CS 32 Worksheet 4

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.

Concepts

Templates, STL

1. 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. 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 the set. This function should also return the number of lists that are removed from the set. If a list is empty, treat its sum as zero. You may assume that none of the pointers is null.

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

2. The following code has 3 errors that cause either runtime or compile time errors. Find 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;
    }
  }
```

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

4. Implement a stack class *Stack* that can be used with any data type using templates. This class should 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.

5. 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);
}
```

6. Implement the vector class *Vector*. Focus purely on the following functionality:
    a. Implementing *push_back*
    b. Allow the vector to hold any type of data (i.e. use templates)

7. What is the output of the following code?

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

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