,\ n \ge q$ and $k \le m+n-3\cdot q$ all holds.

CONSECUTIVE PAINTING OPERATION

Nobody managed to solve this problem, so we will not reveal the solution.

DELICATE TEAM FORMATION

WLOG, let's sort the given strengths and assume $a_1 \geq a_2 \geq \cdots \geq a_{3n}$. We can prove that the answer is $a_2 + a_4 + \cdots + a_{2n}$.

First, we prove that there is no better solution than this. Suppose we made n teams that are optimal, and we sorted the teams according to their strength, in non-ascending order. Let's consider the i-th strongest team, and let the member that has second largest strength as h. We know that there are at least $(2i-1)$ students that are stronger than h. Why? Since there are $i-1$ teams better than this team, so at least 2 people for each $i-1$ teams are stronger than h. Also, in the i-th team, there is one student stronger than h. Thus, the strength of h is at most a_{2i}, and therefore the answer is at most $a_2 + a_4 + \cdots + a_{2n}$. (I didn't "correctly" consider the case when strengths are equal, but we can use the same logic)

Secondly, we show that there is a valid team formation that achieves this sum: $(1,2,3n), (3,4,3n-1), (5,6,3n-2), \cdots (2k-1,2k,3n-k), \cdots$.

Therefore, the solution is to just sort the strengths in non-ascending order and print $a_2 + a_4 + \cdots + a_{2n}$.

ESTIMATION OF PRODUCTS

It is impossible to find the sign of the product $a \times (a+1) \times \cdots \times b$ by actually multiplying all numbers, because

- The product is super large so it cannot be stored in a signed 64-bit integer type. So if you multiply by some numbers, integer overflow happens and the sign is not correct.
- $b - a \leq 2 \times 10^9$, and just multiplying takes $O(b-a)$ time. There are 10,000 intervals to consider (because 100 test cases * 100 intervals for each test case), so we cannot use this in 1 sec. (refer to the 1e8 rule)

So, we must try to find another way.

First, let's solve the case when $n = 1$. We just focus on the "sign" of the value.

- When $1 \leq a \leq b$, all integers in the interval are positive, so the product is positive.
- When $a \leq 0 \leq b$, the interval contains 0, so the product is zero.
- When $a \leq b \leq -1$, all integers in the interval are negative, so the product is positive if $b - a + 1$ is even, and negative otherwise.

So, we can determine the sign of the product without actually iterating all integers, in constant time. However, the problem asks us to calculate the sign of all products, but this is easy. We can find the sign of products for each interval, and then multiply the "signs" of each interval.

FRONT AND BACK

The length of string u is at most $2n$ - we can just concatenate two strings s and t to make u. Also the length of string u is at least n because of the condition

So, let's try to figure out there exists a valid string u with length l, for each $n \leq l \leq 2 \cdot n$. Such string exists if and only if the last $2n - l$ characters of s and the first $2n - l$ characters of t are the same. We can compare this using a loop or `string` on the Standard Template Library, and print the answer if exists.

GIFTS OF SANTA CLAUS

Although nobody solved it, there were some correct approaches, so we give the solution.

If all a_i is even, clearly we cannot pick a subset that has odd sum, so the answer is 0.

Otherwise, suppose we have an odd element a_k. Consider all 2^{n-1} subsets (including the empty one) of bags excluding bag k.

- If the subset has *even* sum: If we pick a_k additionally, the new sum becomes *odd*. Otherwise, the sum remains even.
- If the subset has *odd* sum: If we don't pick a_k, the sum is *odd*. Otherwise, the sum becomes even.

We can see that for each subset, we can determine whether to pick a_k or not, and there is only one possible choice. Therefore, in this case, the answer is 2^{n-1}.

Note that you cannot use the `pow` function in `math.h`, since it returns a `double`-type value, and if you print the result it can probably printed out in scientific format. Possible approaches are to just multiply two $n - 1$ times using a 64-bit type integer, or using the bitwise operator: `1ll << n`.