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...

   Making the constructor private allows only one instance of the object to be created good for static variables

4. It is important to make the GetInstance method static because....

   Static will not give you a singleton, this ensures that there is only one instance of that class object allows for reference to object

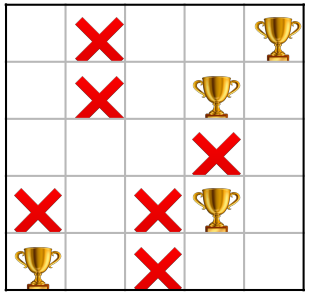
5. The instance variable needs to be static because....

   This ensures that there is only one version of the instance variable and that one instantiated, cannot be altered

6. It is important to delete the assignment operator and copy constructor because.... The singleton once instantiated should not have alterations

```
class Logger {
public:

        GetInstance()









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?

   nxn square types needed

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

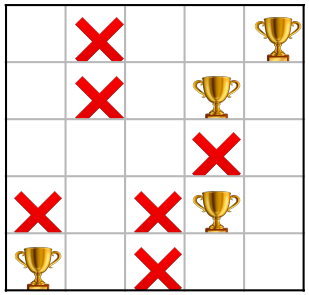
   There are n^2 instances needed to populate the nXn game

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

   6400 bytes of memory needed 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

Create pointers to each squaretype

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

   768 bytes of memory needed

Flyweight (part 2)

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

3. Flyweight is different than Singleton because...

   Sigleton is only one object vs
   flighweight utilizes several

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

   You must create
   several instances by
   referencing/pointing

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.

   while(I<vec.size()){
   std::cout<<vec[I]<<std::endl;

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.

   td::vector<int> iterator::: it
   for(it = vec.begin(); it != vec.end(); it++){
   std::cout<< it <<std::endl;

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

   std::vector<int> iterator::: it = vec.begin
   while(it != vec.end())
   std::cout<<it<<std::endl;
   it++

---

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

   We can use this for any variable type

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

   map
   vector
   strings