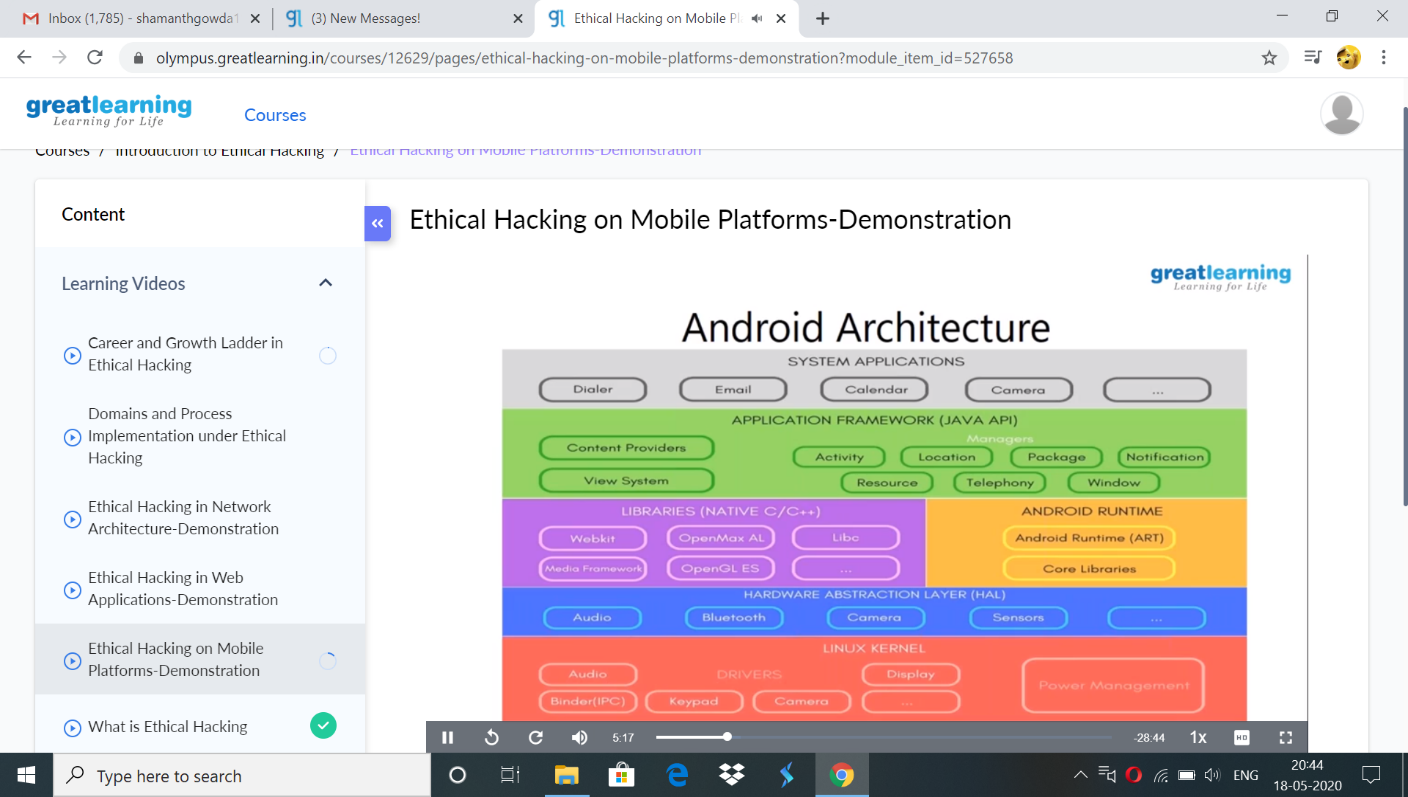
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **19-05-20** | | | | **Name:** | **Sahana c** | |
| **Sem & Sec** | **VI B** | | | | **USN:** | **4AL17CS116** | |
| **Online Test Summary** | | | | | | | |
| **Subject** | | **CVG IA Test** | | | | | |
| **Max. Marks** | | **60** | | **Score** | | **54** | |
| **Certification Course Summary** | | | | | | | |
| **Course** | **Ethical Hacking** | | | | | | |
| **Coding Challenges**  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.  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. | | | | | | | |
| **Certificate Provider** | | | **Great learning** | **Duration** | | | **6 days** |
| **Status:Completed** | | | | | | | |
| **Uploaded the report in Github** | | | | **Yes** | | | |
| **If yes Repository name** | | | | **https://github.com/sahanasanu/Daliy-status** | | | |
| **Uploaded the report in slack** | | | | **Yes** | | | |

**Online Certification Details**

Lesson-2

* Ethical Hacking on Mobile platforms Demonstrations



* Topics covered:
* Android Architecture
* ADB(Android Debug Bridge)
* **Coding Challenge Details**

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.

**package** Tuesday;

**import** java.util.Scanner;

**public** **class** ShortestPalindromeDemo {

**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***);

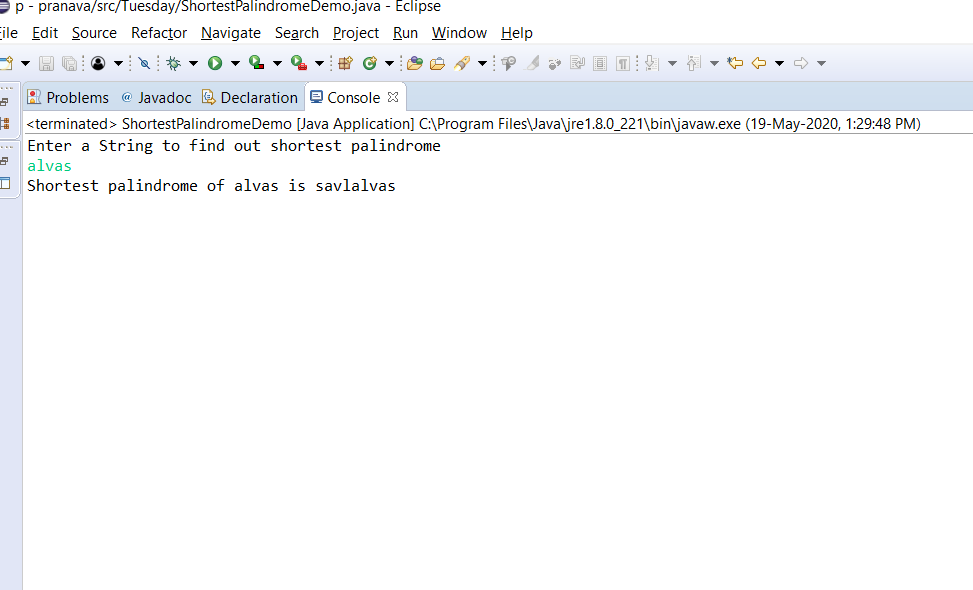
System.***out***.println("Enter a String to find out shortest palindrome");

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.

**package** Tuesday;

**import** java.util.Stack;

**public** **class** LinkedListPalindromeUsingStack {

**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(25);

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***;

}

}

