Assignment 1

Lab Explanation

The requirements for this lab were to create a program that would convert steps to miles and then throw an error message if a negative number was inputted.

Code

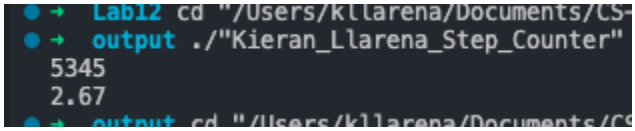
Kieran_Llarena_Step_Counter.cpp Lab12 1

Kieran_Llarena_Step_Counter.cpp > ...

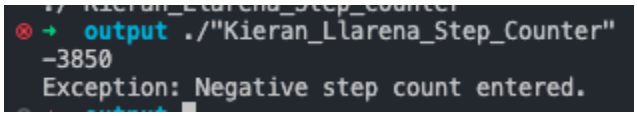
```cpp
#include <iostream>
#include <iomanip>
#include <stdexcept>
using std::cin, std::cout, std::cerr, std::runtime_error, std::fixed, std::setprecision;

double StepsToMiles(int numOfSteps) {
  if(numOfSteps < 0) throw runtime_error("Exception: Negative step count entered.");

  return numOfSteps / 2000.0;
}

int main() {
  int userInput;
  cin >> userInput;

  try {
    double numOfMiles = StepsToMiles(userInput);

    cout << fixed << setprecision(2);
    cout << numOfMiles << '\n';
  } catch(runtime_error& err) {
    cerr << err.what() << '\n';
    return -1;
  }

  return 0;
}
```

The calculations for steps to miles were abstracted into a dedicated function called StepsToMiles. This function has a guard clause that throws a runtime error if the inputted amount of steps were negative.

Test 1


Lab12 cd "/Users/kllarena/Documents/CS-
output ./"Kieran_Llarena_Step_Counter"
5345
2.67

Test 2


output ./"Kieran_Llarena_Step_Counter"
-3850
Exception: Negative step count entered.

Assignment 2

Lab Explanation

The requirements for this lab were to create a program that would store an ordered list of any data type as a template.

Code

```
template<typename TheType>
int OrderedList<TheType>::Size() {
    return list.size();
}

template<typename TheType>
TheType OrderedList<TheType>::At(int index) {
    return list[index];

}

template<typename TheType>
int OrderedList<TheType>::Find(TheType value) {
    for(unsigned int i = 0; i < list.size(); ++i) {
        if(value == list[i])
            return i;
    }
    return -1;
}
```

```
template<typename TheType>
bool OrderedList<TheType>::Remove(TheType oldItem) {
    unsigned int j;
    int indx = Find(oldItem);

    if(indx != -1) {
        list.erase(list.begin() + indx);
        return true;
    }

    return false;
}
```

This code is the implementation code for my ordered list template methods Size(), At(), Find(), and Remove().

Test 1

```
int main() {
    OrderedList<int> intList;

    intList.Insert(11);
    intList.Insert(3);
    intList.Insert(7);

    cout << intList.Size() << '\n';
}
```

```
./"Kieran_Llarena_Ordered_List"
→  output ./"Kieran_Llarena_Ordered_List"
3
→  output
```

Test 2

```
int main() {
    OrderedList<int> intList;

    intList.Insert(11);
    intList.Insert(3);
    intList.Insert(7);

    intList.Print();
}
```

```
→  output cd "/Users/kllarena/Documents/CS-
→  output ./"Kieran_Llarena_Ordered_List"
3 7 11 %
→  output
```

Test 3

```
int main() {
    OrderedList<double> doubleList;

    doubleList.Insert(3.3);
    doubleList.Insert(7.7);
    doubleList.Insert(11.1);
    doubleList.Insert(13.3);

    int index = doubleList.Find(11.1);

    cout << index << '\n';
}
```

```
→  output ./"Kieran_Llarena_Ordered_List"
2
```

Test 4

```
int main() {
    OrderedList<double> doubleList;

    doubleList.Insert(1.1);
    doubleList.Insert(3.3);
    doubleList.Insert(5.5);
    doubleList.Insert(7.7);

    int index = doubleList.Find(9.9);

    cout << index << '\n';
}
```

```
./"Kieran_Llarena_Ordered_List"
-1
```

Test 5

```
int main() {
    OrderedList<string> stringList;

    stringList.Insert("apple");
    stringList.Insert("banana");
    stringList.Insert("mango");
    stringList.Insert("watermelon");

    string valueAtIndex = stringList.At(1);

    cout << valueAtIndex << '\n';
}
```

```
./"Kieran_Llarena_Ordered_List"
banana
```