CMPS 222 Midterm 2 Sample Questions

1. (5pts) In general, what is the difference between a copy constructor and an assignment operator? When answering, consider what the assignment operator has to be concerned about that the copy constructor does not need to be concerned about.
2. You have the files `mystring.h`, `mystring.cpp` and `main.cpp`. Give the commands to create the executable named `mystring` on Sleipnir.
3. (10pts) When your class has a dynamic array as a member variable, certain features need to be present in your class for the dynamic array to work correctly. For each of the following features, **briefly** state what would go **wrong** if you neglected to include it in your class:
   (a) The default constructor
   (b) The destructor
   (c) The assignment operator
4. (10pts) When using inheritance, there are two forms of protection for member variables and member functions in a class. The first form of protection is the protection tag used for the sections within the parent class. The second form of protection is the protection tag used by the child class when it inherits the parent. Answer the following questions about protection levels.
   (a) The protection sections inside a parent class define not only how the outside world can access the member variables and member functions in that section but also how the children class(es) can access that section. For the following protection tags, state whether the child can access that section **AND** whether the outside world can access that section:
      i. Protected section in the parent class
      ii. Private section in the parent class
   (b) The protection tag used by the child class when it inherits from the parent class affects how the outside world and the grandchild class(es) can access a protection section in the parent class. For the following protection tags, state how it affects the way the grandchildren and outside world can access the **public** section of the parent class:
      i. Public inheritance of the parent class
      ii. Protected inheritance of the parent class
5. (5pts) You have the parent class `Employee`, as given in Lab 6, which defines the name and social security number for an employee. You wish to define a new child class from `Employee` called `salariedEmployee`. The `salariedEmployee` class will have a private member variable called `salary` and the public member functions `getSalary`, `setSalary`, `printPaycheck` and `setEmployee`. Give **just** the class definition for this child class with the member function and default constructor prototypes. You do not need to give any function bodies for the functions.

6. (5pts) You wish to add a copy constructor to your `salariedEmployee` class from Question 5. Assume that the parent class `Employee` has a copy constructor defined which copies the name and social security number from the source object. Give the **inline** form of the copy constructor for `salariedEmployee` which invokes the copy constructor of the parent class `Employee` and then copies the salary from the source object.

7. (5pts) There are ten logic and syntax errors in the following code. Identify at least five errors and state how to correct them.

   NOTE: This question continues onto the next page. Make sure to check the next page for more errors.

```
#include <iostream>
using namespace std;

class DoubleList {
  private:
    double *array;
    int size;

  public:
    DoubleList() : size(0) {}
    ~DoubleList() { delete [] array; }

    bool allocate(int);

    DoubleList(const DoubleList&);
    DoubleList& operator=(const DoubleList&);

    double operator [](int);
    friend ostream & operator <<(ostream &, const DoubleList &);
};

bool DoubleList::allocate(int num) {
  if(num <= 0) return false;
  if(num <= size) return true;

  if(array != NULL) {
    delete [] array;
    array = NULL;
  }
  try {
    array = new double[size];
  } catch(bad_alloc) {
    array = NULL;
  }
  if(array == NULL) {
    size = 0;
```

```
    return false;
  }
  size = num;
  return true;
}

DoubleList::DoubleList(const DoubleList &source) {
  array = NULL;
  size = 0;
  if(allocate(source.size)) {
    for(int i = 0; i < size; i++)
      array[i] = source.array[i];
  }
}

DoubleList& DoubleList::operator=(const DoubleList &source) {
  if(allocate(source.size)) {
    for(int i = 0; i < size; i++)
      array[i] = source.array[i];
  }
}

double DoubleList::operator[](int index) {
  return array[index];
}

ostream & operator <<(ostream &o, const DoubleList &right) {
  for(int i = 0; i < right.size; i++) {
    o << right.array[i];
  }
}

int main() {
  DoubleList a;

  if(a.allocate(10)) {
    for(int i = 0; i < 20; i++)
      a[i] = i;
  }

  cout << a << endl;
  return 0;
}
```