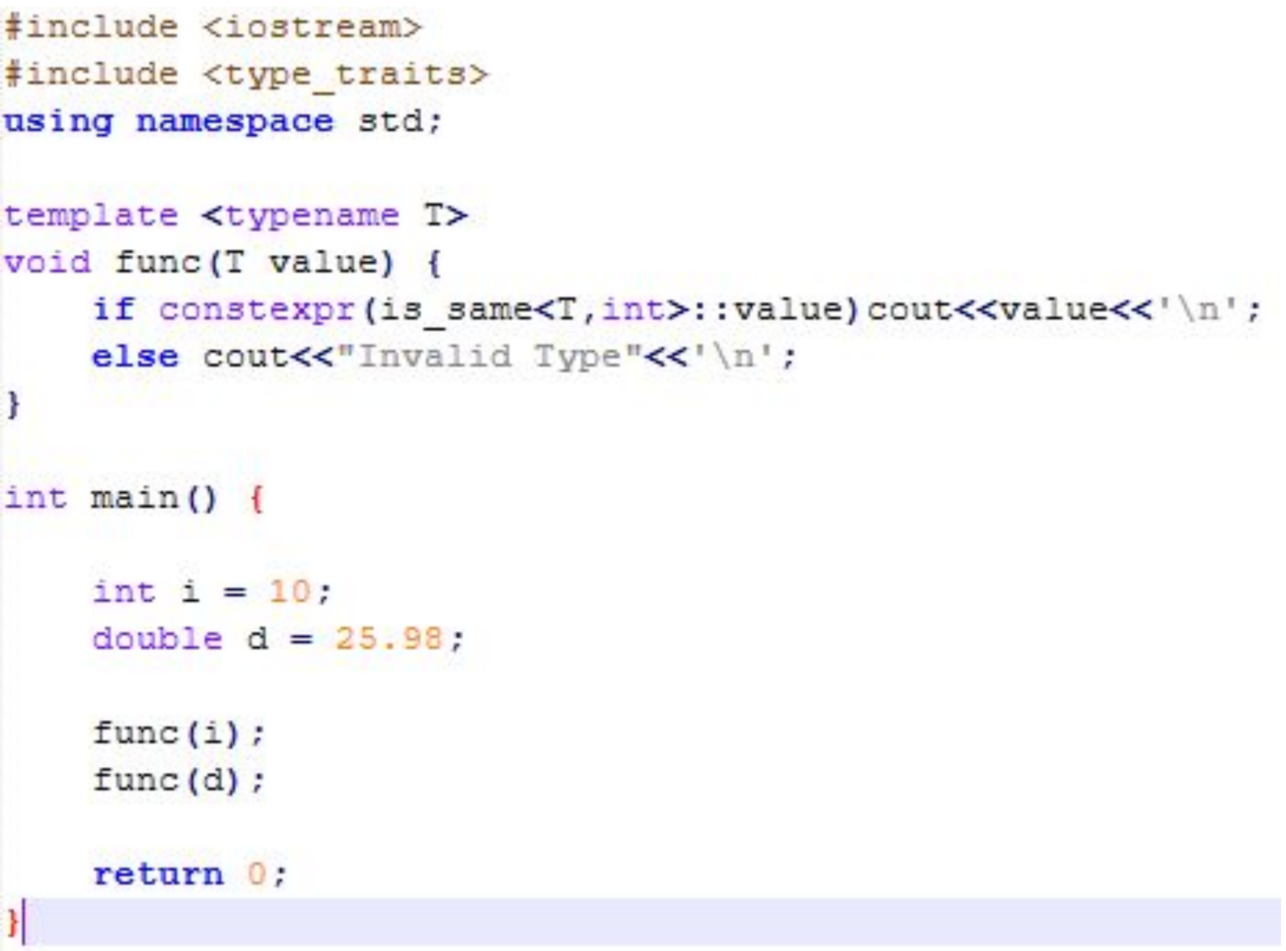
SWE 4504

Memory and Type Safety

Type-Safe Alternative Using Templates



```cpp
#include <iostream>
#include <type_traits>
using namespace std;

template <typename T>
void func(T value) {
    if constexpr(is_same<T,int>::value)cout<<value<<'\n';
    else cout<<"Invalid Type"<<'\n';
}

int main() {

    int i = 10;
    double d = 25.98;

    func(i);
    func(d);

    return 0;
}
```

- *Constexpr* is a keyword specifies that the value of a variable or the result of a function can be evaluated at compile-time.
- is_same: This is a type trait (from <type_traits> library) that checks if two types are the same. It evaluates to true if the types are identical, and false otherwise.

Tasks

1) memorySafety.c has multiple memory safety issues. Rewrite the code to solve them.
2) typeSafety.cpp has some type safety issues issues. Rewrite the code to solve them. Using template functions format in the previous slide is mandatory.

There can be multiple ways to solve a single issue. And the correct output doesn’t mean you will get full marks. Marks will be given on how you write your code.