Problem 1

Function Signature:

```
def print_number_pattern(n: int) -> None:
```

Input:

- An integer `n` (1 <= n <= 100) — the number of integers in the first row.

Output:

- Print the pattern as described.
- Each number should be separated by a space.
- No trailing spaces at the end of any line.

Examples:

Input:

```
n = 3
```

Output:

```
1 2 3
1 2
1
```

Input:

```
n = 6
```

Output:

```
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Constraints:

- 1 <= n <= 100

Problem 2

Function Signature:

```
def single_number(nums: list[int]) -> int:
```

Input:

- A list of integers `nums` where every element appears exactly twice except for one element which appears only once.

Output:

- Return the element that appears only once.

Examples:

Input:

```
nums = [2, 2, 1]
```

Output:

```
1
```

Input:

```
nums = [4, 1, 2, 1, 2]
```

Output:

```
4
```

Input:

```
nums = [1]
```

Output:

```
1
```

Constraints:

- `1 <= nums.length <= 3 * 10^4`
- `-3 * 10^4 <= nums[i] <= 3 * 10^4`
- Each element in the array appears exactly twice except for one element which appears only once.

Problem 3

Function Signature:

```
def length_of_longest_substring(s: str) -> int:
```

Input:

- You are given a string s, consisting only of lowercase English letters. Your task is to determine the length of the longest substring that contains no repeated characters.
- A substring is a contiguous sequence of characters within a string. Note that a subsequence (which may skip characters) is not allowed.

Output:

- Return the length of the longest substring of `s` that contains all unique characters.

Examples:

Input:

```
s = "abcabcbb"
```

Output:

```
3
```

Input:

```
s = "bbbbb"
```

Output:

```
1
```

Input:

```
s = "pwwkew"
```

Output:

```
3
```

Constraints:

- $0 \leq len(s) \leq 5 * 10^4$
- `s` consists of only lowercase english characters.