PIC 10A 1C Week 10b Exercises. TA: Eric Kim.

1. Retail Therapy

a.

```
class ShopCart {
public:
    ShopCart(); // Initialize with empty cart
    ShopCart(const vector<string>& items); // Initialize with given items
    void add(const string& item); // Adds item to the cart
    size_t size() const; // Returns number of items in cart
    string get_item(size_t i) const; // returns item at index i, or empty string "" if
                                     // not valid index
private:
    vector<string> items;
};
```

A ShopCart is used to store items that we want to buy, ie at the grocery store, or on Amazon:

```
ShopCart mycart;
mycart.add("Scrubs Season 4");    mycart.add("Stuffed Dog");
cout << mycart.size() << endl; // Outputs: 2
```

Define the class implementation that achieves the above desired behavior.

b.

We want to be able to add the contents of one cart to another via the "+=" operator:

```
    ShopCart jdcart;    jdcart.add("Appletini");
    ShopCart janitorcart;    janitorcart.add("pager");
    jdcart += janitorcart;
    cout << jdcart.size(); // Outputs: 2, stores: ["Appletini","pager"]
    cout << janitorcart.size(); // Outputs: 1, stores: ["pager"]
```

Define the "+=" operator to implement the above desired behavior.

c.

Next, define the "<<" operator so that we can display carts in the following way:

```
  vector<string> items = {"knifewrench", "mop"};
  ShopCart janitorcart(items);
  cout << janitorcart << endl; // Displays: ShopCart(2, {"knifewrench", "mop"})
```

In other words, the ShopCart should be displayed as:

```
    ShopCart(<nb. of items>, {"<item1>", "<item2>", ..., "<itemN>"})
```

Note: It's OK for your solution to have an extra space at the end of the list, ie:

```
    ShopCart(2, {"knifewrench", "mop" }); // note the space after "mop"
```

d.

Define the "<" operator so that we can compare carts based on the total number of items in the cart:

```
    ShopCart dr_reid;
    dr_reid.add("coffee"); dr_reid.add("clipboard");
    ShopCart turkleton;
    turkleton.add("pancake");
    if (turkleton < dr_reid)
        cout << "Turk has fewer items than Dr. Reid";
```

2. string2double2string

Suppose I have a vector of strings that contain decimal values:

```
    vector<string> v1 = {"2.0", "1.32", "2.44", "4.2"};
```

Write a function that doubles each of these values, but keeps each value as a string:

```
    funny_double(v1); // v1 is now: ["4.0", "2.64", "4.88", "8.4"]
```

Hint: You'll want to use istringstream and ostringstream.

3. I'm in your base, overloading your dudes.

a.

Consider the following code. What is the expected output? If there is an error, explain why.

```
string foo(int a, int b) {
    if (a < b)
        return "foo";
    return "bar";
}
string foo(int a, double b) {
    if (a < b)
        return "baz";
    return "garply";
}
cout << foo(2, 1) << endl;
cout << foo(2, 1/2) << endl;
cout << foo(2, 1.0) << endl;
```
	Output:

b.

Suppose I added the following function to my file:

```
int foo(int a, double b) {
    if (a < b)
        return 42;
    return 0;
}
```

My code no longer compiles correctly. Why?

4. We all make mistakes

Louis Reasoner wants to write the following program:

(1) Creates a file "mynums.txt", and writes the first 5 even numbers to the file, each on a separate line.
(2) Reads the file we just created ("mynums.txt"), but outputs the square of each number.

Louis codes up the following:

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    /* (1) Write the first 3 even numbers to: mynums.txt */
    ofstream out("mynums.txt");
    for (int i = 0; i < 3; ++i)
        out << i*2 << endl;
    out.close();
    /* (2) Read in mynums.txt, output the square of each number */
    ifstream myfile("mynums.txt");
    int num;
    while (!myfile.eof()) {
        myfile >> num;        cout << num*num << endl;
    }
    cout << "Done!" << endl;    return 0;
}
```

However, the output is strange:

```
0
4
16
16
Done!
```

Why did the last number get output twice?

<u>Hint</u>: Consider that myfile's buffer looks like: "0\n2\n4\n". How would cin/ifstream process this?

<u>Hint 2</u>: After the while loop ends, myfile is in a failure state:

```
    cout << "isfail? " << myfile.fail() << endl; // displays: "isfail? 1"
```

b.

Write a fixed version of the while loop in (2).