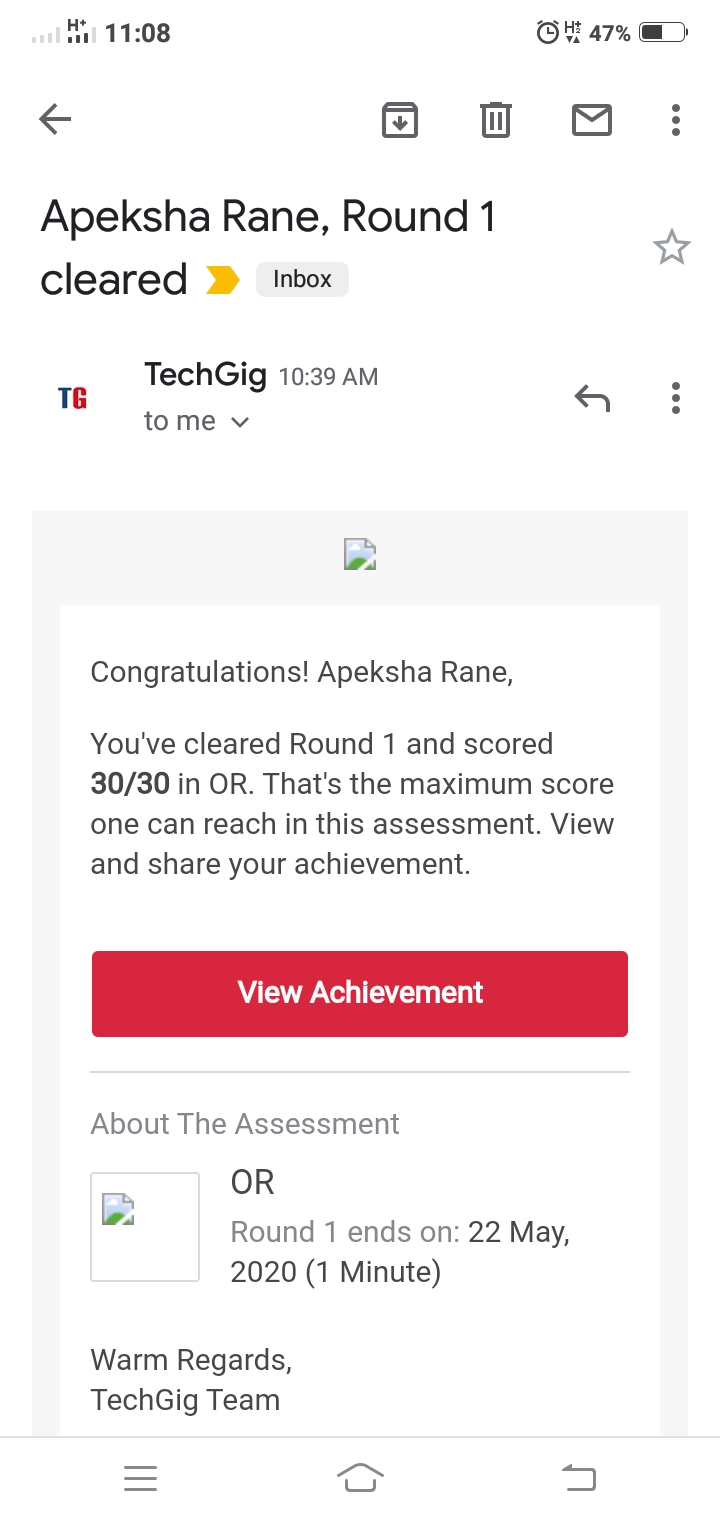
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **22-05-2020** | | | | | **Name:** | **Apeksha Rane** | |
| **Sem & Sec** | **A** | | | | | **USN:** | **4AL17CS010** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **OPERATION RESEARCH.** | | | | | | |
| **Max. Marks** | | **30** | | **Score** | | | **30** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **INTRODUCTION TO ETHICAL HACKING** | | | | | | | |
| **Certificate Provider** | | | Great learning  Academy | | **Duration** | | | 8hours |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement:** 1. Write a C Program to implement various operations of Singly Linked List Stack .  2. Write a Java Program to separate the Individual Characters from a String  3.Write a Java Program to find the largest and smallest word in a string | | | | | | | | |
| **Status: Done** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **YES** | | | |
| **If yes Repository name** | | | | | <https://github.com/Apeksha12appu/19-5-2020-online-coding-activity> | | | |
| **Uploaded the report in slack** | | | | | **YES** | | | |

Online Test Details:

Subject:-Operation Research.



