COMP6771 19T2 (https://webcms3.cse.unsw.edu.au/COMP6771/19T2)

Week 03 - Tutorial - Sample Answers -

Advanced C++ Programming (https://webcms3.cse.unsw.edu.au/COMP6771/19T2)

1. How might we use a lambda function in the following example to sort vec by the length of the strings?

```
    std::vector<std::string> vec{"We", "love", "lambda", "functions"};
```

```
    std::sort(vec.begin(), vec.end(), [] (const auto& lhs, const auto& rhs) { return lhs.size() < rhs.size(
```

2. When writing a lambda function, when would you capture by value, and when would you capture by reference?

   - Capture by value when the variable may change, and you want the variable at the current time
   - Capture by value when the lifetime of the lambda may exceed the lifetime of the variable
   - Capture by reference when the variable may change, and you want to stay updated
   - Capture by reference when the variable is unable to be copied, or expensive to copy

3.
   1. Use standard algorithms to read a list of newline-seperated words from a file (try /usr/share/dict/words or /usr/dict/words) into a vector (hint: see std::istream_iterator)
   2. Write a function that:
      1. Takes in the word list and a string

```
std::vector<std::string> ToWordList(std::istream& input) {
  return {std::istream_iterator<std::string>{input}, std::istream_iterator<std::string>{}
}
```

      2. Uses standard algorithms to split the string into words, filtered to only words that are in the word dict, and reconstruct this into a string (hint: see std::istringstream, std::istream_iterator, std::copy_if, std::ostringstream, and std::ostream_iterator)

```
void PrintValidWords(const std::vector<std::string>& valid_words,
                     std::istream& input,
                     std::ostream& output) {
  std::copy_if(
      std::istream_iterator<std::string>{input},
      std::istream_iterator<std::string>{},
      std::ostream_iterator<std::string>{output, " "},
      [&](const std::string& s) {
        return std::find(valid_words.begin(), valid_words.end(), s) != valid_words.end()
      });
}
```

   3. Discuss why separating your functions you want to test is a good idea

      You can't test any file with a main function in it. Additionally, testing is much easier when you have several small parts to test.

   4. Assume now that the word list and strings are both very large. Discuss how we could make this code run much faster (hint: a different data structure may be required. Tutors, students should know the data type, but not what it is called in C++)

      A std::set would be faster. std::binary_search on a std::vector which was in sorted order would be faster still. std::unordered_set would be fastest.

   5. Discuss the effect the use of automatic type deduction (through the use of auto keyword, and by not having to declare types at all when calling functions) on the quantity of code you had to change, and the depth of the testing required.

When you change the type of the function, only the things that specifically declared their type had to be changed. Anything that declared itself as auto was able to use the same constructors.

4.
1. **Open tutorials/week3/car.(cpp/h):**
2. Create a constructor for the car that takes in the manufacturer name (e.g. Toyota) and the number of seats. Ensure that your constructor uses a member initializer list and uniform initialisation. Why is it important to use a member initializer list? Why is uniform initialisation preferred since C++11?

   Initialiser lists and uniform initialisation avoid having to construct an object once, and then reassign it to a different value after construction. It is more efficient, and for some types, you will not be able to compile without it.

3. Create a default constructor that delegates to the previous constructor using the values of "unknown" and 4
4. Create const member functions to get the manufacturer and number of seats. What does it mean for a class or function to be const correct?

   A const-correct class provides const member functions when no state is changed inside a function, and provides both const and non-const overloads for when a function returns a reference to a data member that the user may be able to modify.

5. Create a static data member to keep count of the number of car objects created. Modify your constructors to ensure that the count increases when a new object is created. Do you need to increase the object count in your delegating constructor?
6. Ensure that your static object count is initialised to 0, where should you do this, in the header file or the cpp file?
7. Create a static function to return the object count. What does it mean for an function or data member to be static? Is the static data member part of the object or the class?
8. Create a destructor that decreases the object count when an object is destroyed
9. Make sure you keep your code - this example will continue next week.

car.cpp

```
#include "tutorials/week3/car.h"

int Car::n_objects = 0;

Car::~Car() noexcept {
  --n_objects;
}

const std::string& Car::GetManufacturer() const {
  return manufacturer_;
}

int Car::GetNumSeats() const {
  return num_seats_;
}

int Car::GetNumCars() {
  return n_objects;
}
```

car.h

```
#ifndef TUTORIALS_WEEK3_CAR_H_
#define TUTORIALS_WEEK3_CAR_H_

#include <string>

class Car {
 public:
  Car(): Car{"unknown", 4} {}
  Car(const std::string& manufacturer, int n_seats): manufacturer_{manufacturer}, num_seats_{
    ++n_objects;
  }

  ~Car() noexcept;

  const std::string& GetManufacturer() const;
  int GetNumSeats() const;

  static int GetNumCars();

 private:
  std::string manufacturer_;
  int num_seats_;

  static int n_objects;
};

#endif  // TUTORIALS_WEEK3_CAR_H_
```