DATE:19/05/2020

1. We have a Letter or a word then we need add some letters to it and need to find out shortest palindrome  
For example we take "S": S will be the shortest palindrome string.  
If we take "xyz": zyxyz will be the shortest palindrome string  
So we need to add some characters to the given string or character and find out what will be the shortest palindrome string by using simple java program

import java.util.\*;

public class Main{

public static String shortestPalindrome(String str) {

int x=0;

int y=str.length()-1;

while(y>=0){

if(str.charAt(x)==str.charAt(y)){

x++;

}

y--;

}

if(x==str.length())

return str;

String suffix = str.substring(x);

String prefix = new StringBuilder(suffix).reverse().toString();

String mid = shortestPalindrome(str.substring(0, x));

return prefix+mid+suffix;

}

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.println("Enter a String to find out shortest palindrome");

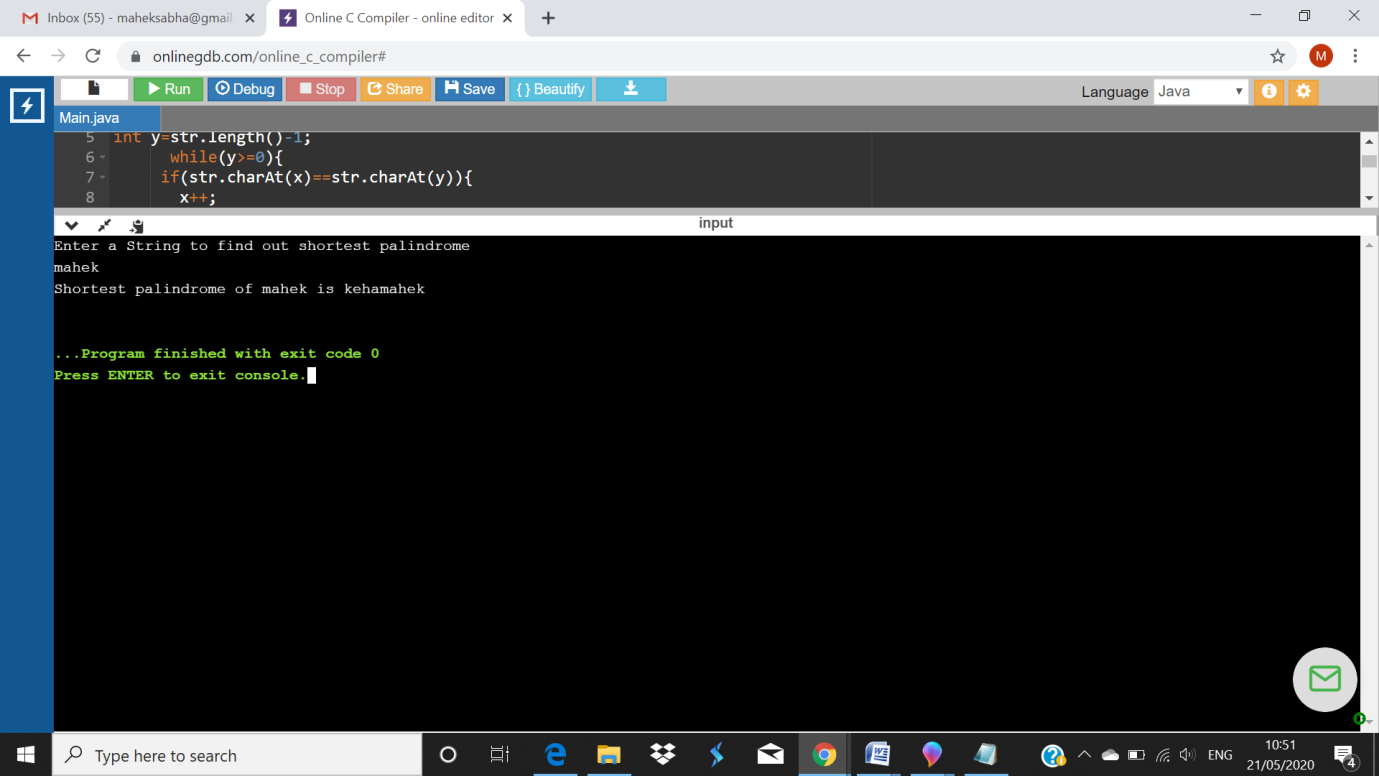
String str=in.nextLine();

System.out.println("Shortest palindrome of "+str+" is "+shortestPalindrome(str));

}

}

**Output:**

****

2. Write a simple code to identify given linked list is palindrome or not by using stack.  
First take a Stack. Traverse through each node of the linked list and push each node value to Stack.  
Once the traversal & copying is done, iterate through linked list from head node again.  
In each iteration, pop one stack element and compare with node value in respective iteration. It is expected to match stack popped value with node value.  
In case of all matches, its a palindrome. Any one element mismatch makes it not a palindrome.