Certification Course Details:

**Introduction to Ethical Hacking:**

Today I have studied **Ethical Hacking on Mobile Platforms-Demonstration:**

Under this topic I have studied:-

\* Mobile App Pen-Testing.

\* Android Operating System.

\* Android Architecture .

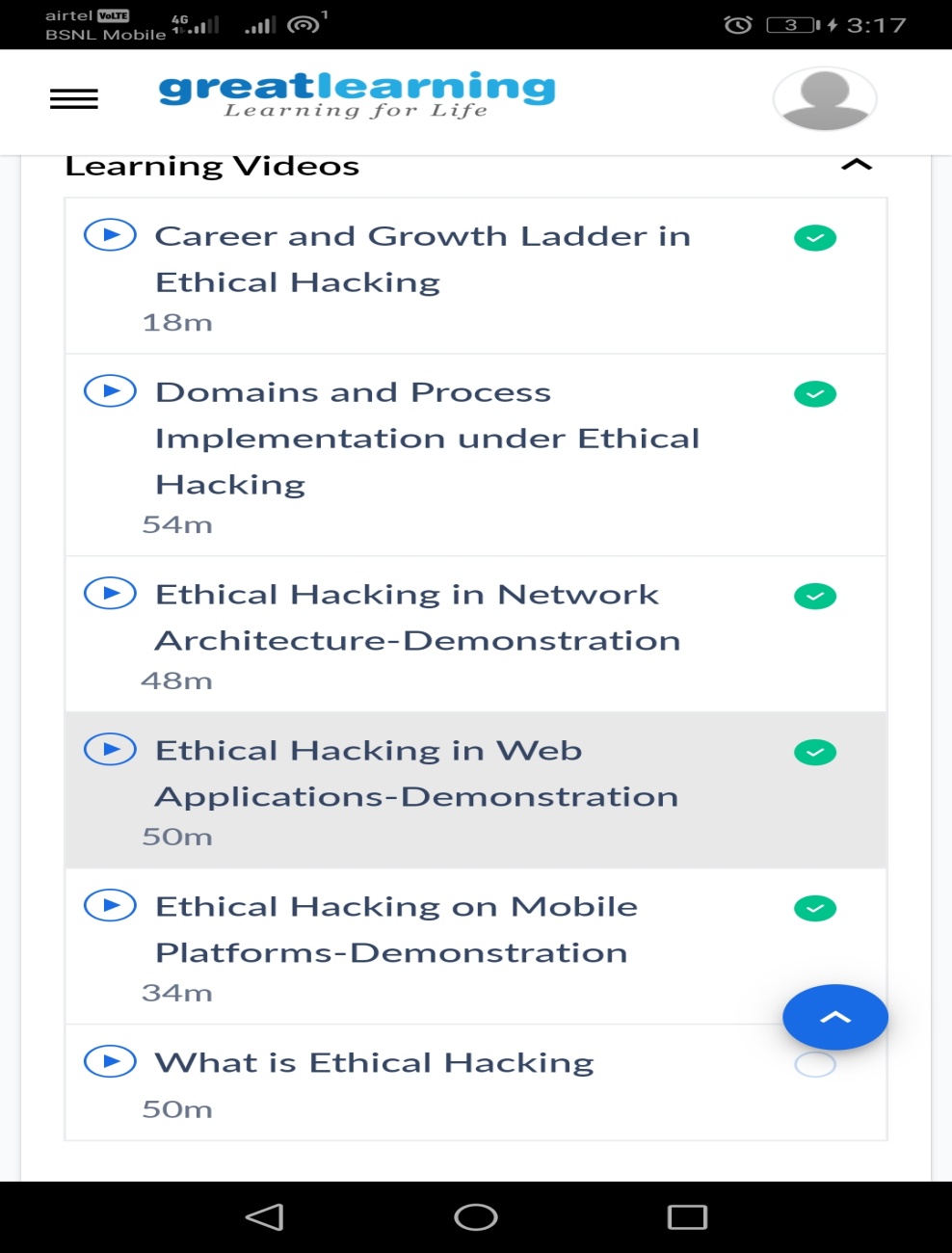
\* Android Manifest.

