Virtual vs. pure virtual vs. non-virtual functions

Recall:

Virtual
- Use virtual functions in a base class when you want to allow derived classes to override the implementation of the function if needed. The function in the base class can have a default implementation.

Pure virtual
- Use pure virtual functions in an abstract base class when you want to define an interface that must be implemented by derived classes. These functions have no implementation in the base class and must be overridden by derived classes.

Non-virtual
- Use non-virtual functions when you want to provide a specific implementation of a function in the base class that should not be overridden by derived classes.

1. Given the following classes and functionalities, list which functions should be virtual, pure virtual, or non-virtual:
   - Base class: Shape
   - Child classes: Triangle, Circle
   - Common variables: color, name
   - Unique variables to Triangle: base, height
   - Unique variable to Circle: radius
   - Functionalities:
     - Print the name
     - Print shape info (including name, color, and other variables)
     - Calculate area (each shape has a different way of calculating area)

Exceptions:

2. What is an exception, and how does it differ from the error handling you have been using?

3. Why would you create your own exception beyond those provided for you?

4. How does the order of the exceptions in the catch block matter?
Templates:

5. Given the following outline of code, use the vector class from the STL to add a new integer, 10, to a vector of integers, v.

```cpp
#include<iostream>
#include<vector>

using namespace std;

int main(){

}
```

6. How would you create a templated class called `dynarray` and provide the same functionality as `push_back()` in the vector templated class from the STL?

```cpp
template<class element>
class dynarray {
    private:
        element *a;
        int size;
    public:
        void add(const element &item);
};

template<class element>
dynarray<element>::add(const element &item){

}
```

7. Now, write code/pseudocode that has the functionality of adding the element to the front in this class.