OOP TEST 2

Question 1

1. Describe the difference between private, public, and protected in controlling the accessibility of object members in C++.
2. What is destructor in C++? When and why do we need to explicitly define the destructor for a class?
3. What is the difference between overloading and overriding?
4. Reuse what is already written is one of the main purposes of OOP, why it is easier to reuse in C++ than in C?

Question 2

Assume all necessary libraries are included, read the C++ code below and answer the following questions:

```
1  class Chef {
2  public:
3      virtual void prepare() = 0;
4      void makeDish() {
5          prepare();
6          cout << "Made by Chef\n";
7      }
8  };
9  class HeadChef: public Chef {
10 public:
11     virtual void prepare() {
12         cout << "Prepared by HeadChef\n";
13     }
14 };
15 class SecondChef : public HeadChef {
16 public:
17     void prepare() {
18         HeadChef::prepare();
19         cout << "Added by SecondChef\n";
20     }
21 };
22 void makeFood(HeadChef c1, SecondChef c2) {
23     c1.makeDish();
24     c2.makeDish();
25 }
26 void main() {
27     SecondChef c;
28     makeFood(c, c);
29
```

```
30      Chef *c1;
31      c1 = new Chef;
32      c1->prepare();
33
34      c1 = new HeadChef;
35      c1->prepare();
36
37      c1 = new SecondChef;
38      c1->prepare();
39
40      SecondChef *c2 = new HeadChef;
41      c2->prepare();
42  }
```

1. Are there any lines in the `main()` function that cannot be compiled? Why can't they be compiled?
2. Assume that all invalid lines of code are removed, what is printed to the screen after the line 28 is run?
3. Assume that all invalid lines of code are removed, what is printed to the screen after the line 42 is run?
4. At line 28, `makeFood` takes two `SecondChef` objects as arguments, explain why it can do that.

Question 3

Consider a basic mathematical expression which is a series of real numbers and arithmetic operations (`+, -, *, /`). A design option to represent this kind of expression is by using a tree.

There are two types of nodes in the expression tree:

- **Number node**: represents a number which has numerical value.
- **Operation node**: represents an operation which contains an operation symbol. Each symbol is either `+`, `-`, `*`, or `/`. Each operation node contains a left and a right node, which can either be number node or operation node.

Applying encapsulation, inheritance and polymorphism in object oriented programming, you are asked to do the following:

1. Draw a UML class diagram to show the tree representation above. The design should include necessary functions to construct an expression tree and evaluate the value of the expression.
2. Write C++ code to implement the design.