### 字符串操作

#### M3. Longest Substring Without Repeating Characters

|  |
| --- |
| **M3. Longest Substring Without Repeating Characters** |
| Given a string, find the length of the longest substring without repeating characters.  **Examples:**  Given "abcabcbb", the answer is "abc", which the length is 3.  Given "bbbbb", the answer is "b", with the length of 1.  Given "pwwkew", the answer is "wke", with the length of 3.  Note that the answer must be a substring, "pwke" is a subsequence and not a substring.  **Tags：**Hash Table、Two Pointers、String  **Similar Problems：** (H) Longest Substring with At Most Two Distinct Characters |
| int lengthOfLongestSubstring(string s); |

#### M5. Longest Palindromic Substring

|  |
| --- |
| **M5. Longest Palindromic Substring** |
| Given a string s, find the longest palindromic substring in s. You may assume that the maximum length of s is 1000.  **Example:**  Input: "babad"  Output: "bab"  Note: "aba" is also a valid answer.  **Example:**  Input: "cbbd"  Output: "bb"  **Tags：**String  **Similar Problems：**(H) Shortest Palindrome、(E) Palindrome Permutation、 (H) Palindrome Pairs、 (M) Longest Palindromic Subsequence |
| string longestPalindrome(string s); |

#### M8. String to Integer (atoi)

|  |
| --- |
| **M8. String to Integer (atoi)** |
| Implement atoi to convert a string to an integer.  **Hint:** Carefully consider all possible input cases. If you want a challenge, please do not see below and ask yourself what are the possible input cases.  **Notes:** It is intended for this problem to be specified vaguely (ie, no given input specs). You are responsible to gather all the input requirements up front.  **Update (2015-02-10):**  The signature of the C++ function had been updated. If you still see your function signature accepts a const char \* argument, please click the reload button to reset your code definition.  **Tags：**Math、String  **Similar Problems**：(E) Reverse Integer、(H) Valid Number |
| int myAtoi(string str); |

#### M12. Integer to Roman

|  |
| --- |
| **M12. Integer to Roman** |
| Given an integer, convert it to a roman numeral.  Input is guaranteed to be within the range from 1 to 3999.  **Tags：**Math、String  **Similar Problems：** (E) Roman to Integer 、(H) Integer to English Words |
| string intToRoman(int num); |

#### M43. Multiply Strings

|  |
| --- |
| **M43. Multiply Strings** |
| Given two non-negative integers num1 and num2 represented as strings, return the product of num1 and num2.  **Note:**   * The length of both num1 and num2 is < 110. * Both num1 and num2 contains only digits 0-9. * Both num1 and num2 does not contain any leading zero. * You must not use any built-in BigInteger library or convert the inputs to integer directly.   **Tags：**Math String  **Similar Problems：**(M) Add Two Numbers、(E) Plus One、(E) Add Binary、(E) Add Strings |
| string multiply(string num1, string num2); |

#### M49. Group Anagrams

|  |
| --- |
| **M49. Group Anagrams** |
| Given an array of strings, group anagrams together.  **For example,** given: ["eat", "tea", "tan", "ate", "nat", "bat"],  **Return:**  [  ["ate", "eat","tea"],  ["nat","tan"],  ["bat"]  ]  **Note:** All inputs will be in lower-case.  **Tags：**Hash Table、String  **Similar Problems：**(E) Valid Anagram、(M) Group Shifted Strings |
| vector<vector<string>> groupAnagrams(vector<string>& strs); |

#### M91. Decode Ways

|  |
| --- |
| **M91. Decode Ways** |
| A message containing letters from A-Z is being encoded to numbers using the following mapping:  'A' -> 1  'B' -> 2  ...  'Z' -> 26  Given an encoded message containing digits, determine the total number of ways to decode it.  **For example,**  Given encoded message "12", it could be decoded as "AB" (1 2) or "L" (12).  The number of ways decoding "12" is 2.  **Tags：**Dynamic Programming、String |
| int numDecodings(string s); |

#### M93. Restore IP Addresses

|  |
| --- |
| **M93. Restore IP Addresses** |
| Given a string containing only digits, restore it by returning all possible valid IP address combinations.  **For example:**  Given "25525511135",  return ["255.255.11.135", "255.255.111.35"]. (Order does not matter)  **Tags：**Backtracking、String |
| vector<string> restoreIpAddresses(string s); |

#### M151. Reverse Words in a String

|  |
| --- |
| **M** **151. Reverse Words in a String** |
| Given an input string, reverse the string word by word.  **For example,**  Given s = "the sky is blue",  return "blue is sky the".  Update (2015-02-12):  For C programmers: Try to solve it in-place in O(1) space.  **Clarification:**   * What constitutes a word?   A sequence of non-space characters constitutes a word.   * Could the input string contain leading or trailing spaces?   Yes. However, your reversed string should not contain leading or trailing spaces.   * How about multiple spaces between two words?   Reduce them to a single space in the reversed string.  Tags：String  Similar Problems：(M) Reverse Words in a String II |

#### M385. Mini Parser

|  |
| --- |
| **M** **385. Mini Parser** |
| Given a nested list of integers represented as a string, implement a parser to deserialize it.  Each element is either an integer, or a list -- whose elements may also be integers or other lists.  **Note:** You may assume that the string is well-formed:   * String is non-empty. * String does not contain white spaces. * String contains only digits 0-9, [, - ,, ].   **Example 1:**  Given s = "324",  You should return a NestedInteger object which contains a single integer 324.  **Example 2:**  Given s = "[123,[456,[789]]]",  Return a NestedInteger object containing a nested list with 2 elements:  1. An integer containing value 123.  2. A nested list containing two elements:  i. An integer containing value 456.  ii. A nested list with one element:  a. An integer containing value 789.  **Tags：**Stack String  **Similar Problems：**(M) Flatten Nested List Iterator (M) Ternary Expression Parser |
| NestedInteger deserialize(string s)； |

#### M

|  |
| --- |
| **M** |
| The |

#### M

|  |
| --- |
| **M** |
| The |

#### M

|  |
| --- |
| **M** |
| The |

#### M

|  |
| --- |
| **M** |
| The |

#### M

|  |
| --- |
| **M** |
| The |

#### M

|  |
| --- |
| **M** |
| The |

#### M

|  |
| --- |
| **M** |
| The |