C++ Assignments | Strings - 2 | Week 7

1.Input a string and concatenate with its reverse string and print it.

Input : str = "PWSkills"
Output : "PWSkillssllikSWP"
Input : str = "pw"
Output : “pwwp”

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

string concatenateWithReverse(const string& str) {
    string revStr = str;
    reverse(revStr.begin(), revStr.end());
    return str + revStr;
}

int main() {
    string str1 = "PWSkills";
    string str2 = "pw";

    cout << "Concatenated string 1: " << concatenateWithReverse(str1) << endl;
    cout << "Concatenated string 2: " << concatenateWithReverse(str2) << endl;

    return 0;
}
```

2.Find the second largest digit in the string consisting of digits from ‘0’ to ‘9’.

Input : str = "2947578"
Output : 8
Input : str = "1241"
Output : 2

```
#include <iostream>
#include <string>
#include <set>
using namespace std;

int secondLargestDigit(const string& str) {
    set<int> digits;
    for (char ch : str) {
        if (isdigit(ch)) {
            digits.insert(ch - '0');
        }
    }
    if (digits.size() < 2) {
        return -1; // No second largest digit
    }
    auto it = digits.rbegin();
    ++it;
    return *it;
}

int main() {
    string str1 = "2947578";
    string str2 = "1241";

    cout << "Second largest digit in string 1: " << secondLargestDigit(str1) <<
endl;
    cout << "Second largest digit in string 2: " << secondLargestDigit(str2) <<
endl;

    return 0;
}
```

3.Input a string and return the number of substrings that contain only vowels.
Input : str = "abjkoe"
Output : 4
Explanation : The possible substrings that only contain vowels are "a" , "o" , "e" , "oe"
Input : str = "hgdhpw"
Output : 0

```
#include <iostream>
#include <string>
#include <set>
using namespace std;

bool isVowel(char ch) {
    ch = tolower(ch);
    return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';
}

int countVowelSubstrings(const string& str) {
    int count = 0;
    for (int i = 0; i < str.length(); ++i) {
        if (isVowel(str[i])) {
            for (int j = i; j < str.length(); ++j) {
                if (isVowel(str[j])) {
                    ++count;
                } else {
                    break;
                }
            }
        }
    }
    return count;
}

int main() {
    string str1 = "abjkoe";
    string str2 = "hgdhpw";

    cout << "Number of vowel substrings in string 1: " <<
countVowelSubstrings(str1) << endl;
    cout << "Number of vowel substrings in string 2: " <<
countVowelSubstrings(str2) << endl;

    return 0;
}
```

4.Given an array of strings. Check whether they are anagram or not.
Input : s = "car" , t = "arc"
Output : True
Input : s = "book" , t = "hook"
Output : False

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

bool areAnagrams(const string& s, const string& t) {
    if (s.length() != t.length()) {
        return false;
    }
    string sortedS = s;
    string sortedT = t;
    sort(sortedS.begin(), sortedT.end());
    sort(sortedT.begin(), sortedT.end());
    return sortedS == sortedT;
}

int main() {
    string s1 = "car";
    string t1 = "arc";
    string s2 = "book";
    string t2 = "hook";

    cout << "Are strings 1 anagrams? " << (areAnagrams(s1, t1) ? "True" :
"False") << endl;
    cout << "Are strings 2 anagrams? " << (areAnagrams(s2, t2) ? "True" :
"False") << endl;
```

```
    return 0;
}
```

5.Given a sentence ‘str’, return the word that is lexicographically maximum.
Input : str = "proud to be pwians"
Output : pwians
Input : str = "decode dsa with pw"
Output : with

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;

string findLexicographicallyMaxWord(const string& str) {
    stringstream ss(str);
    string word, maxWord;
    while (ss >> word) {
        if (word > maxWord) {
            maxWord = word;
        }
    }
    return maxWord;
}

int main() {
    string str1 = "proud to be pwians";
    string str2 = "decode dsa with pw";

    cout << "Lexicographically maximum word in string 1: " <<
findLexicographicallyMaxWord(str1) << endl;
    cout << "Lexicographically maximum word in string 2: " <<
findLexicographicallyMaxWord(str2) << endl;

    return 0;
}
```