\* ADB(Android Debug Bridge)

\* Hardcoding Issue.

\* Insecure Data Storage.

\* Insecure Logging.



Coding Challenges Details:

1.Write a C Program to implement various operations of Singly Linked List Stack.

include <stdio.h>

#inclu#de <stdlib.h>

struct node

{

int info;

struct node \*ptr;

}\*top,\*top1,\*temp;

int topelement();

void push(int data);

void pop();

void empty();

void display();

void destroy();

void stack\_count();

void create();

int count = 0;

void main()

{

int no, ch, e;

printf("\n 1 - Push");

printf("\n 2 - Pop");

printf("\n 3 - Top");

printf("\n 4 - Empty");

printf("\n 5 - Exit");

printf("\n 6 - Dipslay");

printf("\n 7 - Stack Count");

printf("\n 8 - Destroy stack");

create();

while (1)

{

printf("\n Enter choice : ");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("Enter data : ");

scanf("%d", &no);

push(no);

break;

case 2:

pop();

break;

case 3:

if (top == NULL)

printf("No elements in stack");

else

{

e = topelement();

printf("\n Top element : %d", e);

}

break;

case 4:

empty();

break;

case 5:

exit(0);

case 6:

display();

break;

case 7:

stack\_count();

break;

case 8:

destroy();

break;

default :

printf(" Wrong choice, Please enter correct choice ");

break;

}

}

}

/\* Create empty stack \*/

void create()

{

top = NULL;

}

/\* Count stack elements \*/

void stack\_count()

{

printf("\n No. of elements in stack : %d", count);

}

/\* Push data into stack \*/

void push(int data)

{

if (top == NULL)

{

top =(struct node \*)malloc(1\*sizeof(struct node));

top->ptr = NULL;

top->info = data;

}

else

{

temp =(struct node \*)malloc(1\*sizeof(struct node));

temp->ptr = top;

temp->info = data;

top = temp;

}

count++;

}

void display()

{

top1 = top;

if (top1 == NULL)

{

printf("Stack is empty");

return;

}

while (top1 != NULL)

{

printf("%d ", top1->info);

top1 = top1->ptr;

}

}

void pop()

{

top1 = top;

if (top1 == NULL)

{

printf("\n Error : Trying to pop from empty stack");

return;

}

else

top1 = top1->ptr;

printf("\n Popped value : %d", top->info);

free(top);

top = top1;

count--;

}

int topelement()

{

return(top->info);

}

void empty()

{

if (top == NULL)

printf("\n Stack is empty");

else

printf("\n Stack is not empty with %d elements", count);

}

void destroy()

{

top1 = top;

while (top1 != NULL)

{

top1 = top->ptr;

free(top);

top = top1;

top1 = top1->ptr;

}

free(top1);

