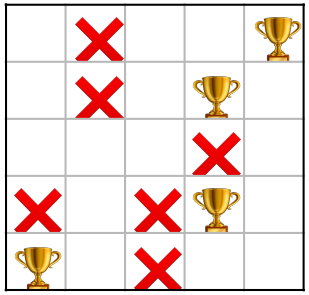
Singleton

1. Singleton is a _____ (creational/structural/ behavioral) design pattern.
2. Singleton in c++ depends mostly on _____ (programmer discipline/language constraints/both) if properly implemented.
3. It is important to make the constructor private because...
4. It is important to make the GetInstance method static because....
5. The instance variable needs to be static because....
6. It is important to delete the assignment operator and copy constructor because....

```
class Logger {
public:




private:



};
```

Flyweight (part 1)



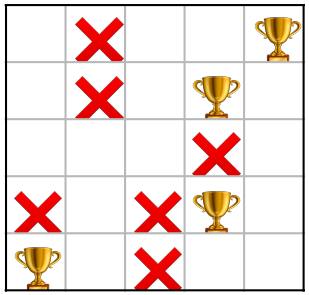
```
enum class SquareType {Empty, Wall, Treasure};

Graphics SquareTypeGraphics(SquareType sq) {
    if (sq == SquareType::Wall) {
        return Graphics(/* Wall parameters */);
    } else if (sq == SquareType::Treasure) {
        return Graphics(/* Treasure parameters */);
    } else {
        return Graphics(/* Empty parameters */);
    }
}
```

1. How many `SquareType` enums does it take to populate an n by n `Board` from the maze game?
2. If I want to display an n by n `Board`, how many `Graphics` objects get generated?
3. How much memory does the `Board` display take up if each `Graphics` object is 256 bytes?

1. Using pointers and only one instance of each of three re-designed `SquareType` objects, reduce the size in memory for the `Board` to be displayed. Your re-designed SquareType objects should include a corresponding `Graphics` object.

Draw a picture of what is happening with the Board

Write a new SquareType object definition



2. How much space in memory does your new Board display take up?

Flyweight (part 2)

1. Flyweight is a _____ (creational/structural/behavioral) design pattern.
2. Flyweight in c++ depends mostly on _____ (programmer discipline/language constraints/both) if properly implemented.

3. Flyweight is different than Singleton because...

4. To make an object that uses the Flyweight pattern in c++:

Iterator

```
std::vector<int> vec = {1, 3, 13, 27};

for (int number : vec) {
    std::cout << number << std::endl;
}
```

1. Write down an equivalent for loop to the one above for the given vector, accessing each element by index.

2. Write down an equivalent `while` loop to your `for` loop from #1.

3. Using the `std::vector::begin` and `std::vector::end` member functions, write down another equivalent for loop to the one that is given. We can increment iterators in c++ with the `++` operator.

4. Write down an equivalent `while` loop to your for loop from #3.

---

1. Iterator is a _____ (creational/structural/behavioral) design pattern.
2. The Iterator design pattern provides....

4. List three c++ containers that implement iterator: