12/10 Assignment

1. Given a string S, find the longest palindromic substring in S. DONE

**Input:** "babad"

**Output:** "bab"

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

**# A O(n^2) time and O(1) space program to find the**

**#longest palindromic substring**

**# This function prints the longest palindrome substring (LPS)**

**# of str[]. It also returns the length of the longest palindrome**

**def longestPalSubstr(string):**

**maxLength = 1**

**start = 0**

**length = len(string)**

**low = 0**

**high = 0**

**# One by one consider every character as center point of**

**# even and length palindromes**

**for i in xrange(1, length):**

**# Find the longest even length palindrome with center**

**# points as i-1 and i.**

**low = i - 1**

**high = i**

**while low >= 0 and high < length and string[low] == string[high]:**

**if high - low + 1 > maxLength:**

**start = low**

**maxLength = high - low + 1**

**low -= 1**

**high += 1**

**# Find the longest odd length palindrome with center**

**# point as i**

**low = i - 1**

**high = i + 1**

**while low >= 0 and high < length and string[low] == string[high]:**

**if high - low + 1 > maxLength:**

**start = low**

**maxLength = high - low + 1**

**low -= 1**

**high += 1**

**print "Longest palindrome substring is:",**

**print string[start:start + maxLength]**

**return maxLength**

1. Given a string, recursively remove adjacent duplicate characters from string. The output string should not have any adjacent duplicates. See following examples. DONE

Input:  azxxzy

Output: ay

First "azxxzy" is reduced to "azzy". The string "azzy" contains duplicates, so it is further reduced to "ay".

Input: caaabbbaacdddd

Output: Empty String

Input: acaaabbbacdddd

Output: acac

**# Python program to remove all adjacent duplicates from a string**

**# Recursively removes adjacent duplicates from str and returns**

**# new string. las\_removed is a pointer to last\_removed character**

**def removeUtil(string, last\_removed):**

**# If length of string is 1 or 0**

**if len(string) == 0 or len(string) == 1:**

**return string**

**# Remove leftmost same characters and recur for remaining**

**# string**

**if string[0] == string[1]:**

**last\_removed = ord(string[0])**

**while len(string) > 1 and string[0] == string[1]:**

**string = string[1:]**

**string = string[1:]**

**return removeUtil(string, last\_removed)**

**# At this point, the first character is definiotely different**

**# from its adjacent. Ignore first character and recursively**

**# remove characters from remaining string**

**rem\_str = removeUtil(string[1:], last\_removed)**

**# Check if the first character of the rem\_string matches**

**# with the first character of the original string**

**if len(rem\_str) != 0 and rem\_str[0] == string[0]:**

**last\_removed = ord(string[0])**

**return (rem\_str[1:])**

**# If remaining string becomes empty and last removed character**

**# is same as first character of original string. This is needed**

**# for a string like "acbbcddc"**

**if len(rem\_str) == 0 and last\_removed == ord(string[0]):**

**return rem\_str**

**# If the two first characters of str and rem\_str don't match,**

**# append first character of str before the first character of**

**# rem\_str.**

**return ([string[0]] + rem\_str)**

**def remove(string):**

**last\_removed = 0**

**return toString(removeUtil(toList(string), last\_removed))**

**# Utility functions**

**def toList(string):**

**x = []**

**for i in string:**

**x.append(i)**

**return x**

**def toString(x):**

**return ''.join(x)**

1. Given two strings, the task is to find if a string ('a') can be obtained by rotating another string ('b') by two places.  
   Examples:

Input : a = "amazon"         b = "azonam"  // rotated anti-clockwise

Output : True

Input : a = "amazon"        b = "onamaz"  // rotated clockwise

Output : True

public class RotatingString {

public static void main(String[] args) {

  String str1 = "amazon";

  String str2 = "azonam";

   isRotated(str1, str2) ;

}

public static boolean isRotated(String str1, String str2)

{

String s = str1 + str2;

if(s.contains(str2))

return true;

else

return false;

}

}

**Output: Yes**

1. Given an string in roman no format (s)  your task is to convert it to integer . Done

**Example:**

**Input**

2

V

III

**Output**

5

3

|  |  |
| --- | --- |
| class py\_solution: |  |
| 2 | def roman\_to\_int(self, s): |
| 3 | rom\_val = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000} |
| 4 | int\_val = 0 |
| 5 | for i in range(len(s)): |
| 6 | if i > 0 and rom\_val[s[i]] > rom\_val[s[i - 1]]: |
| 7 | int\_val += rom\_val[s[i]] - 2 \* rom\_val[s[i - 1]] |
| 8 | else: |
| 9 | int\_val += rom\_val[s[i]] |
| 10 | return int\_val |

1. Given two strings ‘X’ and ‘Y’, find the length of the longest common substring. DONE

Input : X = "abcdxyz", y = "xyzabcd"

Output : 4

The longest common substring is "abcd" and is oflength 4.

