- Exercises

Let's solve a couple of exercises to test your understanding of variadic templates.

WE'LL COVER THE FOLLOWING

- Try It Out!
- Problem Statement 2

Try It Out!

The function `createT` in the Example 2 is the perfect factory function because it can invoke each constructor. Try to find a constructor call, which cannot be used in the function template `createT`.

```
#include <iostream>
#include <utility>

template<typename T, typename ... Args>
T createT(Args&& ... args){
  return T(std::forward<Args>(args) ...);
}

// Try to break the code

struct MyStruct{
  MyStruct(int&, double&, double&&){}
  friend std::ostream& operator<< (std::ostream& out, const MyStruct&){
      out << "MyStruct" << std::endl;
      return out;
  }
};

int main(){

  // Call it here after breaking the code

  std::cout << std::endl;
}
```

Problem Statement 2

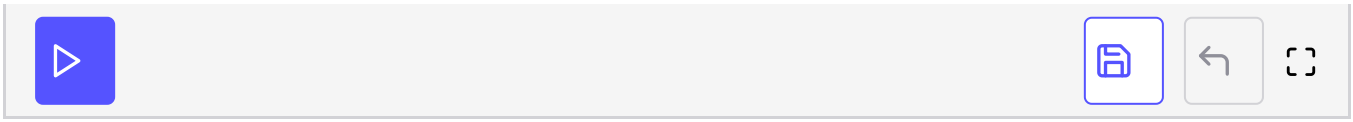
A typical example of a variadic template is the C function `printf`. Try to implement it with variadic templates.

```
#include <iostream>

void myPrintf(const char* format) {
    std::cout << format;
}

template<typename T, typename... Args>
void myPrintf(const char* format, T value, Args... args){
    // Implement the function here
}

int main(){
  // the given line should print "Hello world! 2011"
  myPrintf("% world% %\n","Hello",'!',2011);
}
```



In the next lesson, we'll look at the solution of problem statement 2.