# ****Python Program to Read a Linked List in Reverses.****

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.next = None

class LinkedList:

def \_\_init\_\_(self):

self.head = None

self.last\_node = None

def append(self, data):

if self.last\_node is None:

self.head = Node(data)

self.last\_node = self.head

else:

self.last\_node.next = Node(data)

self.last\_node = self.last\_node.next

def display(self):

current = self.head

while current:

print(current.data, end = ' ')

current = current.next

def reverse\_llist(llist):

before = None

current = llist.head

if current is None:

return

after = current.next

while after:

current.next = before

before = current

current = after

after = after.next

current.next = before

llist.head = current

a\_llist = LinkedList()

data\_list = input('Please enter the elements in the linked list: ').split()

for data in data\_list:

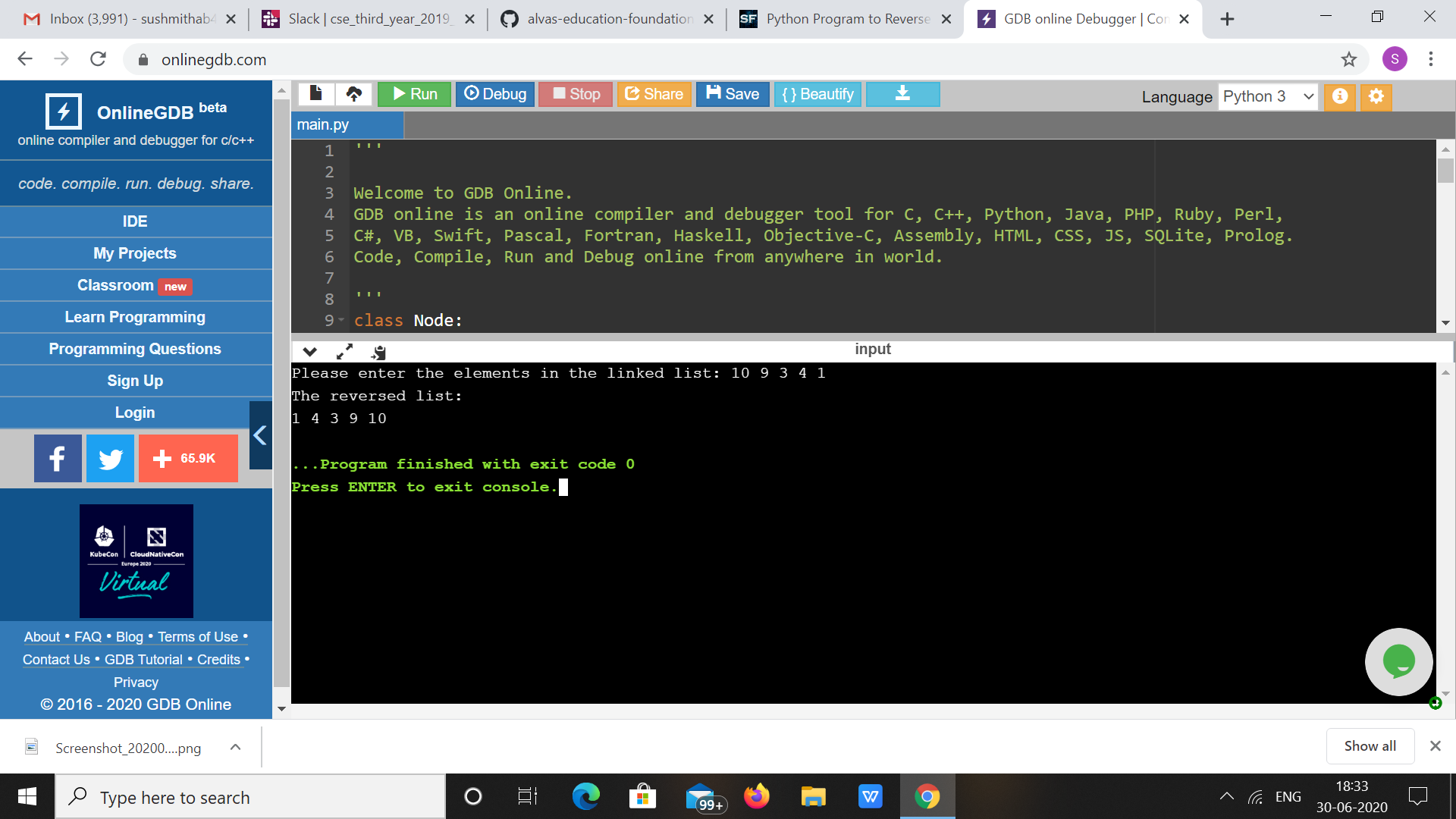
a\_llist.append(int(data))

reverse\_llist(a\_llist)

print('The reversed list: ')

a\_llist.display()

**OUTPUT**

****

2. **Write a Java Program to determine whether one string is a rotation of another.**

Description:  
In this program, we need to check whether a string is a rotation of another string or not.

