Algotutor Assignment Solutions

1.Roman to Integer

Sol:

int getValue(const char \* s){

    switch(\*s) {

        case 'I': return (s[1] == 'V' || s[1] == 'X') ? -1 : 1;

        case 'X': return (s[1] == 'L' || s[1] == 'C') ? -10 : 10;

        case 'C': return (s[1] == 'D' || s[1] == 'M') ? -100 : 100;

        case 'V': return 5;

        case 'L': return 50;

        case 'D': return 500;

        case 'M': return 1000;

    }

    return 0;

}

int romanToInt(char \* s){

    int result = 0;

    for(;\*s != '\0'; ++s) {

        result += getValue(s);

    }

    return result;

}

2.Longest Common Prefix

Sol:

class Solution {

    public String longestCommonPrefix(String[] strs) {

        if(strs.length == 0){

            return "";

        }

        String prefix = strs[0];

        for(String s : strs){

            int len = Math.min(prefix.length(), s.length());

            for(int i = len; i >= 0; i--){

                if(prefix.indexOf(s.substring(0, i)) == 0){

                    prefix = s.substring(0, i);

                    break;

                }

            }

        }

        return prefix;

    }

}

3.Valid Palindrome

Sol:

bool isPalindrome(char \* s){

    if (s == NULL || !strcmp(s,"")) {

        return true;

        }

    int i, j, k = 0;

    int count = 0;

    int len = strlen(s);

    char buffer[len];

    for (i = 0; i < len; i++) {

        if(isalnum(s[i])) {

            buffer[k] = tolower(s[i]);

            k++;

        }

    }

    for (i = 0; i < k; i++) {

        j = k - 1 - i;

        if ((buffer[i] != buffer[j]) || (i >= j)) {

            break;

        }

        count++;

    }

    if (count == ceil((0 + k)/2)) {

         return true;

        }

    return false;

}

4.Valid Anagram

Sol:

class Solution {

    public boolean isAnagram(String s, String t) {

        if (s.length() != t.length())

      return false;

    int[] count = new int[128];

    for (final char ch : s.toCharArray())

      ++count[ch];

    for (final char ch : t.toCharArray())

      if (--count[ch] < 0)

        return false;

    return true;

    }

}

5.Reverse words in a String

Sol:

class Solution {

    public String reverseWords(String s) {

        if (s == null || s.length() == 0) {

      return "";

    }

    String[] arr = s.split(" ");

    StringBuilder sb = new StringBuilder();

    for (int i = arr.length - 1; i >= 0; --i) {

      if (!arr[i].equals("")) {

        sb.append(arr[i]).append(" ");

      }

    }

    return sb.length() == 0 ? "" : sb.substring(0, sb.length() - 1);

    }

}

6.Count and Say

Sol:

class Solution {

    public String countAndSay(int n) {

        StringBuilder sb = new StringBuilder("1");

    while (--n > 0) {

      StringBuilder next = new StringBuilder();

      for (int i = 0; i < sb.length(); ++i) {

        int ctr = 1;

        while (i + 1 < sb.length() && sb.charAt(i) == sb.charAt(i + 1)) {

          ++ctr;

          ++i;

        }

        next.append(ctr).append(sb.charAt(i));

      }

      sb = next;

    }

    return sb.toString();

    }

}

7.Longest Valid Parentheses

Sol:

class Solution {

    public int longestValidParentheses(String s) {

        int result = 0;

        Stack<Integer> stack = new Stack<>();

        stack.push(-1);

        for (int i=0; i<s.length(); i++) {

            if (s.charAt(i) == '(') {

                stack.push(i);

            }

            else {

                stack.pop();

                if (stack.isEmpty())

                    stack.push(i);

                else

                    result = Math.max(result, i - stack.peek());

            }

        }

        return result;

    }

}