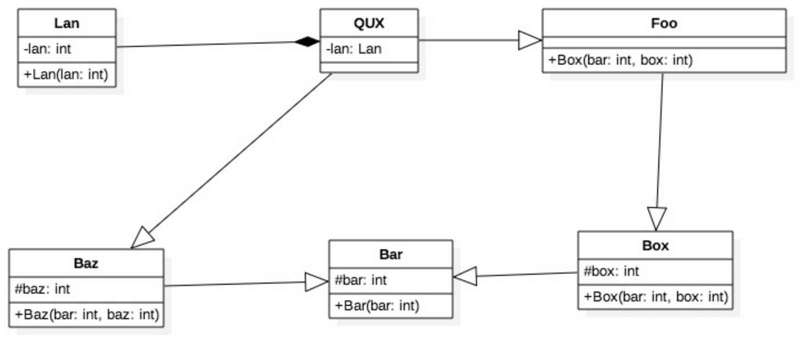
ENSF 480
Principles of Software Design
Fall 2021 – Quiz #2

Question 1 (5 points)

Consider the following class diagram:



Now, answer the following questions, in box below the question:

Question 1: Assuming that we need to use objects of these classes and we need to have access to all data member. Which classes need to have a virtual base class?

Question 2: write the implementation of constructor for class QUX.

Possible Answer for Question 1:

Answer must clearly states that:
Classes **Box** and **Baz** must have class **Bar** as their Virtual Base Class.

possible Answer for Question 2:

```
QUX::QUX(int a, int b, int c): Baz(a, b), Foo(a, b), Bar(a),lan(c) { }
```

Question 2 (7 points)

In this question there is a program with two classes: class `Human`, and class `Student`. Then, assume all member function of both classes are properly defined, except the assignment operator for class Student. In the following space write the definition of the assignment operator for class Student. Please notice that class Student is derived from class Human.

```
class Human {
protected:
    char *name;
public:
    ~Human() {delete [] name;}
    const char* get_name()const {return name;}
    Human(const char* nam="");
    Human(const Human& src);
    Human& operator= (const Human& rhs);
};
Human::Human(const char* nam): name(new char[strlen(nam)+1]) {
    assert(name!= nullptr);
    strcpy(this ->name, nam);
}
Human::Human(const Human& src){
    name = new char[strlen(src.name)+1];
    assert(name != nullptr);
    strcpy(this ->name, src.name);
}
Human& Human::operator= (const Human& rhs)
```

```
{
    if(this == &rhs) return *this;
    delete [] name;
    name = new char[strlen(rhs.name)+1];
    assert(name != nullptr);
    strcpy(this ->name, rhs.name);
    return *this;
}
class Student: public Human{
protected:
    char* phone_number;
public:
    Student(const char* student_name, const char* student_phone);
    ~Student() {delete [] phone_number;}
  //  Student (const Student& src);
    Student& operator= (const Student& rhs);
};
Student::Student(const char* student_name, const char* student_phone)
: Human(student_name), phone_number(new char[strlen (student_phone)+1])
{
    assert (phone_number!= nullptr);
    strcpy(phone_number, student_phone);
}
```

Answer for Question 2:

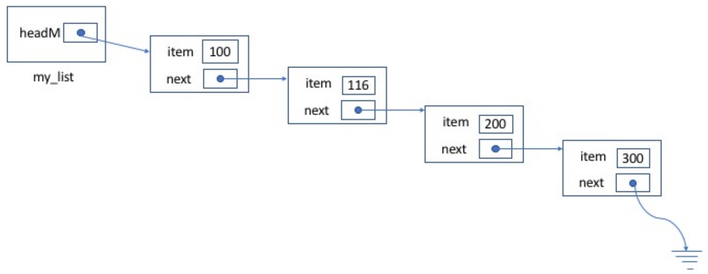
```
Student& Student::operator= (const Student& rhs){
         if(this == &rhs) return *this;
         Human::operator = (rhs); //this is an important statement - shouldn't be missing
         delete [] phone_number;
         phone_number = new char[strlen(rhs.phone_number)+1];
         assert(phone_number != nullptr); // assert is acceptable for quiz
         strcpy(this ->phone_number, rhs.phone_number);
         return *this;
 }
```

Question 3 (5 points)

This problem concerns a template class called *List* that maintains a list of objects of different types. Partial definition of class *List* and class *Node* in C++ are given. Don't worry about forward declarations if needed.

`template <class T>`
`struct Node {`
`  T item;`
`  Node <T> *next;`
`};`
	`template <class T>`
	`class List {`
	`public:`
	` List() {headM = nullptr;}`
	` void insert(T & itemA);`
	` friend ostream& operator<< <T>(ostream& os, const List<T>& x);`
	`private:`
	`    Node <T> *headM;`
	`};`

In the following space write the definition of `operator[]`, for class `List`. By using this operator you should be able to retrieve or modify the value of the item in the `i`th node of the list. For example, if `mylist` is an instance of `List <int>` with four nodes as follows,



you should be able to write the following statement to retrieve the value of the first node:

```
   int y  = mylist[0];
```
Means the value of y after the above statement must be 100. Or, to change the value of item in the third node from 200 to 1000, you should be able to write:
```
  mylist[2] = 1000;
```
Note: You don't need to worry about cases that the index number is out of bound (less than zero or passed the number nodes)

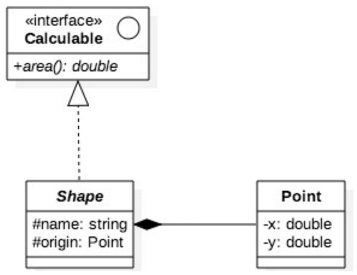
Answer for quesiton 3:

```
template <class T>
T& List<T>::operator [] (int index) {
     Node <T> *p = headM;
     assert (p!= nullptr);

     for (int i = 0; i < index; i++)
         p = p -> next;

     return p -> item;
}
```

Question 4 (5 points):



Consider the following UML diagram. Now, convert the diagram to C++ code. in the following space, ONLY write the definition of the classes.**Note:** you DON'T need to worry about inclusion of constructor, destructor, assignment operator, copy constructor, getters, setter, **Or implementation of any of the functions**. Only focus on what goes into a .h file.

Answer for Question 4:

```
// You need to keep the class in the right order Calculable and Point must be
// visible to Shape.

class Calculable{
public:
    virtual double area() = 0;
};
class Point {
private:
    double x, y;
};
class Shape : public Calculable {
protected:
    string name;
    Point origin;

};
```