Singleton

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

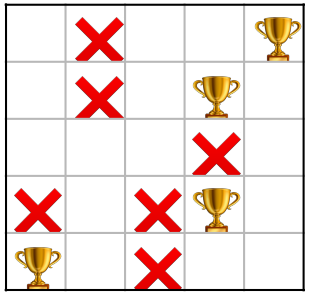
3. It is important to make the constructor private because...
   You rely on the fact there is only ever one instance of the object and you don't want to accidently create an unsynced copy
4. It is important to make the GetInstance method static because....
   You can't call it with the object

5. The instance variable needs to be static because....
   You can't modify non static fields in a static method

6. It is important to delete the assignment operator and copy constructor because.... They create instances of the object

```
class Logger {
public:
  static Logger& GetInstance(){
     static Logger instance;
     return instance;
  }

   Logger(Logger const&) = delete;
   void operator=(Logger const&) = delete;



private:
   Logger();



};
```

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 */);
    }
}
```

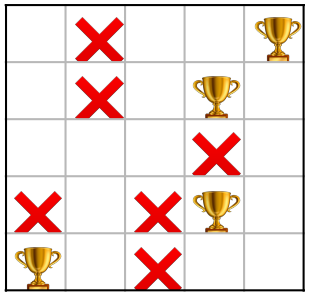
1. How many `SquareType` enums does it take to populate an n by n `Board` from the maze game?
   n*n

2. If I want to display an n by n `Board`, how many `Graphics` objects get generated?
   n*n

3. How much memory does the `Board` display take up if each `Graphics` object is 256 bytes?
   n*n*256

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

```
Graphics * graphWall = new Graphics(wall)
Graphics * graphTreasure = new Graphics
(Treasure)
Graphics * graphEmpty = new Graphics
(Empty)

Graphics SquareTypeGraphics
(SquareType sq){
   if(sq == SquareType::Wall){
      return(graphWall)
}
  if(sq == SquareType::Treasure){
      return(graphTreasure)
}
```

2. How much space in memory does your new Board display take up?
   3*256

Flyweight (part 2)

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

3. Flyweight is different than Singleton because...
   You can still create copies of the object

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.

```
for(int i = 0; i < vec.size(); i++ ){
  std::cout << vec[i] << std::endl;
}
```

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

```
int iter = 0;
while(iter < vec.size()){
  std::cout << vec[iter] << std::endl;
  i++;
}
```

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.

```
for(std::vector<int>::iterator i = vec.begin(); i < vec.end(); i++){
  std::cout<<*i <<std::endl;
}
```

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

```
(std::vector<int>::iterator i = vec.begin()
while(i!=vec.end()){
   std::cout << *i << std::endl;
   i++;
}
```

---

1. Iterator is a behavioral (creational/structural/behavioral) design pattern.
2. The Iterator design pattern provides....
   built in way to traverse data structure

4. List three c++ containers that implement iterator:
   vector, string, hashmap