Instructor Solution

C++ Week One Quiz

Q1: What is wrong with the following code that prints the first line of a file (Hint: There is more than just one thing wrong)?

```
#include <ifstream>

#include <string>
#Include <iostream>

int main(){

  std::ifstream inFile("myFile.txt");   <- no checking to see if the file is open without errors

  ~~inFile.open("myFile.txt");~~   //opening twice

  std::string line;

  std::get_line(inFile, line);   <- should be getline

  std::cout << line << std::endl;

  return 0;

}
```

```
if (!inFile)
{
    std::cerr << "Bad File \n";
    std::exit(-1);
}
```

Q2: How many bytes is a int, long and a pointer on a LP64 machine?

int 32 bits/4 bytes long, pointer 64 bit/8 bytes.

Q3: How would you use a lambda to bind parameter one of foo to 5?

```
int foo(int a, int b);

…

auto fooLambda = [](int a){ ~~foo(5, a~~ return foo(5, a); };
```

Q4: How would you use bind to do the same thing as in Q3?

```
int foo(int a, int b);

…

auto fooBind = std::bind( foo, 5, std::placeholders::_1);
```

Q5: If I want to pass a std::string to foo as an in parameter, how would I do it and why?

```
void foo(const std::string s);

void foo(const std::string* s);

void foo(const std::string& s);   <-

void foo(std::string s);
```

more effecient than a copy and const so it won't change the callers version.

Q6: If I wanted to pass an int to foo as an in parameter how would I do it and why?

[✓ `void foo(int x );`] – primitive type are just as cheap to copy and no indirections to use.

`void foo(const int& x);`

`void foo(const int* x);`

`void foo(int* x);`

Q7: What is the main difference between a share_ptr<int> and a unique_ptr<int>?

shared_ptr - mean multiple owners and can by copied
unique_ptr - one owner and can not by copied.

Q8: How do I declare a pointer that is const but not the data that it points to?

int* const ptr;

Q9: constexpr objects are const and their values are known at compile time.

Q10: What type is X below:

```
int a = 6;
int& b = a;
auto x = b;
```

int

Q11: Give an example of a narrowing conversion: int x = 0XDEADBEEF;
short y = x; //narrowing conversion

Q12: What is the most recently ratified C++ standard?

C++ 14

Q13: What is a type that I might use if I needed to rely on items I put into it being contiguous in memory?

std::vector std::array

Q14: What would I use for a look up table that was constantly growing and shrinking?

std::map unordered map would be expensive to move the bins around durring resize

Q15: What is wrong with the overloaded functions in the code below?

```
void foo();
void foo(int x = 5);
```

The compiler can't figure out which to call for foo();