Cpp exception handling

The task is to extend the existing code so that it handles `std::invalid_argument` exception properly.

More specifically, you have to extend the implementation of `process_input` function. It takes integer n as an argument and has to work as follows:

1. It calls function `largest_proper_divisor(n)`.
2. If this call returns a value without raising an exception, it should print in a single line `result=d` where d is the returned value.
3. Otherwise, if the call raises a `std::invalid_argument` exception, it has to print in a single line the string representation of the raised exception, i.e. its message.
4. Finally, no matter if the exception is raised or not, it should print in a single line `returning control flow to caller` after any other previously printed output.

To keep the code quality high, you are advised to have exactly one line printing `returning control flow to caller` in the body of `process_input` function.

Your function will be tested against several cases by the locked template code.

Input Format

The input is read by the provided locked code template. In the only line of the input, there is a single integer n, which is going to be the argument passed to function `process_input`.

Constraints

- $0 \leq n \leq 100$

Output Format

The output should be produced by function `process_input` as described in the statement.

Sample Input 0

```
0
```

Sample Output 0

```
largest proper divisor is not defined for n=0
returning control flow to caller
```

Explanation 0

In the first sample, $n = 0$, so the call `largest_proper_divisor(0)` raises an exception. In this case, the function `process_input` prints two lines. In the first of them it prints the string representation of the raised exception, and in the second line it prints `returning control flow to caller`.

Sample Input 1

```
9
```

Sample Output 1

```
result=3
```

```
returning control flow to caller
```

Explanation 1

In the first sample, $n = 9$, so the call `largest_proper_divisor(9)` doesn't raise an exception and returns value 3. In this case, the function `process_input` prints two lines. In the first of them it prints `result=3` because the returned value by `largest_proper_divisor(9)` is 3, and in the second line it prints `returning control flow to caller`.