Program: Computer Science (BS)
Semester: Spring-2021
Time Allowed: 03 hours
Course: OOp

Examination: Final
Total Marks: 80 Weightage: 50
Date: 30/6/21
Instructor Name: Shoaib

NOTE: Attempt all questions.

1 Create an Inventory class that a warehouse might use to represent their stock of products and raw materials. Include a data member of type `string` to provide a description of the product, and data member of type `int` to represent the balance stock. Provide a constructor that receives an initial product and uses it to initialize the data members. The constructor should validate the initial product to ensure it has a stock greater than 20, which is the company's minimum stock level. If not, it should display an error message. Provide three member functions. Member function `Purchase` should add a product to the current stock. Member function `Sale` should reduce stock. Ensure after each sale that the stock level does not drop below 20. Member function `getStock` should return the current stock. Create a program that creates two `Inventory` objects and tests the member functions of the class. 15 marks

2 Write a class that contains two class data members *numBorn* and *numLiving*. The value of *numBorn* should be equal to the number of objects of the class that have been instanced. The value of *numLiving* should be equal to the total number of objects in existance currently (ie. the objects that have been constructed but not yet destructed.) 10 marks

3 Write the definitions of the member functions. 20 Marks

```
class arrayListType
{
public:
    bool isEmpty() const;
    bool isFull() const;
    int listSize() const;
    int maxListSize() const;
    void print() const;
    bool isItemAtEqual(int location, int item) const;
    void insertAt(int location, int insertItem);
    void insertEnd(int insertItem);
    void removeAt(int location);
    void retrieveAt(int location, int& retItem);
    void replaceAt(int location, int repItem);
    void clearList();
    int seqSearch(int item) const;
    void insert(int insertItem);
    void remove(int removeItem);
    arrayListType(int size = 100);
    arrayListType (const arrayListType& otherList);
    ~arrayListType();

protected:
    int *list;      //array to hold the list elements
    int length;     //variable to store the length of the list
    int maxSize;    //variable to store the maximum
                    //size of the list

};
```

4 Create an inheritance hierarchy that a bank might use to represent customers' bank accounts. All customers at this bank can deposit (i.e., credit) money into their accounts and withdraw (i.e., debit) money from their accounts. More specific types of accounts also exist. Savings accounts, for instance, earn interest on the money they hold. Checking accounts, on the other hand, charge a fee per transaction (i.e., credit or debit). 20 marks

Create an inheritance hierarchy containing base class `Account` and derived classes `SavingsAccount` and `CheckingAccount` that inherit from class `Account`. Base class `Account` should include one data member of type `double` to represent the account balance. The class should provide a constructor that receives an initial balance and uses it to initialize the data member. The constructor should validate the initial balance to ensure that it's greater than or equal to `0.0`. If not, the balance should be set to `0.0` and the constructor should display an error message, indicating that the initial balance was invalid. The class should provide three member functions. Member function `credit` should add an amount to the current balance. Member function `debit` should withdraw money from the `Account` and ensure that the debit amount does not exceed the `Account`'s balance. If it does, the balance should be left unchanged and the function should print the message "`Debit amount exceeded account balance.`" Member function `getBalance` should return the current balance.

Derived class `SavingsAccount` should inherit the functionality of an `Account`, but also include a data member of type `double` indicating the interest rate (percentage) assigned to the `Account`. `SavingsAccount`'s constructor should receive the initial balance, as well as an initial value for the `SavingsAccount`'s interest rate. `SavingsAccount` should provide a `public` member function `calculateInterest` that returns a `double` indicating the amount of interest earned by an account. Member function `calculateInterest` should determine this amount by multiplying the interest rate by the account balance. [*Note:* `SavingsAccount` should inherit member functions `credit` and `debit` as is without redefining them.]

Derived class `CheckingAccount` should inherit from base class `Account` and include an additional data member of type `double` that represents the fee charged per transaction. `CheckingAccount`'s constructor should receive the initial balance, as well as a parameter indicating a fee amount. Class `CheckingAccount` should redefine member functions `credit` and `debit` so that they subtract the fee from the account balance whenever either transaction is performed successfully. `CheckingAccount`'s versions of these functions should invoke the base-class `Account` version to perform the updates to an account balance. `CheckingAccount`'s `debit` function should charge a fee only if money is actually withdrawn (i.e., the debit amount does not exceed the account balance). [*Hint:* Define `Account`'s `debit` function so that it returns a `bool` indicating whether money was withdrawn. Then use the return value to determine whether a fee should be charged.]

After defining the classes in this hierarchy, write a program that creates objects of each class and tests their member functions. Add interest to the `SavingsAccount` object by first invoking its `calculateInterest` function, then passing the returned interest amount to the object's `credit` function.

5 Short Questions:

a) Distinguish between virtual functions and pure virtual functions.

b) Distinguish between static binding and dynamic binding.

c) How is it that polymorphism enables you to program "in the general" rather than "in the specific" ? Discuss the key advantages of programming "in the general."