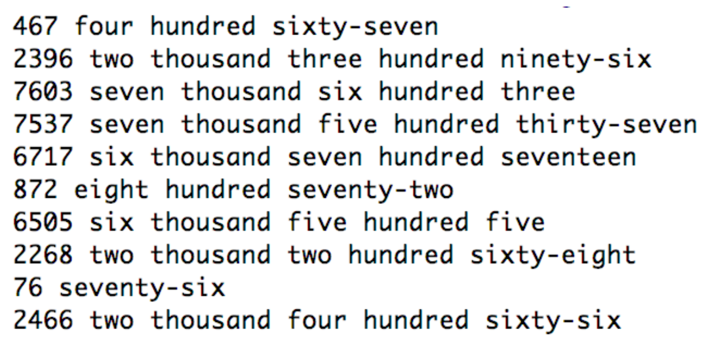
CSC 122 001 Computer Science II
Julius Ranoa

Chapter 11 Programming Challenge 1. Check-Writing.
Write a class that accepts integers (range 0 – 9999, inclusive) and converts that into English descriptions of those integers.

Screenshot of runtime:



```
467 four hundred sixty-seven
2396 two thousand three hundred ninety-six
7603 seven thousand six hundred three
7537 seven thousand five hundred thirty-seven
6717 six thousand seven hundred seventeen
872 eight hundred seventy-two
6505 six thousand five hundred five
2268 two thousand two hundred sixty-eight
76 seventy-six
2466 two thousand four hundred sixty-six
```

Files Included: (1) main.cpp, (2) Numbers.h, (3) Numbers.cpp

main.cpp

```
#include <iostream>
#include <random>
#include "Numbers.h"

int main() {

    Numbers n(0);

    srand(time(NULL));
    for (int i = 0; i < 10; i++) {
        static int x;
        x = rand() % 9999;
        n.setNumber( x );
        std::cout << x << " ";
        n.print();
        std::cout << "\n";
    }

    return 0;
}
```

Numbers.h

```
#ifndef CH11_PR1_CHECK_WRITING_NUMBERS_H
#define CH11_PR1_CHECK_WRITING_NUMBERS_H

#include <string>

class Numbers {

    // Min and Max Definition.
    static const int min;
    static const int max;

    // Int to Text Conversions. Values in Implementation.
    static std::string lessThan20[ ];
    static std::string tens[ ];
    static std::string hundred;
    static std::string thousand;

private:
    int number;
    std::string text;
    std::string stringify(int);

public:
    Numbers();
    Numbers(int);
    bool isInRange(const int&);

    void print();
    std::string getText();
    void setNumber(const int);

};


#endif //CH11_PR1_CHECK_WRITING_NUMBERS_H
```

Numbers.cpp

```cpp
#include <iostream>
#include "Numbers.h"

// Number to Text Conversions - Reference
const int Numbers::min = 0;
const int Numbers::max = 9999;
std::string Numbers::lessThan20[ ] = {
        "zero",
        "one", "two", "three", "four", "five",
        "six", "seven", "eight", "nine", "ten",
        "eleven", "twelve", "thirteen", "fourteen", "fifteen",
        "sixteen", "seventeen", "eighteen", "nineteen"
};
std::string Numbers::tens[ ] = {
        "",
        "ten", "twenty", "thirty", "forty", "fifty",
        "sixty", "seventy", "eighty", "ninety"
};
std::string Numbers::hundred = "hundred";
std::string Numbers::thousand = "thousand";

// Default Constructor
Numbers::Numbers() {
    number = 0;
    text = stringify(number);
}

// Constructor. Accepts int as argument.
Numbers::Numbers(int val) {
    if (!Numbers::isInRange(val)) {
        std::cout << "Number not in range. Exiting program.";
        exit(-1);
    }
    number = val;
    text = stringify(val);
}

// Converts an int value as argument.
// Private function. Assumes value is in range.
std::string Numbers::stringify(int val) {
    if (val == 0) {
        return lessThan20[0];
    }

    std::string temp_str;
    int temp_int;

    // Tens
    temp_int = val % 100;
    if (temp_int < 20 && temp_int != 0) {
        temp_str = lessThan20[temp_int];
    } else {
        temp_str = tens[ temp_int / 10 ];
        temp_int = temp_int % 10;
        if (temp_int != 0) {
            temp_str += "-" + lessThan20[ temp_int % 10 ];
        }
    }

    // Hundred
    temp_int = val % 1000;
    temp_int = temp_int / 100; // Get the hundreds digit.
    if ( temp_int != 0 ) {
        temp_str = lessThan20[ temp_int ]
                   + " " + hundred + " " + temp_str;
    }

    // Thousand
    temp_int = val / 1000;
    if (temp_int != 0) {
        temp_str = lessThan20[temp_int] + " "
               + thousand + " " + temp_str;
    }

    return temp_str;
}

// Tests if value is in accepted range.
bool Numbers::isInRange(const int& val) {
    if (val < min) { return false; }
    else if (val > max ) { return false; }
    else return true;
}

// UX Functions

void Numbers::print() {
    std::cout << text;
}

void Numbers::setNumber(const int val) {
    if (!Numbers::isInRange(val)) {
        std::cout << "Number not in range. Exiting program.";
        exit(-1);
    }
    number = val;
    text = stringify(val);
}
```