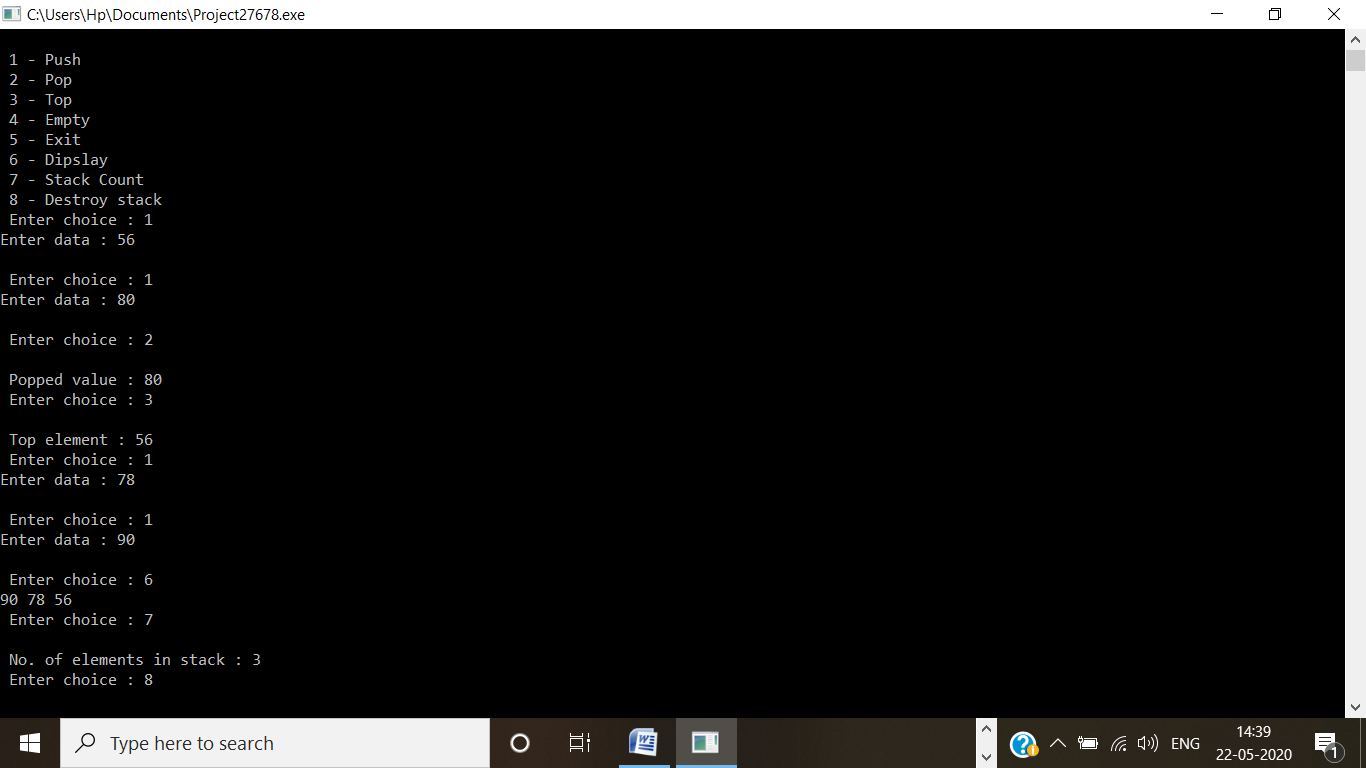
top = NULL;

