CMP2801M Workshop Week 8

FUNCTION OVERLOADING AND COPY CONSTRUCTORS

Write and test your code, and add comments so that you can refer to it later (focus on *why* you made a particular decision to do something a certain way). This will help you to understand your work when you refer to it in future. Please refer to the lecture notes or ask for help if needed! There are two parts to this workshop: **I)** looking back to the last topic, for review and practice of Function Overloading and Copy Constructors, and **II)** exploring the fundamentals of this week's topic, Operator Overloading and Friends.

PART I – Function Overloading and Copy Constructors

Task 1:

Write a function `variableType(…)`, which is overloaded twice more. Each of the three functions should take one argument: `int, double, string`. The functions should print to screen a message indicating the variable type. Test with a range of variable values (in `main`). Example inputs/outputs:

```
2 – This variable is an integer
0.123 – this variable is a double
“Lovely weather” – this variable was a string
```

Extension task: modify this program to take user input from the console. Note: for this task, you should be careful about type casting and conversion between types.

Task 2:

Write a program to sort a `vector` of numbers, where the `vector` could be populated with `integers`, `long` values, or `double` values. The appropriate vectors should be initialised within your program. The function(s) should take two arguments: the `vector` of values (consider how this should be passed in), and a Boolean value indicating whether the numbers should be sorted in ascending order (`true`), or descending order (`false`).

Task 3:

Given the following code (`main()` function), implement an appropriate class to ensure that it runs as would be expected given that appropriate copies are being made.

```
int main() {
      A *a1 = new A(“a”);
      a1->setName(“IamA”);
      A *a2 = *a1;
      delete a1;
      a2->printName();
}
```

PART II – Operator Overloading and Friends

In this part of the workshop, we preview/review (depending on whether you have seen the lecture yet this week or not) the topic for week 8. More in depth exercises on this topic will come in the workshop next week (week 9), as in previous weeks.

Task 4:

The workshop in week 7 (last week), Task 5, actually gave an example of the assignment operator, NOT the copy constructor (this was corrected in the lecture). In this task, please review Task 5 from workshop 7 and identify where an assignment operator would have been invoked. Following the example in the lecture (week 7), determine what the implications of this are in the context of memory management.

Task 5:

Given the following code snippets (you will have to implement the rest of the code as appropriate to enable compilation), determine what is happening, and why this is the case. Please refer to the compiler errors/warnings when doing so. How could this situation be resolved?

```
class Example {
public:
      getInt() { return i; }
private:
      int i;
};

int getMemberInt(Example &in) {
      Return in.i;
}
```