// Kevin Lien
// Exception Project 2 Part 1
try
  The code that lives in the try-block will be checked for exceptions. If there is an exception, the try-block will relinquish control to an exception handler and skip the rest of the code in the try-block. If there isn't an exception, the try-block will continue running the code normally, ignoring the exception handlers.

Example try-block
```
try {
  int numGreaterThanTen;
  cin >> numGreaterThanTen;
  if (numGreaterThanTen < 10) {
    cout << "throw some error" << endl;
  }
  cout << "success. no error" << endl;
}
```

throw
  After an exception is found inside the try-block, there needs to be a way to give control to an exception handler. That is where "throw" comes into play. The keyword "throw" needs to be inside of a try-block so that it can direct the try-block to a specific exception handler. The "throw" expression can have one parameter which will be passed to an exception handler. There can be multiple "throw" expressions in one try-block.

Example throw
```
try {
  int numberTen;
  if (numberTen > 10) {
    throw 'e';
  }
  if (numberTen < 10) {
    throw 9;
  }
  cout << "success. no error" << endl;
}
```

catch
  After we throw an exception, we go into an exception handler which is defined by the keyword "catch." A catch-block must follow a try-block to handle the exception caught in the try-block. There can be multiple exception handlers which can be distinguished by its parameter. Exception handlers with the "..." parameter catches exceptions not caught by the previous exception handlers.

Example catch

```
try {
  int numberTen;
  if (numberTen > 10) {
    throw 'e';
  }
  if (numberTen < 10) {
    throw numberTen;
  }
  cout << "success. no error" << endl;
}
catch (int numError) {
  cout << numError << " is less than 10." << endl;
}
catch (char e) {
  cout << "Error: " << e << endl;
}
catch (...) {
  cout << "All other errors." << endl;
}
```

```
#include <stdexcept>
```

Stdexcept is a collection of common error classes divided into two types: logical errors and runtime errors. It allows libraries (such as vectors) and programs to catch common errors.

Example stdexcept

```
try {
  int array[2] = {0, 1};
  int index;
  cin >> index;
  if (index < 0 || index >= 2) {
    throw range_error("index out of range");
  }
  cout << array[index] << endl;
}
catch (exception& error) {
  cout << "Error: " << error.what() << endl;
}
```