Introduction to `std::vector`

The `std::vector` from C++ standard library provides a *generic* implementation of *dynamic arrays*. To use `std::vector`, you need to include the header `<vector>`. Also, when declaring variable of type `vector`, you need to specify the type of elements it will contain. Following are some examples of vectors of different types:

```
vector<int> v1; // a vector of int values
vector<double> v2; // a vector of double values
vector<string> v3; // a vector of string values
```

The following code shows how to create a vector of integers using initializer list.

```
#include <vector>
#include <iostream>

int main() {
  using std::vector, std::cout;

  vector<int> v = {1, 2, 3, 4, 5};
  cout << "Size of v: " << v.size() << '\n';

  cout << "Elements of v: ";
  for (int i = 0; i < v.size(); i++)
    cout << v[i] << ' ';
  cout << '\n';
}
```

We call is dynamic array because we can add elements to it at run time. The following example creates a vector of integers and adds elements at it's end using `push_back()` method.

```
#include <vector>
#include <iostream>

int main() {
  using std::vector, std::cout;

  vector<int> v = {1, 2, 3, 4, 5};
  v.push_back(11);
  v.push_back(12);
  // at this point v contains {1, 2, 3, 4, 5, 11, 12}

  cout << "Size of v: " << v.size() << '\n';

  cout << "Elements of v: ";
  for (int i = 0; i < v.size(); i++)
    cout << v[i] << ' ';
  cout << '\n';
}
```

Lab Questions

1. Write a program that takes one integer input n from the user and prints out $n!$, where $0! = 1$ and for $n \geq 1$,

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$$

   Handle the input/output in `main()` function and to compute the factorial, write a function that has the following signature:

```
long long factorial(long long n)
```

   What is the largest value of n that your function can handle without overflow?

2. Write a function with signature '`int count7(int n)`' that given a non-negative integer `n`, returns the count of the occurrences of `7` as a digit, so for example `count7(717)` yields `2`. Following are some more examples

   - `count7(7170123)` should return 2
   - `count7(7)` should return 1
   - `count7(123)` should returns 0

   Hint: Note that mod (%) by 10 yields the rightmost digit (126%10 is 6), while divide (/) by 10 removes the rightmost digit (126/10 is 12).

3. Write a function with signature '`int count_vowels(string s)`' that given a string, returns the count of the occurrences of vowels in the string.

4. Write a function named `percent_even()` that accepts a vector of integers as a parameter and returns the percentage of even numbers in the array as a real number. For example, if a variable named `nums` refers to an vector of the elements `{6, 2, 9, 11, 3}`, then the call of `percentEven(nums)` should return `40.0`. If the vector contains no even elements or no elements at all, return `0.0`.

   Hint: Use the following function signature:

```
double percent_even(vector<int> v)
```

   Following example shows how the function should work:

```
int main() {
    vector<int> v1 = {6, 2, 9, 11, 3};
    cout << percent_even(v1); // should prints 40

    vector<int> v2 = {6, 2, 9, 11, 4};
    cout << percent_even(v2); // should prints 60

    vector<int> v3 = {1, 3, 5, 7, 9};
    cout << percent_even(v3); // should prints 0
}
```

5. Consider the leftmost and rightmost appearances of some value in an array. We will say that the *span* is the number of elements between the two inclusive. A single value has a span of 1.

   Write a function `max_span()` that returns the largest span found in the given array. Use the following function signature:

   ```
   int maxSpan(vector<int> v)
   ```

   Following is an example of how the function should work:

   ```
   int main() {
       vector<int> v1 = {1, 2, 1, 1, 3};
       cout << maxSpan(v1); // should prints 4

       vector<int> v2 = {1, 4, 2, 1, 4, 1, 4};
       cout << maxSpan(v2); // should prints 6

       vector<int> v3 = {1, 4, 2, 1, 4, 4, 4};
       cout << maxSpan(v3); // should prints 6
   }
   ```

6. Write a funtion `vector_eq()` that takes two int vectors as arguments and returns **`true`** if the arrays have the same length and all corresponding pairs of elements are equal, and **`false`** otherwise.

   Hint: Use the following function signature:

   ```
   bool vector_eq(vector<int> v1, vector<int> v2)
   ```

7. Write a function `canBalance()` that given a non-empty `vector`, return **`true`** if there is a place to split the array so that the sum of the numbers on one side is equal to the sum of the numbers on the other side.

   - `canBalance({1, 1, 1, 2, 1})` returns **`true`** because when splitted between index 2 and 3, the sum of `{1,1,1}` is equal to the sum of the numbers on the other side `{2,1}`.
   - `canBalance({2, 1, 1, 2, 1})` returns **`false`**
   - `canBalance({10, 10})` returns **`true`**

8. Write a function that takes three real arguments, x, y, and s, and plots an equilateral triangle centered on (x, y) with side length s. Call the function a number of times in `main()` to produce an entertaining pattern.

   Hint: Vertices of such equilateral triangle of side length s and centered at (x, y) are:

$$A = (x, \quad y + \frac{\sqrt{3}}{2}s)$$
$$B = (x - \frac{s}{2}, \quad y - \frac{\sqrt{3}}{6}s)$$
$$C = (x + \frac{s}{2}, \quad y - \frac{\sqrt{3}}{6}s)$$

9. *SawTooth wave:* Write a program to plot the following function for $0 \leq t \leq 6\pi$.

$$f(t) = \frac{2}{\pi}\left(\frac{\sin(1t)}{1} + \frac{\sin(2t)}{2} + \frac{\sin(3t)}{3} + \cdots\right)$$

As you plot more and more terms, the wave converges to a sawtooth wave.

Hint: Write a C++ function '`double eval(double t, int k)`' that takes t and k as arguments and returns the value of $f(t)$ using the first k terms. See the example we did in the lectures for plotting function using *raylib* library.