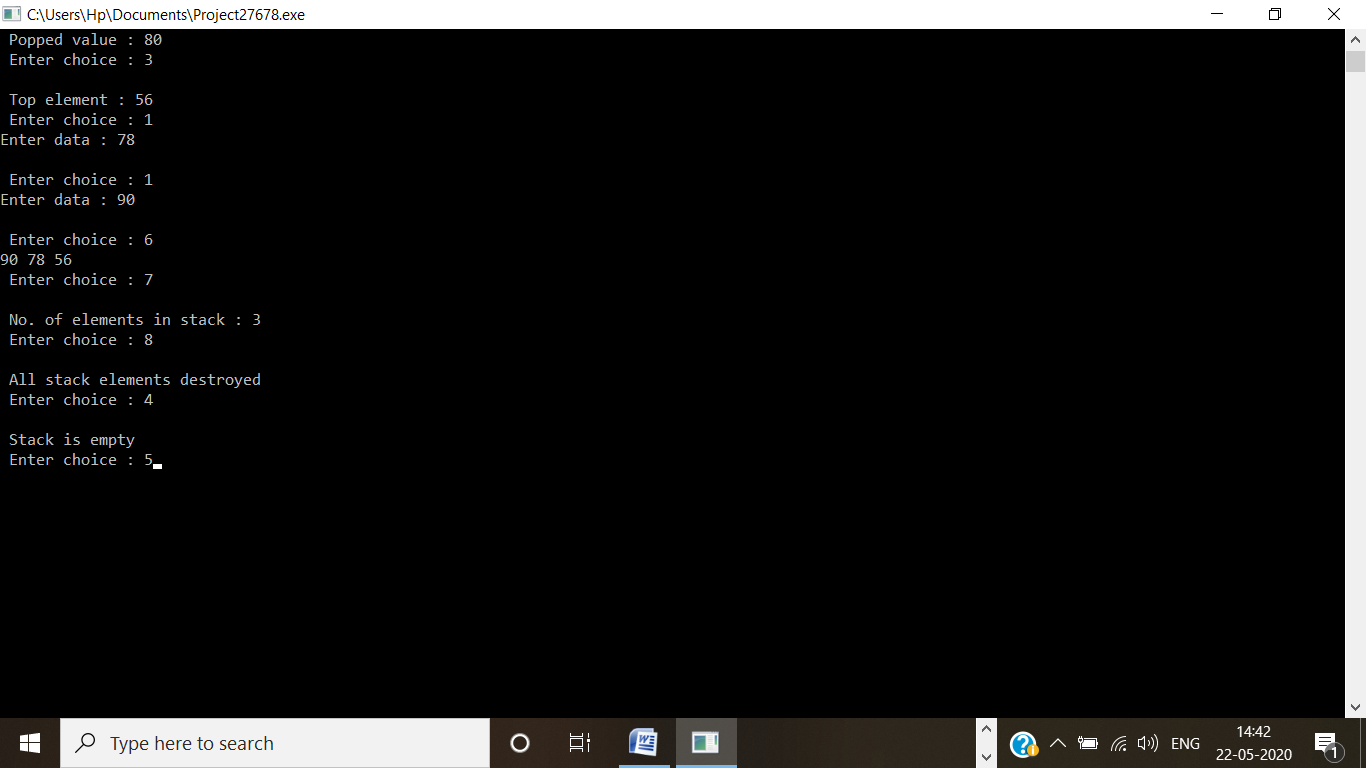
printf("\n All stack elements destroyed");

count = 0;

}

**Output:-**





2.Write a Java Program to separate the Individual Characters from a String

Description:  
In computer science, collection of characters including spaces is called as string. To separate an individual character from the string, individual characters are accessed through its index.

Algorithm  
STEP 1: START  
STEP 2: DEFINE String string = "characters "  
STEP 3: PRINT "Individual characters from given string: "  
STEP 4: SET i=0. REPEAT STEP 5 to STEP 6 UNTIL i<string.length()  
STEP 5: PRINT string.charAt(i)  
STEP 6: i=i+1  
STEP 7: END

public class Main

{

public static void main(String[] args) {

String string = "characters";

//Displays individual characters from given string

System.out.println("Individual characters from given string:");