String 1: abcde  
String 2: deabc  
String 1 + String 1: abcdeabcde

Consider the above example, suppose we need to check whether string 2 is a rotation of string 1. To find this, we concatenate string 1 with string 1. Then, try to find the string 2 in concatenated string. If string 2 is present in concatenated string then, string 2 is rotation of string 1. String 2 deabc is found on the index 3 in concatenated string. So, deabc is rotation of abcde.

ALGORITHM  
STEP 1: START  
STEP 2: DEFINE String str1 = "abcde", str2 = "deabc"  
STEP 3: IF length of str1 not equals to str2 then PRINT "No"  
else go to STEP 4  
STEP 4: CONCATENATE str1 with str1.  
STEP 5: IF str2 present in str1 then PRINT "Yes" else PRINT "No".  
STEP 6: END

public class RotationString {

public static boolean checkRotation(String st1, String st2) {

if (st1.length() != st2.length()) {

return false;

}

String st3 = st1 + st1;

if (st3.contains(st2))

return true;

else

return false;

}

public static void main(String[] args) {

String str1 = "avajava";

String str2 = "javaava";

System.out.println("Checking if a string is rotation of another");

if (checkRotation(str1, str2)) {

System.out.println("Yes " + str2 + " is rotation of " + str1);

} else {

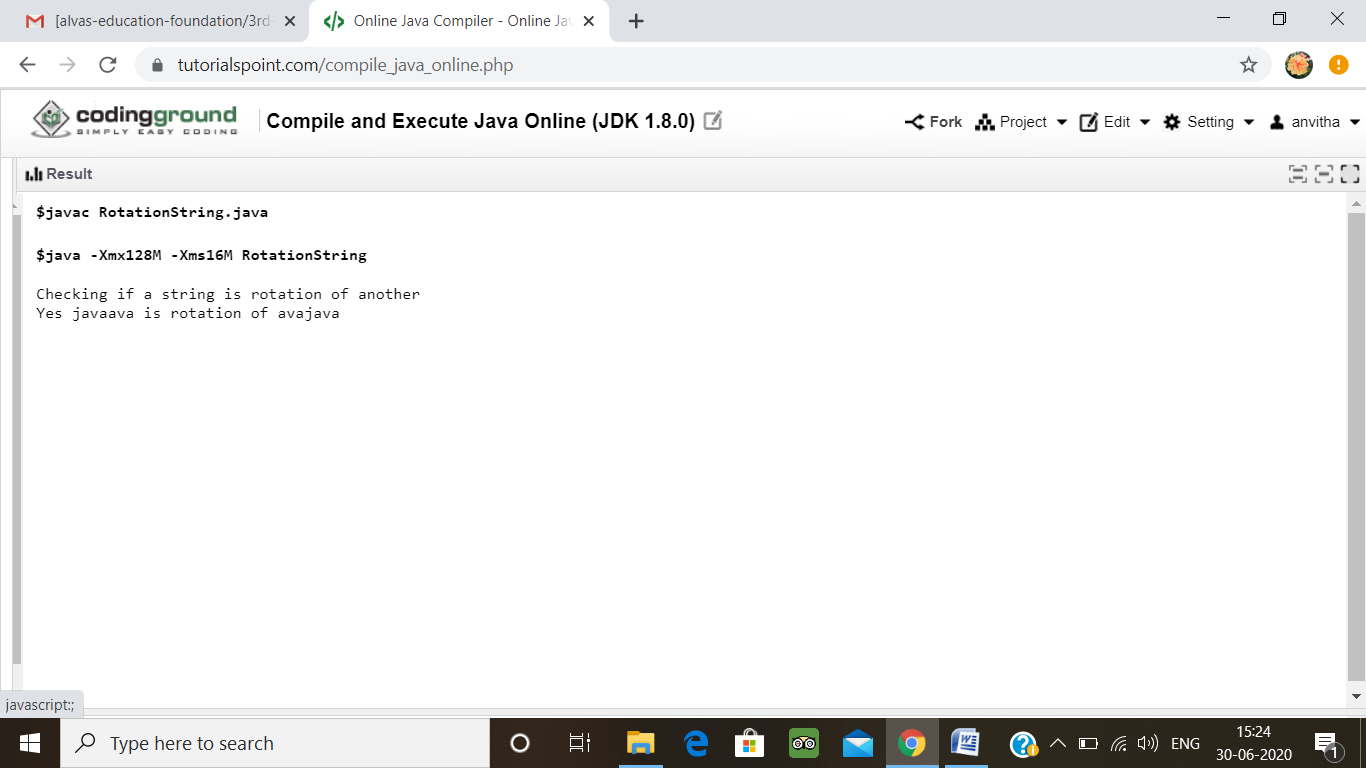
System.out.println("No " + str2 + " is not rotation of " + str1);

