FACULTY OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under Section 3 of UGC Act, 1956)
S.R.M. NAGAR, KATTANKULATHUR – 603 203
Kancheepuram District

BONAFIDE CERTIFICATE

Register No: _____

Certified to be the bonafide record of work done by _____ of _____B.Tech Degree course in the Practical _____ in **SRM Institute of Science and Technology, Kattankulathur** during the academic year .

Lab Incharge

Date:

Head of the Department

Submitted for University Examination held in _____ at **SRM Institute of Science and Technology,** Kattankulathur.

Date :

Examiner 1

Examiner 2

INDEX

ASSIGNMENTS

UNIT 1

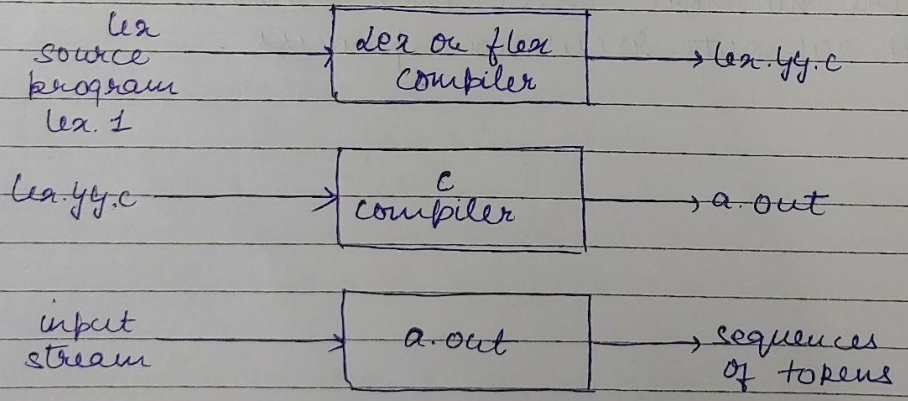
Palak Sharma
RA1911033010112

PAGE NO.:
DATE: / /

UNIT - 1

Scanner Generator tool

- The lex and flex scanner Generators
- Generates lexical analysers from the input that consists of regular expression description based on tokens of new language
- Lex and it's newer cousin flex are scanner generators.
- It systematically translate regular definations into C source code for efficient scanning.
- Generated code is easy to integrate in C applications.



→ Lex specifications

- A lex specifications of three parts:
  regular definations, C declaration in %{ %}
  %% %%

translation rules
%%
user-defined auxiliary procedures....

- The translation rules are of the form
  P_1 { action$_1$ }
  P_2 { action$_2$ }

  P_n { action$_n$ }

UNIT 2 AND UNIT 3

Palak Sharma
RA1911033010112

PAGE NO. :
DATE : / /

Unit 2 & Unit 3

Batch 2 :- Set D

1] Construct SLR parsing table for grammar

$E \rightarrow E+T / T$
$T \rightarrow T*F / F$
$F \rightarrow (E) / id$

First & follow :

FIRST (E) = { (, id }	FOLLOW (E) = { $, +,) }
FIRST (T) = { (, id }	FOLLOW (T) = { *, $, +,) }
FIRST (F) = { (, id }	FOLLOW (F) = { *, $, +,) }

Productions :

$E \rightarrow E+T$
$E \rightarrow T$
$T \rightarrow T*F$
$T \rightarrow F$
$F \rightarrow (E)$
$F \rightarrow id$

Augmented Grammar :

$E' \rightarrow E$
$E \rightarrow E+T$
$E \rightarrow T$
$T \rightarrow T*F$
$T \rightarrow F$
$F \rightarrow (E)$
$F \rightarrow id$

LR(0) items :

goto (I_0, E) - closure $[\{E' \to E^* \}, \{E \to E^* + T\}]$

$E' \to E^*$
$E \to E^* + T$ } I_1

goto (I_0, T) - closure $(\{E \to T.\}, \{T \to T.*F\})$

$E \to T.$
$T \to T.*F$ } I_2

goto $(I_0, ($

$F \to (.E)$
$E \to .E+T$
$E \to .T$
$T \to .T*F$
$T \to .F$
$F \to .(E)$
$F \to .id$ } I_4

goto (I_0, F)

$\{T \to F.\}$ I_3

goto (I_0, id)

$F \to id.$ I_5

goto $(I_1, +)$

$E \to E + .T$
$T \to .T * F$
$T \to .F$
$F \to .(E)$
$F \to .id$

} I_6

goto $(I_2, *)$

$T \to T * .F$
$F \to .(E)$
$F \to .id$

} I_7

goto (I_4, E)

$F \to (E.)$
$E \to E. + T$

} I_8

goto (I_6, T)

$E \to E + T.$
$T \to T. * F$

} I_9

goto (I_7, F)

$T \to T * F.$ } I_{10}

goto $(I_8,)$

$F \to (E).$

SLR parsing table:

State	Action						Goto		
	+	*	(	)	id	$	E	T	F
0			S_4		S_5		1	2	3
1	S_6					acc			
2	r_2	S_7		r_2		r_2			
3	r_4	r_4		r_4		r_4			
4			S_4		S_5		8	2	3
5	r_6	r_6		r_6		r_6			
6			S_4		S_5			9	3
7			S_4		S_5				10
8	S_6			S_{11}					
9	r_1	S_7		r_1		r_1			
10	r_3	r_3		r_3		r_3			
11		r_5		r_5		r_5			

Qe] CFG :

$L \to AQ \mid A$

$Q \to ug \mid Pg \mid dip$

$A \to 1z \mid YX$

$Y \to 2 \mid 3 \mid 4 \mid 5$

$z \to 8 \mid 9$

$X \to 8 \mid 1 \mid Y \mid 6 \mid 7 \mid z$

Eliminating left factoring of production 'L'.

