Answers to the questions

Question 1: Compilation Exercises

1. Will not compile. Return statement is missing.
2. boolean is not the correct way to declare (also initialize) a boolean variable. The correct way would be `bool x = true;`
3. Works fine, the x value will be automatically casted into a double.
4. The struct is ok, the problem is that in the same time it is a typedef, but the name of the new type is missing at the end of the statement. The code will compile but you get a warning that the typedef was ignored.
5. Ok, 0 in return statement is evaluated just fine. 0 is equal to false. 0 will be returned.

Question 2: Error Search

a)

Commented the errors directly in the respective files.

b)

Using the g++ compiler, you can first compile the object file for pyramid.cpp with: `g++ -c pyramid.cpp`. Than you can compile the programm (and link main.cpp with the new pyramid.o) with: `g++ main.cpp pyramid.o -o pyramid_prog`. If you want to link the math library, you need to add `-lm` compiling the programm, but since it wasn't needed we commented the include statement out, in order to save some memory.

Question 5: Enum, Struct and Union

c)

Structure is a data type that stores different data types in the same memory location; the total memory size of the structure is the summation of memory sizes of all its members. In contrast, Union is a data type that stores different data types in the same memory location; the total memory size depends on the memory size of its largest elements. A class is a way to group data and functionality for an object type. By default it is set to private, unlike a struct that is public, however you can add the keyword *public* to make something visible outside the class. Probably a class is more suitable in case there is a complex hierarchical structure, with inheritance.

Structure: Represent a complex number

```
struct Complex {
    int real;
    int img;
};
```

Union: Represent a character (save memory because a character will have either one attribute or the other)

```
struct Character {
    string name;
    bool isRobot;
    union {
        string personality;
        int firmwareVersion;
    };
};
```

Class: Represent a player

```
class Player {
    public:
        int x, y;
        int speed;

        void Move(int xa, int ya) {
            x += xa * speed;
            y += ya * speed;
        }
};
```