import java.util.Stack;

public class Main {

public static void main(String[] a){

Node n1 = new Node(10);

Node n2 = new Node(28);

Node n3 = new Node(15);

Node n4 = new Node(29);

Node n5 = new Node(10);

n1.next = n2;

n2.next = n3;

n3.next = n4;

n4.next = n5;

boolean result = isPalindrome(n1);

System.out.println("Is it palindrome: "+result);

}

static class Node {

int data;

Node next;

Node(int tmp) {

data = tmp;

}

}

static boolean isPalindrome(Node head) {

Node tempNode = head;

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

while(tempNode != null) {

stack.push(tempNode.data);

tempNode = tempNode.next;

}

while(head != null) {

if(head.data != stack.pop()) {

return Boolean.FALSE;

}

head = head.next;

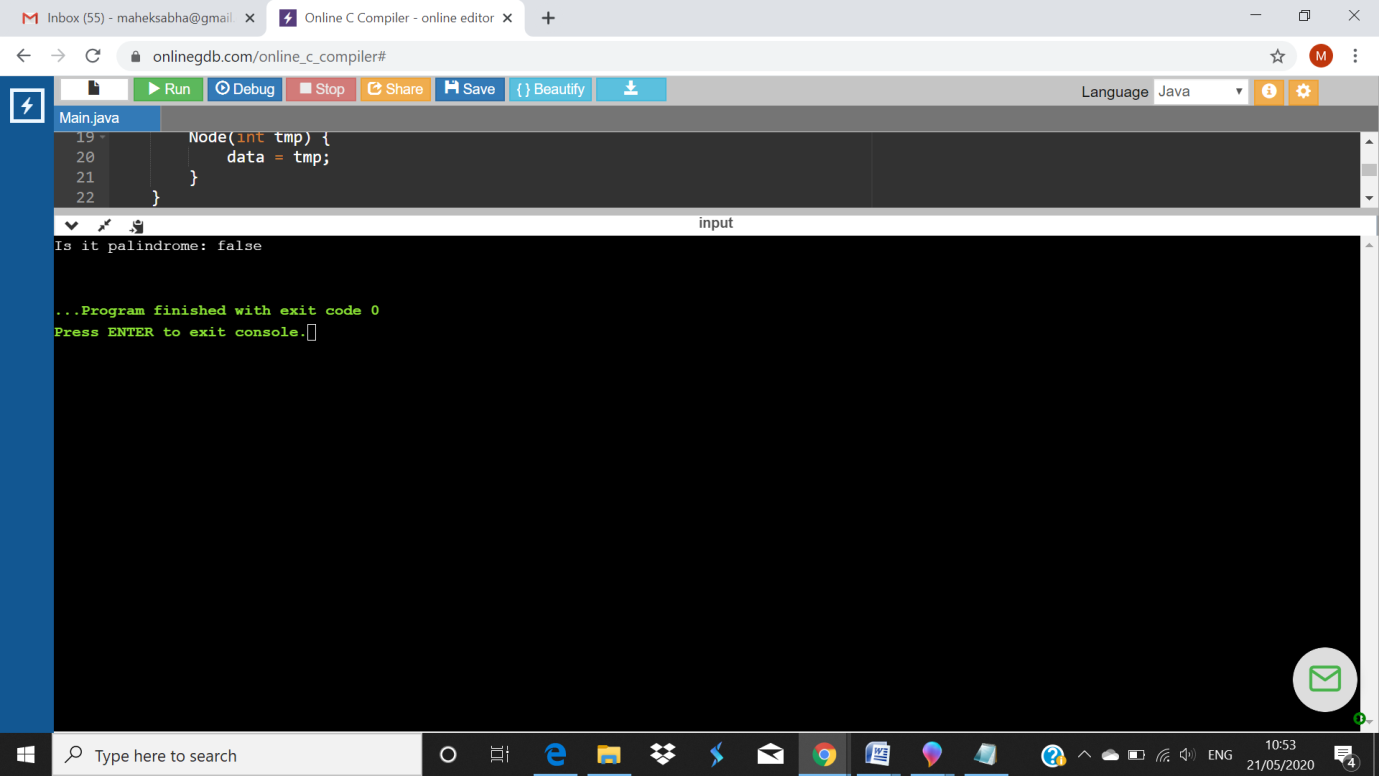
}

return Boolean.TRUE;

}

}

**Output:**

****

3.A user will input two strings, and we find if one of the strings is a sub sequence of the other. Program prints “yes” if either the first string is a sub sequence of the second string or the second string is a sub sequence of the first string.  
Assume that, the length of the first string is smaller than or equal to the length of the second string.

