1D Vector Access

To begin, it is helpful to know how to access vector elements of an existing vector. Execute the cells below to see how this can be done:

In [ ]:

```
#include <iostream>
#include <vector>
using std::vector;
using std::cout;

int main() {
    vector<int> a = {0, 1, 2, 3, 4};
    cout << a[0];
    cout << a[1];
    cout << a[2];
    cout << "\n";
}
```

Run Code

Loading terminal (id_3x3ju6v), please wait...

Great! Now try accessing some of the elements of vector `a` yourself in the cell bellow:

In [ ]:

```
#include <iostream>
#include <vector>
using std::vector;
using std::cout;

int main() {
    vector<int> a = {0, 1, 2, 3, 4};
    // Add some code here to access and print elements of a.
    cout << "\n";
}
```

Run Code Show Solution

Loading terminal (id_za35048), please wait...

If you tried to access the elements of `a` using an out-of-bound index, you might have noticed that there is no error or exception thrown. If you haven't seen this already, try the following code in the cell above to see what happens:

```
cout << a[10];
```

In this case, *the behavior is undefined*, so you can not depend on a certain value to be returned. Be careful about this! In a later lesson where you will learn about exceptions, we will discuss other ways to access vector elements that don't fail silently with out-of-range indices.

2D Vector Access

In the previous exercise, you stored a 2D vector - a `vector<vector<int>>`. The syntax for accessing elements of a 2D vector is very similar to accessing in a 1D vector. In the second cell below, try accessing an element of `b`. If you get stuck, click the solution button for help.

In [ ]:

```
#include <iostream>
#include <vector>
using std::vector;
using std::cout;

int main() {
    vector<vector<int>> b = {{1, 1, 2, 3},
                             {2, 1, 2, 3},
                             {3, 1, 2, 3}};
}
```

Run Code Show Solution

Loading terminal (id_cihsz4b), please wait...

Getting a Vector's Length

1D Vector Length

One method of a `vector` object that will be useful in the next code exercise is the `.size()` method. This returns the length of the vector. Execute the cell below to see how this can be used:

In [ ]:

```
#include <iostream>
#include <vector>
using std::vector;
using std::cout;

int main() {
    vector<int> a = {0, 1, 2, 3, 4};

    // Print the length of vector a to the console.
    cout << a.size() << "\n";
}
```

Run Code

Loading terminal (id_jzilsrt), please wait...

2D Vector Length

For the `vector<vector<int>> b` defined above, try to get the size of one of the inner vectors - this should be 4. If you have trouble, click the button below for some help.

In [ ]:

```
#include <iostream>
#include <vector>
using std::vector;
using std::cout;

int main() {

    vector<vector<int>> b = {{1, 1, 2, 3},
                             {2, 1, 2, 3},
                             {3, 1, 2, 3}};
    // Print the length of an inner vector of b here.
    cout << b[2].size();
    cout << "\n";
}
```

Run Code Show Solution

Loading terminal (id_7gepq7h), please wait...

Nice work! You now know a little more about C++ vectors. After learning about for loops, you should be well prepared for the upcoming code exercises.