We want to choose integers i and j such that $a_i + a_j - 1$ is divisible by two (i.e. that it is even).

We recall the following properties of integers,

$$\begin{aligned} \text{odd} + \text{odd} &= \text{even}, \\ \text{odd} + \text{even} &= \text{odd}, \\ \text{even} + \text{even} &= \text{even}. \end{aligned}$$

We can formally prove these by returning to the definition of even and odd. Even integers are multiples of two, whereas odd integers are one above a multiple of two. More formally, an integer n is even if there exists an integer k such that $n = 2k$; an integer n is odd if there exists an integer k such that $n = 2k + 1$.

For example, to prove that odd + odd = even, we let the odd numbers be written as $2s + 1$ and $2t + 1$, for some integers s and t. Then, we add these two odd numbers:

$$\begin{aligned} (2s + 1) + (2t + 1) &= 2s + 2t + 2 \\ &= 2(s + t + 1). \end{aligned}$$

If s and t are integers, then $s + t + 1$ is also an integer. Thus, the sum of two odd numbers is an even number, since it can be written as $2 \times$ some integer. Similar logic can be used for the other two cases.

With this, we also see that $a_i + a_j - 1$ is even if and only if $a_i + a_j$ is odd.

If we want $a_i + a_j$ to be odd, then there should be at least one odd and one even number in the given list for the task to be possible. And if not (i.e. the list is all-even or all-odd), then the task is impossible.

Note that even though there are only a fixed 7 integers in the list, we still recommend using a loop in your implementation, as this makes the code cleaner, more debugging-friendly, and less prone to error. We check for even or odd-ness by using the modulo operator `%`, which tells us if there is a remainder after dividing by 2.

```python
n = 7

has_even = False
has_odd = False
for i in range(n):
    x = int(input())
    if x % 2 == 1:
        has_odd = True
    else:
        has_even = True

if has_odd and has_even:
    print("Candies for everyone!")
else:
    print("No candy for Mom :(")
```