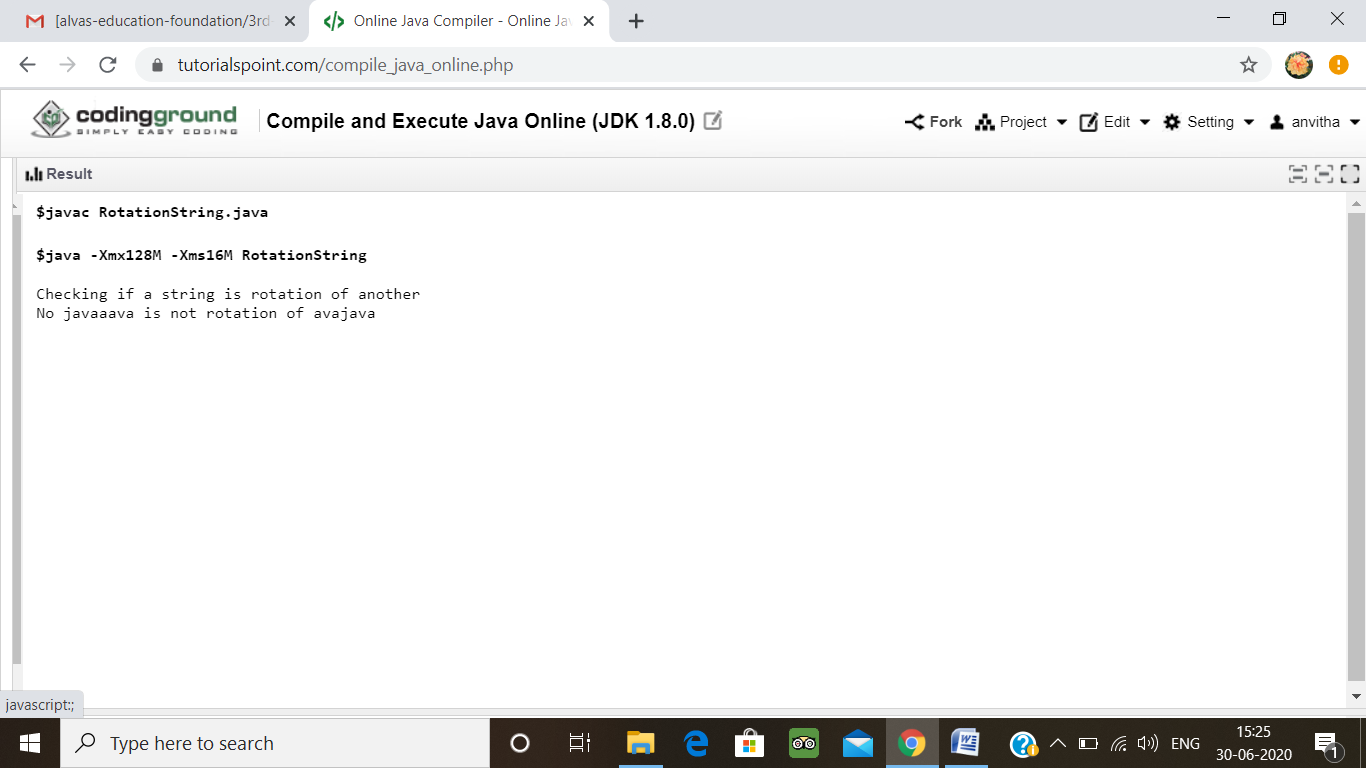
}

}

}

**Output:**





**3.Write a C Program to generate first n Ugly Numbers Write a C Program to generate first n Ugly Numbers.**

Ugly numbers are those number whose prime factors are 2, 3 or 5. From 1 to 15, there are 11 ugly numbers 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15. The numbers 7, 11, 13 are not ugly because they are prime. The number 14 is not ugly because in its prime factor the 7 will come.

# include<stdio.h>

# include<stdlib.h>

int maxDivide(int a, int b)

{

while (a%b == 0)

a = a/b;

return a;

}

int isUgly(int no)

{

no = maxDivide(no, 2);

no = maxDivide(no, 3);

no = maxDivide(no, 5);

return (no == 1)? 1 : 0;

}

int getNthUglyNo(int n)

{

int i = 1;

int count = 1;

while (n > count)

{

i++;

if (isUgly(i))

count++;

}

return i;

}

int main()

{

unsigned no = getNthUglyNo(15);

printf("15th ugly no. is %d ", no);

getchar();

return 0;

}

**Output:**

