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

Week 01 Tutorial Sample Answers

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

1. Your tutor will take some time to get to you, and for you to get to know them.

2. Write a program addnumbers.cpp that reads in two integers and prints out the sum of them. It's behaviour should follow:

```
$ ./prog
Please enter two numbers: 2 3
Sum: 5
```

```
#include <iostream>

int main() {
  int a, b;
  std::cout << "Please enter two numbers: ";
  std::cin >> a >> b;
  std::cout << "Sum: " << (a + b) << "\n";
  return 0;
}
```

3. Write a program in C++, cat.cpp, equivalent to the below C program? For each change, what advantages does C++ provide?

```
int main() {
  char buffer[100];
  fgets(buffer, 100, stdin);
  printf("%s\n", buffer);
  return 0;
}
```

```
#include <iostream>
#include <string>

int main() {
  // a std::string is a dynamically-sized character array, which
  //  means we are not required to worry about the fixed-sized arrays
  //  that we're used to in C. In this way it makes it harder to overflow
  //  a char array or waste memory. std::string is also a class meaning
  //  a lot of important pieces of data associated with the types are stored in
  //  the object itself (e.g. size). Strings also have the added benefits
  //  of providing an iterator to loop through the string
  std::string buffer;


  // getline, although slower than fgets, is easier to use as
  //  no explicit buffer size or maximum size is required. That's handled
  //  by the steram and string object. fgets also doesn't work with
  //  a string, but instead only a raw character array
  std::getline(std::cin, buffer);

  // printf has some benefits such as complex format strings, however,
  //  using streams and cout gives more power by allowing for the use
  //  of operator overloading, so that objects themselves have more
  //  control over how they're printed out
  std::cout << buffer << "\n";
  return 0;
}
```

4. Which parts of this program are declarations and which parts are definitions?

```
#include <iostream>
#include <string>

int getAge();

int main() {
  std::string name;
  name = "Hayden";
  std::cout << name << " is " << getAge() << "\n";
  return 0;
}

int getAge() {
  return 24;
}
```

Declarations:

- int getAge();

Definitions:

- std::string name;
- int getAge() { return 24; }

5. Do any of the following have errors? If so, what are they?
   1. ```
      int i = 3;
      i = 4;
      ```
   2. ```
      const int j = 5;
      j--;
      ```
   3. ```
      int age = 18;
      int& myAge = age;
      myAge++;
      ```
   4. ```
      int age = 21;
      const int& myAge = age;
      myAge--;
      ```

   1. No issue
   2. ERROR: j is an integer constant which cannot be modified.
   3. No issue
   4. ERROR: myAge is a reference to const int, which cannot be modified.

6. Write a C++ program, sort3.cpp that prompts the user to enter three ints, calls an "sort3" function by reference and prints the sorted numbers.

   Starting point:

```
#include <iostream>

void sort3(int& a, int& b, int& c);

int main() {
  // TODO
  // sort3(....);
  // TODO
  return 0;
}

void sort3(int& a, int& b, int& c) {

};
```

```
#include <iostream>

void sort3(int& a, int& b, int& c);

int main() {
  int a, b, c;
  std::cout << "Enter 3 integers: ";
  std::cin >> a >> b >> c;
  sort3(a, b, c);
  std::cout << a << ", " << b << ", " << c << "\n";
  return 0;
}

void sort3(int& a, int& b, int& c) {
  if (a > c) {
    int t = a;
    a = c;
    c = t;
  }
  if (b > c) {
    int t = b;
    b = c;
    c = t;
  }
  if (a > b) {
    int t = a;
    a = b;
    b = t;
  }
}
```