//Iterate through the string and display individual character

for(int i = 0; i < string.length(); i++){

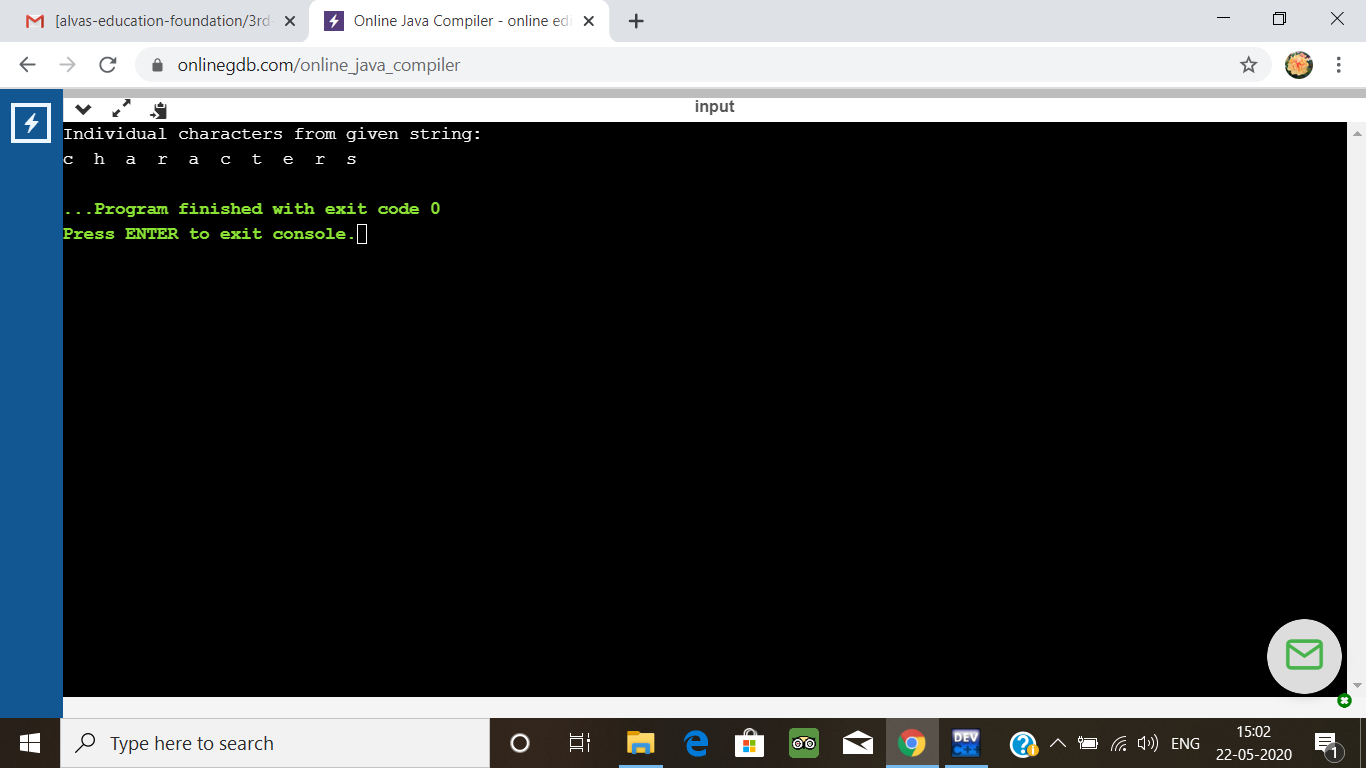
System.out.print(string.charAt(i) + " ");

}

}

}

**Output:**

****

3.Write a Java Program to find the largest and smallest word in a string.

Description:  
ALGORITHM  
STEP 1: START  
STEP 2: DEFINE String string="Hardships often prepare ordinary people for an extraordinary destiny"  
STEP 3: DEFINE word = " ", small = " ", large = " ".  
STEP 4: Make object of String[] words.  
STEP 5: SET length =0  
STEP 6: string = string + " "  
STEP 7: SET i=0. REPEAT STEP 8 to 9 STEP UNTIL i  
STEP 8: IF(string.charAt(i) != ' ') then  
word =word + string.charAt(i)  
else  
word[length]=word  
length =length + 1  
word = " "  
STEP 9: i=i+1  
STEP 10: small = large =words[0]  
STEP 11: SET k = 0. REPEAT STEP 12 to STEP 14 UNTIL k  
STEP 12: IF(small.length() > words[k].length())  
then  
small = words[k]  
STEP 13: IF(large.length() < words[k].length())  
then  
large = words[k]  
STEP 14: k = k + 1  
STEP 15: PRINT small  
STEP 16: PRINT large  
STEP 17: END

public class Main {

public static void main(String[] args){

String string = "Hardships often prepare ordinary people for an extraordinary destiny";

String word = "", small = "", large="";

