1. Two Sum

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to `target`*.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

Example 1:

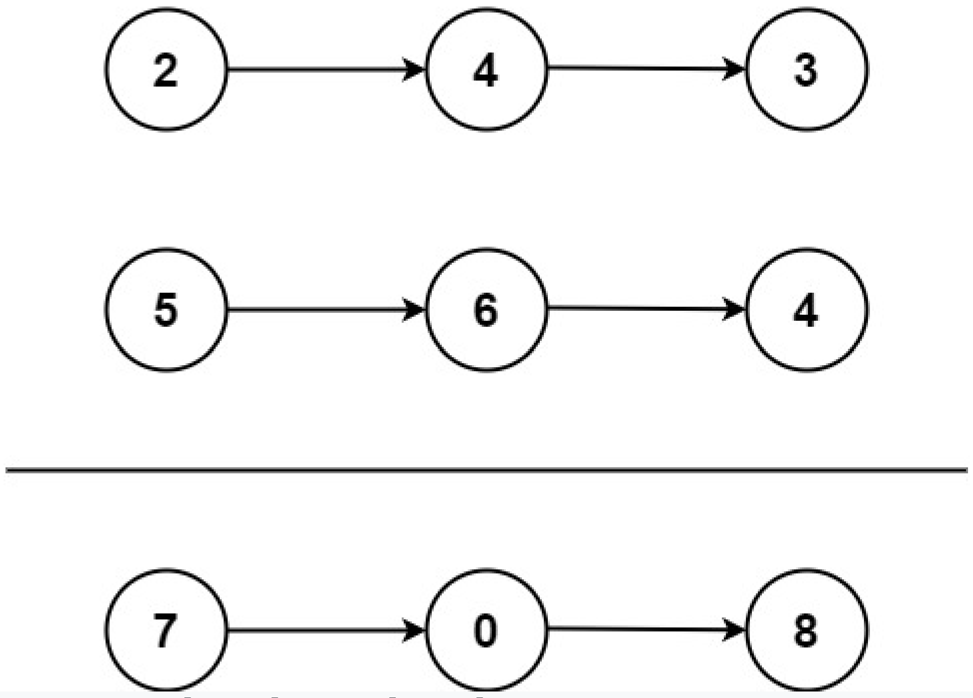
```
Input: nums = [2,7,11,15], target = 9

Output: [0,1]

Output: Because nums[0] + nums[1] == 9, we return [0, 1].
```

2. Add Two Numbers

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example 1:



```
Input: l1 = [2,4,3], l2 = [5,6,4]

Output: [7,0,8]

Explanation: 342 + 465 = 807.
```

3. Longest Substring Without Repeating Characters

Given a string `s`, find the length of the **longest substring** without repeating characters

Example 1:

```
Input: s = "abcabcbb"

Output: 3

Explanation: The answer is "abc", with the length of 3.
```

4. Median of Two Sorted Arrays

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return **the median** of the two sorted arrays.

The overall run time complexity should be `O(log (m+n))`.

Example 1:

```
Input: nums1 = [1,3], nums2 = [2]

Output: 2.00000

Explanation: merged array = [1,2,3] and median is 2
```

5. Longest Palindromic Substring

Given a string `s`, return *the longest palindromic substring* in `s`.

Example 1:

```
Input: s = "babad"

Output: "bab"

Note: "aba" is also a valid answer.
```

6. ZigZag Conversion

The string `"PAYPALISHIRING"` is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)

```
P   A   H   N

A P L S I I G

Y   I   R
```

And then read line by line: `"PAHNAPLSIIGYIR"`

Write the code that will take a string and make this conversion given a number of rows:

```
string convert(string s, int numRows);
```

Example 1:

```
Input: s = "PAYPALISHIRING", numRows = 3

Output: "PAHNAPLSIIGYIR"
```

7. Reverse Integer

Given a signed 32-bit integer `x`, return `x` *with its digits reversed*. If reversing `x` causes the value to go outside the signed 32-bit integer range $[-2^{31}, 2^{31} - 1]$, then return `0`.

Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1:

```
Input: x = 123

Output: 321
```

8. String to Integer (atoi)

Implement the `myAtoi(string s)` function, which converts a string to a 32-bit signed integer (similar to C/C++'s `atoi` function).

The algorithm for `myAtoi(string s)` is as follows:

1. Read in and ignore any leading whitespace.
2. Check if the next character (if not already at the end of the string) is `'-'` or `'+'`. Read this character in if it is either. This determines if the final result is negative or positive respectively. Assume the result is positive if neither is present.
3. Read in next the characters until the next non-digit charcter or the end of the input is reached. The rest of the string is ignored.
4. Convert these digits into an integer (i.e. `"123" -> 123`, `"0032" -> 32`). If no digits were read, then the integer is `0`. Change the sign as necessary (from step 2).
5. If the integer is out of the 32-bit signed integer range $[-2^{31}, 2^{31} - 1]$, then clamp the integer so that it remains in the range. Specifically, integers less than -2^{31} should be clamped to -2^{31}, and integers greater than $2^{31} - 1$ should be clamped to $2^{31} - 1$.
6. Return the integer as the final result.

Note:

- Only the space character `' '` is considered a whitespace character.
- **Do not ignore** any characters other than the leading whitespace or the rest of the string after the digits.

Example 1:

```
Input: s = "42"

Output: 42

Explanation: The underlined characters are what is read in, the caret is the
current reader position.
```

```
Step 1: "42" (no characters read because there is no leading whitespace)

Step 2: "42" (no characters read because there is neither a '-' nor '+')

Step 3: "42" ("42" is read in)

The parsed integer is 42.

Since 42 is in the range [-2^31, 2^31 - 1], the final result is 42.
```

9. Palindrome Number

Given an integer `x`, return `true` if `x` is palindrome integer.

An integer is a **palindrome** when it reads the same backward as forward. For example, `121` is palindrome while `123` is not.

Example 1:

```
Input: x = 121

Output: true
```

10. Regular Expression Matching

Given an input string `s` and a pattern `p`, implement regular expression matching with support for `'.'` and `'*'` where:

- `'.'` Matches any single character.
- `'*'` Matches zero or more of the preceding element.

The matching should cover the **entire** input string (not partial).

Example 1:

```
Input: s = "aa", p = "a"

Output: false

Explanation: "a" does not match the entire string "aa".
```