Sample Final Exam
EC327 Introduction to Software Engineering – Spring 2020

Total: 100 points

Q1. – Written Questions

a. What possible error could occur when running the following code? How would you fix it? Note that Bird inherits from Animal, and defines a method chirp() that does not exist in animal.

[5 points]

```
void movesHow(Animal& a) {
        Bird* pb = static_cast<Bird*>(&a);
        pb->chirp();
}
```

The program might crash when it tries to static_cast because not all the inherited classes have the chirp function. To fix this, we can use dynamic _cast, which casts during runtime and returns NULL if fails instead of terminating the program.

b. What does including the "virtual" keyword in the declaration of a member in a base class do?

[5 points]

If a function is declared virtual, it enables the system to decide which function is invoked at runtime based on the actual type of the object.

Abstract functions are called pure virtual functions. A class contains pure virtual functions becomes an abstract class. A pure virtual does not have a body or implementation in the abse class.

c. When a class inherits publicly from a base class, which members from the base class are retained in the derived class?

[5 points]

Public inheritance: public members are inherited as public members; protected members are inherited as protected members. Private members are not inherited.

d. Specify if the following statements are true or false. **[10 points, 2 per correct answer]**

Variables that are dynamically allocated “survive” function calls and a pointer to them can be returned.	True
If a copy constructor is not implemented in a class A, calling A x = y (where y is an instance of A) causes x to become a deep copy of y.	False
If a variable is declared as static inside a class, the variable can be accessed from the main function even if no object of that class was instantiated.	True
Protected variables of a class are directly accessible from the main function.	False
When an object of a derived class is instantiated, the default constructor of the base class is called after the constructor for the derived class.	False

Q2. - Files to submit: Q2.cpp. You are provided a header file Q2.h with the class declaration and a skeleton implementation at Q2.cpp. You are also provided a sample main Q2main.cpp to test your implementation. The file Q2output.txt contains the output that Q2main.cpp should produce if your class is implemented correctly. You can either compile your files manually to test them or use the provided makefile.

Consider a class Account representing a bank account (as declared in `Q2.h`). The class declares the following private members:

- `unsinged int id`, which contains the account id
- `string owner`, which contains the first and last name of the account owner
- `int balance`, which contains the dollar amount available in the account

a) Implement a default constructor that sets id and amount to 0, and owner to "X".

[5 points]

b) Implement a constructor `Account(int id, string owner, int balance)` that sets the object members to the ones provided as parameter in the constructor.

[5 points]

c) Implement getters and setters for all members as declared in `Q2.h`.

[5 points]

d) Implement a method `bool enoughBalance(unsigned int x)` which returns true if there is enough balance in the account (that is, if balance is greater or equal than x) and false otherwise.

[5 points]

e) Implement a method `void withdraw(int x)` that decreases the account balance by the amount specified in x. If the account does not have enough balance, the method should throw a `bad_exception` exception.

[5 points]

Q3. - Files to submit: Q3.cpp and Q3.h. You are provided a skeleton header file Q3.h and a skeleton implementation Q3.cpp. You are also provided a sample main Q3main.cpp to test your implementation. The file Q3output.txt contains the output that Q3main.cpp should produce if your class is implemented correctly. You can either compile your files manually to test them or use the provided makefile.

You are given an abstract class Vehicle, declared in `Q3.h` and implemented in `Q3.cpp`. Vehicle declares the following members:

- `unsigned short year`, containing the year in which the vehicle was build
- `unsigned int mileage`, containing the miles that the vehicle traveled

It also implements getters and setters for year and mileage, and a pure virtual function void move() that specifies how the vehicle moves.

Declare a class Car that publicly inherits from Vehicle (in Q3.h). A Car object should have the following additional members:

- a `string make`, containing the make of the car
- a `string color`, containing the color of the car

Implement the following methods for the class Car, declaring them in Q3.h and implementing them in Q3.cpp:

a) appropriate setters and getters for color (`void setColor(string x)`, `string getColor()`) and make (`void setMake(string x)`, `string getMake()`)
[5 points]

b) a constructor `Car(short year, string make, string color, unsigned int mileage)` that sets the members of Car to the parameters passed to the function
[5 points]

c) a copy constructor that allows you to copy an object of Car into another object, but that sets the `mileage` of the new car to 0
[5 points]

d) an implementation of the `void move()` method that prints out the make of the car and the fact that cars drive. For example, if the make of an instance of Car is Toyota, calling move() should print on the screen

```
A Toyota drives
```

[5 points]

e) overload the output stream operator << so that when it is called on an instance of Car it prints out the year, the color, the make, and the mileage, so that `cout << c << endl;` would print

```
1987 red Toyota 156793
```

[5 points]

Declare the class Car and its members and methods in `Q3.h` and implement it in `Q3.cpp`.

Q4. - Files to submit: Q4.cpp. You are provided a skeleton implementation of your submission in Q4.cpp. You are also provided a set of example input and output files as detailed in the README file, to help you test that your solution is correct. You can either compile and test your file by hand or use the provided makefile.

This question requires you to use data structures that are implemented by the Standard Template Library.

Write a program that allows the user to compile a simple grocery list. A skeleton for the program is provided in `Q4.cpp`. The user is asked to provide a new grocery item (as a string) until they enter N when asked if they want to insert a new item. Implement the following functionalities:

a) Create a List of strings to store the items provided by the user

[5 points]

b) Every time a new item is provided, first check that an identical item does not already exist in the list. If it already exists, print out the following message:

```
An item with that name is already included in the list!
```

If the item does not exist, add it to the list.

[10 points]

c) After the user is done inserting items, sort them in alphabetical order.

[5 points]

d) After the user selects that they don't want to add any more items, write the list of items to a file named grocery_list.txt, one item per line.

[5 points]