String[] words = new String[100];

int length = 0;

string = string + " ";

for(int i = 0; i < string.length(); i++){

if(string.charAt(i) != ' '){

word = word + string.charAt(i);

}

else{

words[length] = word;

length++;

word = "";

}

}

small = large = words[0];

for(int k = 0; k < length; k++){

if(small.length() > words[k].length())

small = words[k];

if(large.length() < words[k].length())

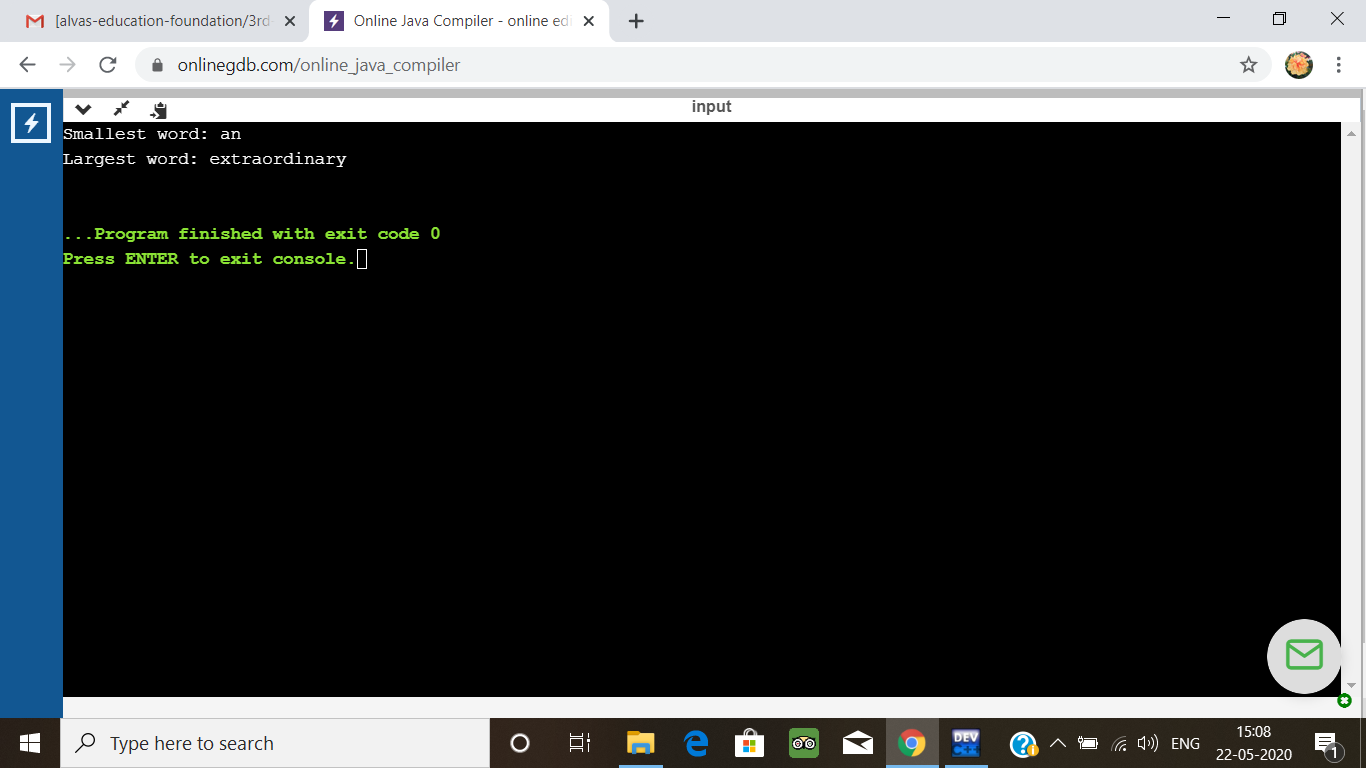
large = words[k];

}

System.out.println("Smallest word: " + small);

System.out.println("Largest word: " + large);

} }

**Output:** ****