CMPS 222 Sample Midterm 2

Part 1: Concepts - Each question is worth 5 points, 30 points in total.

1. Define the following terms by stating their purpose when used in your code:
   (a) Parent/base class
   (b) Polymorphic function
   (c) Redefined function
2. When a child object is passed to a function which expects a parent object, a phenomena called slicing may occur. Give an example of when slicing would occur and explain why slicing is necessary when using a parent datatype to refer to a child object in that situation.
3. Polymorphic functions differ from normal functions due to runtime binding of the function call to the function body. Describe how dynamic/runtime/late binding differs from the static binding which is used for normal or redefined functions.
4. When using separate compilation, what is the purpose of putting

```
#ifndef __HEADER_H__
#define __HEADER_H__
// Header file
#endif
```

   in the header/interface file?
5. When your class has a dynamic array as a member variable, certain features need to be present in your class. For each of the following features, briefly state what would go **wrong** if you neglected to include it in your class:
   (a) Destructor
   (b) Copy constructor
6. When using inheritance, there are two forms of protection for member variables and functions. The first form of protection is the protection tag used for the section within the parent class. The second form is the inheritance protection tag used by the child class when inheriting from the parent. For the following inheritance scenarios, answer "Yes" if the section is accessible or "No" if it is not accessible:

	Accessible to child?	Accessible to grandchild?	Accessible to world?
Protected section in parent, inherited publicly			
Public section in parent, inherited privately			

Part 2: Coding - 20 points in this section

1. (5 pts) Add the copy constructor and destructor to the following myString class. Make sure that your code properly handles the pointer for the dynamically sized character array.

```
class myString {
  private:
    char *str;
    int size;
    void allocateArray(int);

  public:
    myString();

  //*************************************************************************
  // Add prototypes for copy constructor and destructor
  //*************************************************************************
};

myString::myString() {
  str = NULL;
  size = 0;
  allocateArray(21);
}

void myString::allocateArray(int num) {
  if(num < 0 || num <= size) return;
  if(str != NULL) {
    delete [] str;
    str = NULL;
    size = 0;
  }
  try {
    str = new char[num];
  } catch(bad_alloc) {
    str = NULL;
  }
  if(str == NULL) {
    cout << "Allocation failed!\n";
    exit(1);
  }
  size = num;
  str[0] = '\0';
}

//*************************************************************************
```

```
// Add the bodies for copy constructor and destructor
//************************************************************************
```

2. (10 pts) You have the parent class Employee which contains member variables for the name and SSN of the employee. You also have the child class salariedEmployee which contains a member variable for the monthly salary. Add the following specified features to these two classes

   (a) Give the function prototype that would be added to both class definitions to add a polymorphic `void print_check()` function.

   (b) Give the `class salariedEmployee` line which would inherit from Employee privately.

   (c) Assume that Employee contains a default constructor which sets the name and SSN to the empty string. Give the body of the default constructor for salariedEmployee which invokes Employee's default constructor then sets salary to 0.

   (d) Assume that Employee contains an assignment operator which appropriately copies the name and SSN from the source object. Give the body of the assignment operator for salariedEmployee. The assignment operator for salariedEmployee must call Employee's assignment operator.

3. (5 pts) The following code contains multiple syntax and logic errors. Circle the errors and briefly explain why they are errors. You will not be penalized for any incorrect answers. Each correctly found error is worth one point, up to the maximum of five points.

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

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

class DoubleList {
  private:
    double *array;
    int capacity;
    void copyObject(const DoubleList &);

  public:
    DoubleList() { capacity = 0; }
    ~DoubleList() { delete [] array; }
    DoubleList(const DoubleList &) { copyObject(source); }
    DoubleList& operator=(DoubleList &);
    bool allocate(int);
    double operator[](int index) { return array[index]; }

    friend ostream & operator <<(ostream &o, const DoubleList &right);
};

void DoubleList::copyObject(const DoubleList &source) {
  array = NULL;
  capacity = source.capacity;

  array = new double[capacity];
  if(array == NULL) capacity = 0;
  else {
    for(int i = 0; i < capacity; i++)
      array[i] = source.array[i];
  }
}

DoubleList& DoubleList::operator=(const DoubleList &source) {
  if(this == source) return *this;
  copyObject(source);
  return *this;
}

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

```
  if(num >= capacity) return true;

  if(array == NULL) {
    delete [] array;
    array = NULL;
  }

  array = new double[capacity];
  if(array == NULL) {
    capacity = 0;
    return false;
  }
  capacity = num;
  return true;
}

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

int main() {
  DoubleList a, b;

  if(a.allocate(10)) {
    cout << "Enter 10 doubles: ";
    for(i = 0; i < size; i++)
      cin >> a[i];
  }
  b = a;
  cout << b << endl;
  return 0;
}
```