Conditional Statements in C

Objective

if and *else* are two of the most frequently used conditionals in C/C++, and they enable you to execute zero or one conditional statement among many such dependent conditional statements. We use them in the following ways:

1. if : This executes the body of bracketed code starting with $statement1$ if $condition$ evaluates to *true*.

    ```
    if (condition) {
        statement1;
        ...
    }
    ```

2. if - else : This executes the body of bracketed code starting with $statement1$ if $condition$ evaluates to *true*, or it executes the body of code starting with $statement2$ if $condition$ evaluates to *false*. Note that only *one* of the bracketed code sections will ever be executed.

    ```
    if (condition) {
        statement1;
        ...
    }
    else {
        statement2;
        ...
    }
    ```

3. if - else if - else : In this structure, dependent statements are chained together and the $condition$ for each statement is only checked if all prior conditions in the chain are evaluated to *false*. Once a $condition$ evaluates to *true*, the bracketed code associated with that statement is executed and the program then skips to the end of the chain of statements and continues executing. If each $condition$ in the chain evaluates to false, then the body of bracketed code in the *else* block at the end is executed.

    ```
    if(first condition) {
        ...
    }
    else if(second condition) {
        ...
    }
    .
    .
    .
    else if((n-1)'th condition) {
        ....
    }
    else {
        ...
    }
    ```

Task

Given a positive integer denoting n, do the following:

- If $1 \leq n \leq 9$, then print the lowercase English word corresponding to the number (e.g., one for 1, two for 2, etc.).

- If $n > 9$, print Greater than 9 .

Input Format

The first line contains a single integer denoting n.

Constraints

- $1 \leq n \leq 10^9$

Output Format

If $1 \leq n \leq 9$, then print the lowercase English word corresponding to the number (e.g., `one` for 1, `two` for 2, etc.); otherwise, print `Greater than 9` instead.

Sample Input

```
5
```

Sample Output

```
five
```

Sample Input #01

```
8
```

Sample Output #01

```
eight
```

Sample Input #02

```
44
```

Sample Output #02

```
Greater than 9
```