- Solution

This lesson contains the solution to the exercise in the previous lesson.

Solution

```
#include <functional>
#include <string>
#include <iostream>

std::function<std::string()> makeLambda() {
  const std::string val = "on stack created";
  return [val]{return val;};
}

int main(){

  std::cout << std::endl;

  auto bad = makeLambda();
  std::cout << bad() << std::endl;

  std::cout << std::endl;

}
```

Explanation

- The trick is to bind `val` to the lambda as a copy.
- Initially, we were binding `val` as a reference. When the function ends, `val` is destroyed and its reference is undefined. This will result in undefined behavior.

- It could work, cause and error, or return some garbage value.
- Returning a copy of it ensures that the value of `val` is preserved even when the actual variable is freed from memory.

Next, we’ll tackle **classes and objects** in C++.