**Collaboration Policy:** You are encouraged to collaborate with up to 4 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite them. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

**Collaborators**: list your collaborators

**Sources**: list your sources

## problem 1 *Arithmetic Optimization*

You are given an arithmetic expression containing $n$ integers and the only operations are additions ($+$) and subtractions ($-$). There are no parenthesis in the expression. For example, the expression might be: $1 + 2 - 3 - 4 - 5 + 6$.

You can change the value of the expression by choosing the best order of operations:

$$((((1+2)-3)-4)-5)+6 = -3$$
$$(((1+2)-3)-4)-(5+6) = -15$$
$$((1+2)-((3-4)-5))+6 = 15$$

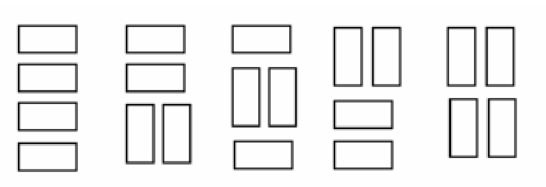
Give a **dynamic programming** algorithm that computes the maximum possible value of the expression. You may assume that the input consists of two arrays: `nums` which is the list of $n$ integers and `ops` which is the list of operations (each entry in `ops` is either `'+'` or `'-'`), where `ops[0]` is the operation between `nums[0]` and `nums[1]`. *Hint: consider a similar strategy to our algorithm for matrix chaining.*

**Solution:**

## problem 2 *More Dominos*

In class, we considered the following problem: suppose you are given a rectangular board of size 2 by $w \geq 1$ (the units here don't matter). You are also given an endless bag of dominoes, each of size 2 by 1. Write an algorithm that accepts the integer $w$ as a parameter and returns the total number of unique ways the 2 by $w$ board can be fully tiled.

Now, for homework, solve the exact same problem as before, except with a board of size 4 by $w$. Describe your algorithm and state its runtime. For example, if the input given is $w = 2$, then the solution is 5, as shown in the following image (all of the combinations of fully tiled boards of width $w = 2$):



**Solution:**

PROBLEM 3 *Backpacking*

You are going on a backpacking trip through Shenandoah National park with your friend. You two have just completed the packing list, and you need to bring $n$ items in total, with the weights of the items given by $W = (w_1, w_2, \ldots, w_n)$. You need to divide the items between the two of you such that the difference in weights is as small as possible. The total number of items that each of you must carry should differ by at most 1. Use dynamic programming to devise such an algorithm, and prove its correctness and running time. You may assume that $M$ is the maximum weight of all the items (i.e., $\forall i,\ w_i \leq M$). The running time of your algorithm should be a polynomial function of $n$ and $M$. The output should be the list of items that each will carry and the difference in weight.

**Solution:**