CMPT 135: Midterm Answer Key

Last name *exactly as it appears on your student card*	**First name** *exactly as it appears on your student card*

Student Number									

SFU Email		**Section** if you know it!	

This is a **50 minute** test. It is **closed book**: no calculators, computers, notes, books, etc. are allowed.

Important: Do **not** use any C++ library functions unless a question specifically permits it. Also, use only features of C++ discussed in the lectures and lecture notes.

Question	Out Of	Your Mark
Arrays	8	
Pointers and Dynamic Arrays	10	
Classes and Objects	20	

Total	**38**	

Arrays

a) (3 marks) Write a fragment of C++ code that creates a variable named `temps` that is of type array of `double` (*not* a pointer!). Make it of length 500, and you use a loop to initialize all its value to 0.

Sample solution:

```
double temps[500];
for(int i = 0; i < 500; ++i) {
   temps[i] = 0;
}
```

b) (5 marks) Write a function that calculates and returns the sum of all the elements in *any* array of `double`s. The passed-in array should be of type array of `double` (*not* a pointer!). Write both the function header and its body.

Sample solution:

```
double sum(double arr[], int n) {
   double result = 0.0;
   for(int i = 0; i < n; ++i) {
      result += arr[i];
   }
   return result;
}
```

Pointers and Dynamic Arrays

a) (1 mark) Write a fragment of C++ code that defines a variable `x` to be a `double` with the value 5, and then defines a pointer `p` that points to `x`.

Sample solution:

```
double x = 5;
double* p = &x;
```

b) (2 marks) Write a fragment of C++ code that creates a new array of 150 `double`s on the free store, and then immediately afterwards de-allocates that array.

Sample solution:

```
double* arr = new double[150];
delete[] arr;                    // [] is required!
```

c) (2 marks) Suppose `arr` points to an array of 150 `double`s on the free store. Write a fragment of C++ code that prints each element `arr` to the screen. **Important**: your code fragment must access the elements of `arr` **without** using []-notation anywhere.

Sample solution:

```
for(int i = 0; i < 150; ++i) {
    cout << *(arr + i);
}
```

d) (5 marks) Write a function called `make_fill(n, val)` that returns a pointer to a newly created array of `double`s of length `n`. Each element of the returned array should have the value `val`. If `n` is less than 0, then cause an error using `cmpt::error`.

Sample solution:

```
double* make_filled(int n, double val) {
    if (n < 0) cmpt::error("n must be 0 or greater");
    double* result = new double[n];
    for(int i = 0; i < n; ++i) {
        result[i] = val;
    }
    return result;
}
```

Classes and Objects

(20 marks) Write a class called `Student` that stores the name (as a `string`) and `age` (as an `int`) of a university student. You class must have these features:

- All class variables are **private**, and all methods are **public**.
- A **default constructor** that uses an **initialization list** to set the student's name to "none" and age to -1.
- A **constructor** that uses an **initialization list** to set `name` and `age` to values passed into the constructor.
- A **copy constructor** that uses an initialization list to set the student's name and age to be the same as the name and age of another passed-in `Student` object.
- A **destructor** that prints the message "object deleted" when the `Student` object it's part of goes out of scope, or is deleted.
- A **getter method** that returns the name of the student, and another **getter method** that returns their age. Make sure these can be used with constant `Student` objects.
- Define an `<<` operator that lets you print the name and age of a `Student` object. Importantly, define this `<<` *outside* of the `Student` class. No special format is required: print the name and age in any convenient way.

Sample solution:

```
class Student {
private:
    string name;
    int age;
public:
    Student();                          // default constructor
    Student(const string& n, int a);
    Student(const Student& other);      // copy constructor
    ~Student();                         // destructor

    string get_name() const;
    int get_age() const;
}; // class Student

bool operator==(const Student& a, const Student& b) {
    return (a.get_name() == b.get_name())
        && (a.get_age() == b.get_age());
}

ostream& operator<<(ostream& out, const Student& s) {
    out << "(" << s.get_name() << ", " << s.get_age() << ")";
    return out;
}

Student::Student()
: name("none"), age(-1)
{ }

Student::Student(const string& n, int a)
: name(n), age(a)
{ }

Student::Student(const Student& other)
: name(other.name), age(other.age)
{ }

Student::~Student() {
    cout << *this << " deleted\n";
}

string Student::get_name() const { return name; }
int Student::get_age() const { return age; }
```