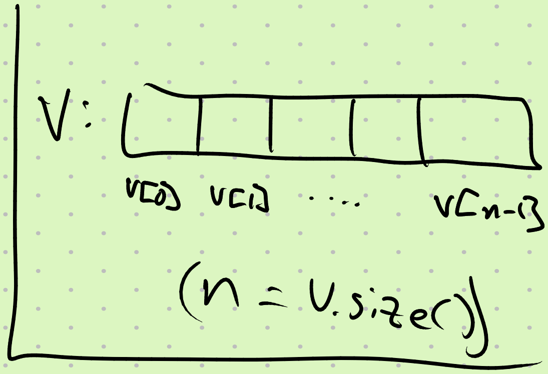
Last time: introduced <u>vectors</u>.

Basic vector stuff:

- Declare one: $\underbrace{\text{vector<int>}}_{\text{datatype of V}}$ V;



- Add an element: V.push_back(3);

- Remove <u>last</u> element: V.pop-back();

- Access/modify an element: V[i] = 7;
- Get the size of V: V.size()

<u>Warnings</u>

$*$ Don't access V[i] if $i \geq$ V.size().

valid elements range from V[0, ..., V.size()-1]

what if you do?

Might not notice ... could read other memory...

if you read far enough OOB, program will crash! (seg fault)

Similarly, don't call V.pop_back() if V is empty.

Basic exercise:
write a function which accepts a vector<int> & returns another vector<int> containing only the even #s.

E.g. if V = $\boxed{3 | 7 | 2 | 8}$,

evens(V) = $\boxed{2 | 8}$

```
~~cout << V ;~~

for(i=0; i<V.size(); i++)
    cout << V[i] << " ";
```

✓

Outline: examine each V[i]. Add even ones to a new vector... then return the new vector.

```
vector<int> evens(vector<int> V)
{   vector<int> E; // will hold the evens...
    for(size_t i=0; i<V.size(); i++) {
        if(V[i] % 2 == 0)
            E.push_back(V[i]);
    }
    return E;
}
```

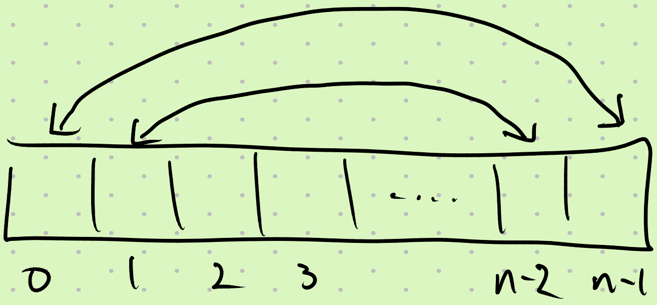
More advanced? Or no?

Exercise: write a function that takes a vector & reverses the order of the elements.

Example: say V = $\boxed{2 | 3 | 4}$.
After calling reverse(V);, V = $\boxed{4 | 3 | 2}$.

observation: V should be by reference.

```
void reverse(vector<int>& V);
```



Say n = V.size()

$0 \longleftrightarrow n-1$

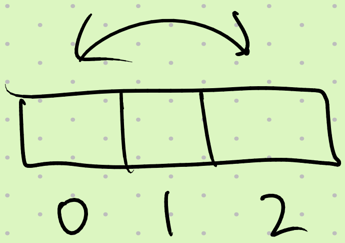
$1 \longleftrightarrow n-2$

$\vdots$

$i \longleftrightarrow n-(i+1)$

$= (n-1)-i$

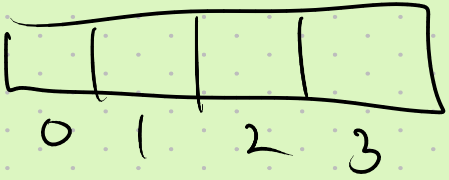
```
void reverse(vector<int>& V)
{   size_t n = V.size();
    for(size_t i = 0; i < n/2; i++) {
        // swap V[i] <--> V[n-1-i]
        int t = V[i];
        V[i] = V[n-i-1];
        V[n-i-1] = t;
    }
}
```



n=3

n/2 = 1

< 1? ✓



n=4, n/2 = 2

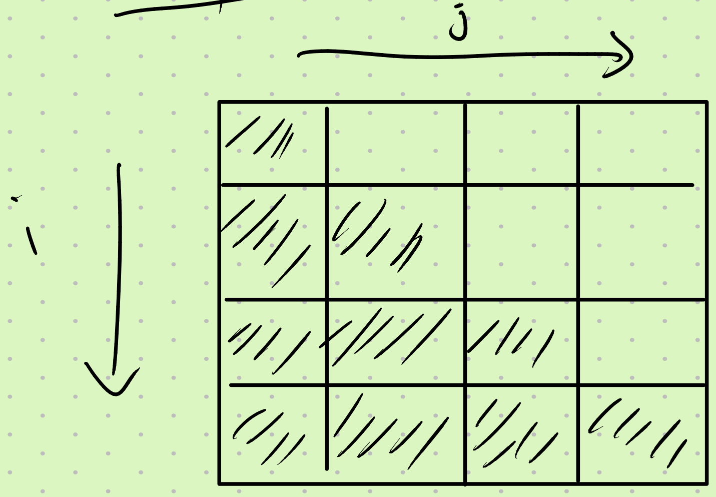
< 2? ✓

Exercise given a vector of integers + a target value t, see if there exist indexes $i \neq j$ s.t.

$$V[i] + V[j] = t.$$

Ideas?. Let's try "Brute force". we'll run through every potential solution + see if anything works...

"potential solution" = pair of indexes (i,j), $i \neq j$



▨ = ruled out.

$V[i] + V[j]$
$=$
$V[j] + V[i]$ ...

So could restrict search space to

$0 \leq i < j < n$