CS32 Week 7 Worksheet

This worksheet is entirely **optional**, and meant for extra practice. Some problems will be more challenging than others and are designed to have you apply your knowledge beyond the examples presented in lecture, discussion or projects. All exams will be done on paper, so it is in your best interest to practice these problems by hand and not rely on a compiler.

If you have any questions or concerns, please go to any of the office hours.

Concepts

Templates, STL

1. What is the output of this program?

```
template <class T>
void foo(T input) {
      cout << "Inside the main template foo(): " <<input<< endl;
}

template<>
void foo(int input) {
      cout << "Specialized template for int: " << input << endl;
}

int main() {
      foo<char>('A');
      foo<int>(19);
      foo<double>(19.97);
}
```

   Difficulty: Easy

2. The following code has 3 errors that cause either runtime or compile time errors. Find and fix all of the errors.

```
class Potato {
public:
  Potato(int in_size) : size(in_size) { }
  int getSize() const {
    return size;
  }
private:
  int size;
};

int main() {
  vector<Potato> potatoes;
  Potato p1(3);
  potatoes.push_back(p1);
  potatoes.push_back(Potato(4));
  potatoes.push_back(Potato(5));

  vector<int>::iterator it = potatoes.begin();
  while (it != potatoes.end()) {
    potatoes.erase(it);
    it++;
  }

  for (it = potatoes.begin(); it != potatoes.end(); it++) {
    cout << it.getSize() << endl;
  }
}
```

Difficulty: Easy

3. Create a function that takes a container of integers and removes all zeros while preserving the ordering of all the elements. Do the operation in place, which means do not create a new container. Make sure to have the correct #include commands.

    a. Implement this function using STL list
    ```
    void removeAllZeroes(list<int>& x){
        //Implement me
    }
    ```
    b. Implement the function using STL vectors
    ```
    void removeAllZeroes(vector<int>& x){
        //Implement me
    }
    ```

c. Implement the function that takes a STL vector of pointers, and removes and deletes all pointers that point to integers of value zero

```
void removeAllZeroes(vector<int*>& x){
      //Implement me
}
```

Difficulty: Easy

4. Will this code compile? If so, what is the output? If not, what is preventing it from compiling?

   Note: We did not use `namespace std` because `std` has its own implementation of `max` and `namespace std` will thus confuse the compiler.

```
#include <iostream>

template <typename T>
T max(T x, T y)
{
      return (x > y) ? x : y;
}

int main()
{
      std::cout << max(3, 7) << std::endl;
      std::cout << max(3.0, 7.0) << std::endl;
      std::cout << max(3, 7.0) << std::endl;
}
```

5. Implement a stack class *Stack* that can be used with any data type using templates. Use a linked list (not an STL `list`) to store the stack and implement the functions *push(), pop(), top(), isEmpty(),* a default constructor, and a destructor that deletes the linked list nodes.

6. Implement a vector class *Vector* that can be used with any data type using templates. Use a dynamically allocated array to store the data. Implement only the *push_back()* function, default constructor, and destructor.

--- Problems using STL set --- (may wait until set is covered in class)

7. The following code has 3 errors that cause either runtime or compile time errors. Find and fix all of the errors.

```
class Potato {
public:
  Potato(int in_size) : size(in_size) { }
  int getSize() const {
    return size;
  }
private:
  int size;
};

int main() {
  set<Potato> potatoes;
  Potato p1(3);
  Potato p2(4);
  Potato p3(5);
  potatoes.insert(p1);
  potatoes.insert(p2);
  potatoes.insert(p3);

  set<Potato>::iterator it = potatoes.begin();
  while (it != potatoes.end()) {
    potatoes.erase(it);
    it++;
  }

  for (it = potatoes.begin(); it != potatoes.end(); it++) {
    cout << it.getSize() << endl;
  }
}
```

   Difficulty: Easy

8. You are given an STL `set<list<int>*>`. In other words, you have a set of pointers, and each pointer points to a list of ints. Consider the sum of a list to be the result of adding up all elements in the list. If a list is empty, treat its sum as zero.
   Write a function that removes the lists with odd sums from the set. The lists with odd sums should be deleted from memory and their pointers should be removed from the set. This function should return the number of lists that are removed. You may assume that none of the pointers is null.

```
int deleteOddSumLists(set<list<int>*>& s);
```