Input : X = "zxabcdezy", y = "yzabcdezx"

Output : 6

The longest common substring is "abcdez" and is oflength 6.

# Python3 implementation of Finding

# Length of Longest Common Substring

# Returns length of longest common

# substring of X[0..m-1] and Y[0..n-1]

**def** LCSubStr(X, Y, m, n):

    # Create a table to store lengths of

    # longest common suffixes of substrings.

    # Note that LCSuff[i][j] contains the

    # length of longest common suffix of

    # X[0...i-1] and Y[0...j-1]. The first

    # row and first column entries have no

    # logical meaning, they are used only

    # for simplicity of the program.

    # LCSuff is the table with zero

    # value initially in each cell

    LCSuff **=** [[0 **for** k **in** range(n**+**1)] **for** l **in** range(m**+**1)]

    # To store the length of

    # longest common substring

    result **=** 0

    # Following steps to build

    # LCSuff[m+1][n+1] in bottom up fashion

**for** i **in** range(m **+** 1):

**for** j **in** range(n **+** 1):

**if** (i **==** 0 **or** j **==** 0):

                LCSuff[i][j] **=** 0

**elif** (X[i**-**1] **==** Y[j**-**1]):

                LCSuff[i][j] **=** LCSuff[i**-**1][j**-**1] **+** 1

                result **=** max(result, LCSuff[i][j])

**else**:

                LCSuff[i][j] **=** 0

**return** result

1. Given a string, find length of the longest substring with all distinct characters.  For example, for input "abca", the output is 3 as "abc" is the longest substring with all distinct characters. DONE

def distinctcharacter(txt):

lst = []

maxlength = 0

for i in range(0,len(txt)):

mem = set()

mem.add(txt[i])

j = i + 1

while True:

if j == len(txt):

if maxlength < j- i:

maxlength = j- i

break

if txt[j] not in mem:

mem.add(txt[j])

j = j + 1

else:

if maxlength < j- i:

maxlength = j- i

break

return maxlength

t = int(input())

while t>0:

txt = input()

print(distinctcharacter(txt))

t = t - 1

1. Your task  is to implement the function **atoi**. The function takes a string(str) as argument and converts it to an integer and returns it.

# function should return an integer /didn’t finish

def atoi(string):

# Code here

try:

return int(string)

except:

return -1

1. Your task  is to implement the function **strstr**. The function takes two strings as arguments(s,x) and  locates the occurrence of the string x in the string s. The function returns and integer denoting  the first occurrence of the string x . DONE

# you may python module's

def strstr(s, subs):

if subs not in s:

return -1

else:

rj = s.replace(subs, ' ').split()

if len(rj) != 0:

return len(rj[0])

else:

return 0

1. Given a array of **N**strings, find the longest common prefix among all strings present in the array.

Input: apple ape April

Output: ap

t=int(input())

for x in range(t):

n=int(input())

a=input().split()

s=[]

for i in range(len(a)):

s.append(len(a[i]))

q=min(s)

st=""

i=0

while 1:

if i>=q:

break

k=a[0][i]

c=0

for j in range(len(a)):

if k==a[j][i]:

c=c+1

if c<n:

break

i=i+1

st=st+k

if len(st)==0:

print(-1)

else:

print(st)

1. Design a URL shortner like bit.ly

#!/usr/bin/env python # Import the modules

import bitlyapi

import sys

# Define your API information

API\_USER = "your\_api\_username"

API\_KEY = "your\_api\_key"

b = bitlyapi.BitLy(API\_USER, API\_KEY)

# Define how to use the program

usage = """Usage: python shortener.py [url] e.g python shortener.py <http://www.google.com>"""

if len(sys.argv) != 2:

print usage

sys.exit(0)

longurl = sys.argv[1]

response = b.shorten(longUrl=longurl)

print response['url']