**An expected output of the program:**

Input the first string  
tree  
Input the second string  
Computer science is awesome  
YES

#include <stdio.h>  
#include <string.h>

int check\_subsequence (char [], char[]);

int main () {  
  int flag;  
  char s1[1000], s2[1000];

  printf("Input first string**\n**");  
  gets(s1);

  printf("Input second string**\n**");  
  gets(s2);

  if (strlen(s1) < strlen(s2))  
    flag = check\_subsequence(s1, s2);  
  else  
    flag = check\_subsequence(s2, s1);

  if (flag)  
    printf("YES**\n**");  
  else  
    printf("NO**\n**");

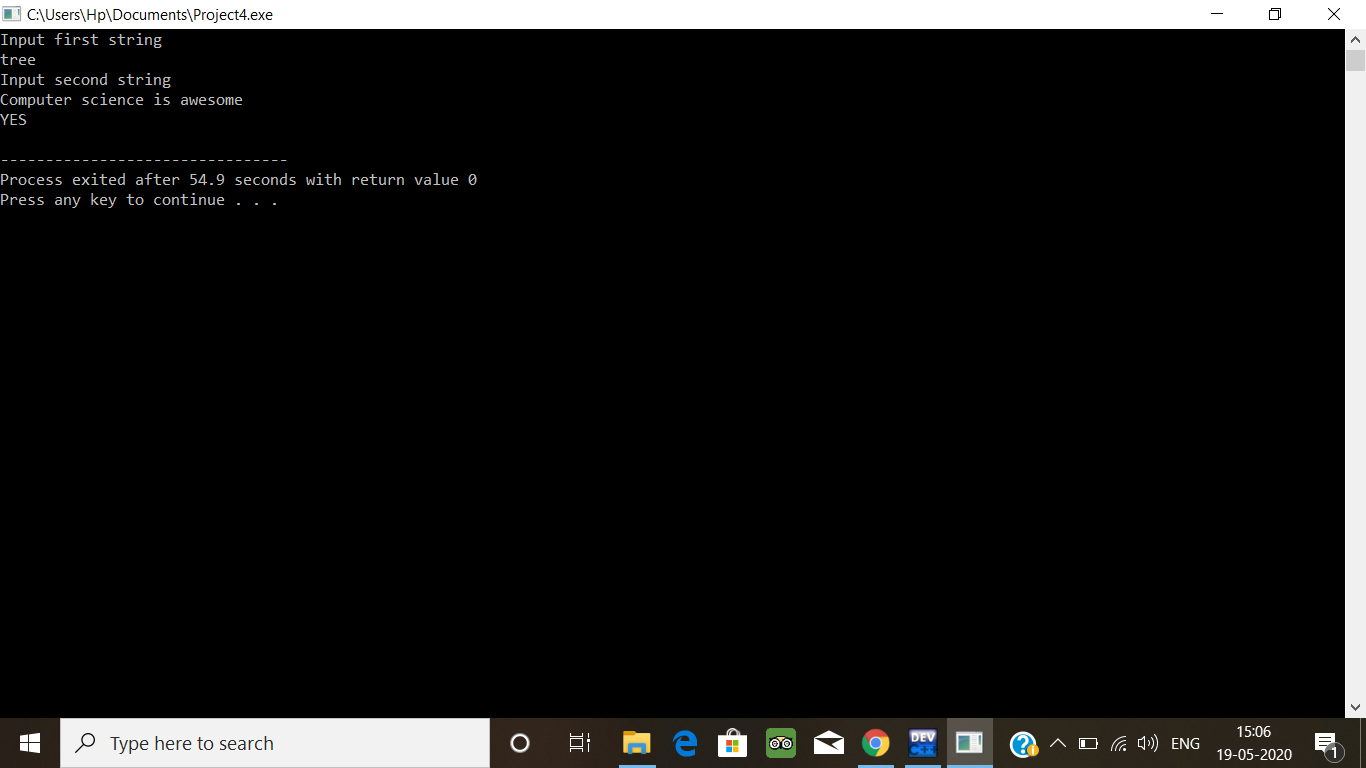
  return 0;  
}

int check\_subsequence (char a[], char b[]) {  
  int c, d;

  c = d = 0;

  while (a[c] != '**\0**') {  
    while ((a[c] != b[d]) && b[d] != '**\0**') {  
      d++;  
    }  
    if (b[d] == '**\0**')  
      **break**;  
    d++;  
    c++;  
  }  
  if (a[c] == '**\0**')  
    return 1;  
  else  
    return 0;  
}

**output**

****