1. Two sum

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int, int> m;
        for (int i = 0;; ++i) {
            int x = nums[i];
            int y = target - x;
            if (m.count(y)) {
                return {m[y], i};
            }
            m[x] = i;
        }
    }
};
```

2.Add two numbers

```
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        ListNode* dummy = new ListNode();
        int carry = 0;
        ListNode* cur = dummy;
        while (l1 || l2 || carry) {
            int s = (l1 ? l1->val : 0) + (l2 ? l2->val : 0) + carry;
            carry = s / 10;
            cur->next = new ListNode(s % 10);
            cur = cur->next;
            l1 = l1 ? l1->next : nullptr;
            l2 = l2 ? l2->next : nullptr;
        }
        return dummy->next;
    }
};
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        ListNode* dummy = new ListNode();
        int carry = 0;
        ListNode* cur = dummy;
        while (l1 || l2 || carry) {
            int s = (l1 ? l1->val : 0) + (l2 ? l2->val : 0) + carry;
            carry = s / 10;
            cur->next = new ListNode(s % 10);
            cur = cur->next;
            l1 = l1 ? l1->next : nullptr;
            l2 = l2 ? l2->next : nullptr;
        }
        return dummy->next;
    }
};
```

3.Longest Substring Without Repeating Characters

```
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        bool ss[128]{};
        int ans = 0;
        for (int i = 0, j = 0; j < s.size(); ++j) {
            while (ss[s[j]]) {
```

```
            ss[s[i++]] = false;
        }
        ss[s[j]] = true;
        ans = max(ans, j - i + 1);
    }
    return ans;
  }
};
```

4. Median of Two Sorted Arrays

```
class Solution {
public:
  double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
    int m = nums1.size(), n = nums2.size();
    function<int(int, int, int)> f = [&](int i, int j, int k) {
        if (i >= m) {
            return nums2[j + k - 1];
        }
        if (j >= n) {
            return nums1[i + k - 1];
        }
        if (k == 1) {
            return min(nums1[i], nums2[j]);
        }
        int p = k / 2;
        int x = i + p - 1 < m ? nums1[i + p - 1] : 1 << 30;
        int y = j + p - 1 < n ? nums2[j + p - 1] : 1 << 30;
        return x < y ? f(i + p, j, k - p) : f(i, j + p, k - p);
    };
```

```
        int a = f(0, 0, (m + n + 1) / 2);
        int b = f(0, 0, (m + n + 2) / 2);
        return (a + b) / 2.0;
    }
};
```

5. Longest Palindromic Substring

```
class Solution {
public:
    string longestPalindrome(string s) {
        int n = s.size();
        vector<vector<bool>> f(n, vector<bool>(n, true));
        int k = 0, mx = 1;
        for (int i = n - 2; ~i; --i) {
            for (int j = i + 1; j < n; ++j) {
                f[i][j] = false;
                if (s[i] == s[j]) {
                    f[i][j] = f[i + 1][j - 1];
                    if (f[i][j] && mx < j - i + 1) {
                        mx = j - i + 1;
                        k = i;
                    }
                }
            }
        }
        return s.substr(k, mx);
    }
};
```

6. Zigzag Conversion

```
class Solution {
public:
    string convert(string s, int numRows) {
        if (numRows == 1) {
            return s;
        }
        vector<string> g(numRows);
        int i = 0, k = -1;
        for (char c : s) {
            g[i] += c;
            if (i == 0 || i == numRows - 1) {
                k = -k;
            }
            i += k;
        }
        string ans;
        for (auto& t : g) {
            ans += t;
        }
        return ans;
    }
};
```

7. Reverse Integer

```
class Solution {
public:
    int reverse(int x) {
        int ans = 0;
        for (; x; x /= 10) {
            if (ans < INT_MIN / 10 || ans > INT_MAX / 10) {
                return 0;
            }
            ans = ans * 10 + x % 10;
        }
        return ans;
    }
};
```

8. String to Integer

```
class Solution {
 public:
  int myAtoi(string s) {
    trim(s);
    if (s.empty())
      return 0;

    const int sign = s[0] == '-' ? -1 : 1;
    if (s[0] == '+' || s[0] == '-')
      s = s.substr(1);

    long num = 0;

    for (const char c : s) {
      if (!isdigit(c))
        break;
      num = num * 10 + (c - '0');
      if (sign * num < INT_MIN)
        return INT_MIN;
      if (sign * num > INT_MAX)
        return INT_MAX;
    }

    return sign * num;
  }
 private:
  void trim(string& s) {
    s.erase(0, s.find_first_not_of(' '));
    s.erase(s.find_last_not_of(' ') + 1);
  }
};
```

9. Palindrome Number

```
class Solution {
public:
    bool isPalindrome(int x) {
        if (x < 0 || (x && x % 10 == 0)) {
            return false;
        }
        int y = 0;
        for (; y < x; x /= 10) {
            y = y * 10 + x % 10;
        }
        return x == y || x == y / 10;
    }
};
```

10. Regular Expression Matching

```
class Solution {
public:
    bool isMatch(string s, string p) {
        int m = s.size(), n = p.size();
        int f[m + 1][n + 1];
        memset(f, 0, sizeof f);
        function<bool(int, int)> dfs = [&](int i, int j) -> bool {
            if (j >= n) {
                return i == m;
            }
            if (f[i][j]) {
                return f[i][j] == 1;
            }
            int res = -1;
            if (j + 1 < n && p[j + 1] == '*') {
                if (dfs(i, j + 2) or (i < m and (s[i] == p[j] or p[j] == '.') and dfs(i + 1, j))) {
                    res = 1;
                }
            } else if (i < m and (s[i] == p[j] or p[j] == '.') and dfs(i + 1, j + 1)) {
                res = 1;
            }
            f[i][j] = res;
            return res == 1;
        };
        return dfs(0, 0);
    }
};
```