$L \to AL'$	
$L' \to Q \mid E$	FIRST(L) = {1,2,3,4,5}
$Q \to ug$	FIRST(L') = {E, ug, Pg, dip}
$Q \to Pg$	FIRST(Q) = {ug, Pg, dip}
$Q \to dip$	FIRST(A) = {1,2,3,4,5}
$A \to Lz$	FIRST(Y) = {2,3,4,5}
$A \to YX$	FIRST(z) = {8,9}
$Y \to 2$	FIRST(X) = {0,1,2.....9}
$Y \to 3$	
$Y \to 4$	FOLLOW(L) = FOLLOW(L') = FOLLOW(Q)
$4 \to 5$	FOLLOW(A) = FOLLOW(u) = FOLLOW(4) = {$}
$z \to 8$	FOLLOW(Y) = {$,0,1,2 ... 9}
$z \to 9$	
$X \to 0$	
$X \to 1$	
$X \to Y$	
$X \to 6$	
$X \to 7$	
$X \to z$	

	0	1	2	3	4	5	6	7	8	9	ug	kg	dip	$
L		L→AL'	L→AL'	L→AL'	L→AL'	L→AL'					L→Q	L=Q	L→Q	L→E
L'											Q→ug	Q→Pg	Q→dip	
Q														
A		A→1z1yx	A→1z1yx	A→1z1yx	A→1z1yx	A→1z1yx								
y			y→2	y→3	y→4	y→5								
Z									폭8	2→9				
X		x→0	x→1	x→y	x→y	x→y	x→6	x→z	x→z	x→z				

Q3] Grammar

A → T | b
T → pcmcs | pcmb
b → ce | it | ee | ec
i/p : pccsb

Stack	I/p buffer	Action
$	pccsb$	Shift p
$p	ccsb$	Shift c
$pc	csb$	Shift cs
$pccs	b$	Shift b
$pccsb	$	No further shift action

Conclusion :-
The input can not be accepted. The student who has taken pccsb is not eligible to get admission.

UNIT 4

Palak Sharma
RA1911033010112

PAGE NO.:
DATE: / /

UNIT - 4

CROSS COMPILER

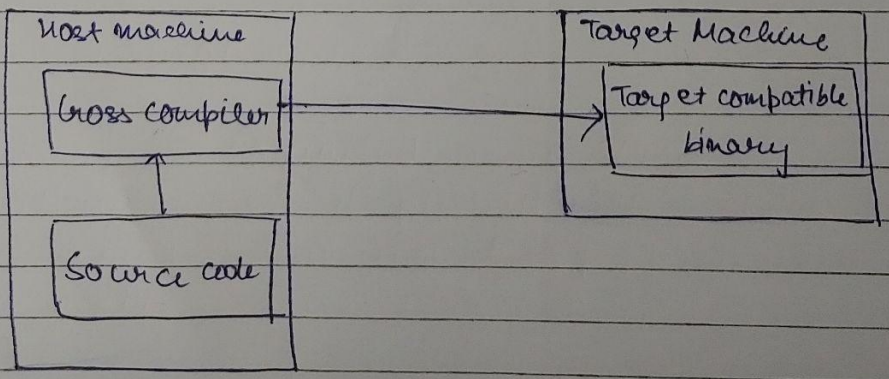
A cross compiler is a type of compiler, that generates the machine code targeted to run on a system different than the one generating it.

For example, A compiler that runs on windows platform also generates a code that runs on linux platform is called a cross compiler.

The process of creating executable code for different machines is also called "retargeting"

The cross compiler is also known a "Retargetable Compiler"
GNU GCC is an example of cross compiler

Cross compiler Operation



Uses of cross compiler :

- Embedded computers where a device has extremely limited resources.
- Compiling for multiple machines.
- Compiling on a server farm.
- Bootstraping to a new platform
- Compiling native code for emulators for older now-obsolete platforms like the Commodore 64 or Apple II by enthusiasts who use cross compilers that run on a current platform.

UNIT 5

Palak Sharma
RA1911033010112

PAGE NO.:
DATE: / /

UNIT-5

Storage allocation techniques

The different ways to allocate memory are :-

1. Static ~~allocation~~ storage allocation
2. Stack storage allocation
3. Heap storage allocation

Static ~~allocation~~ storage allocation :

- In it names are bounded to storage locations.
- If memory is created in compile time then memory will be created in static area and only once.
- It supports the dynamic structure, that means memory is created only at compile time and deallocated after program completion.
- The drawback is that the size and position of data objects should be known at compile time.
- Another draw back is recursion procedure.

Stack storage allocation :

- In it storage is organized as a stack.
- An activation record is pushed into a stack when activation begins and it is popped when the activation ends.

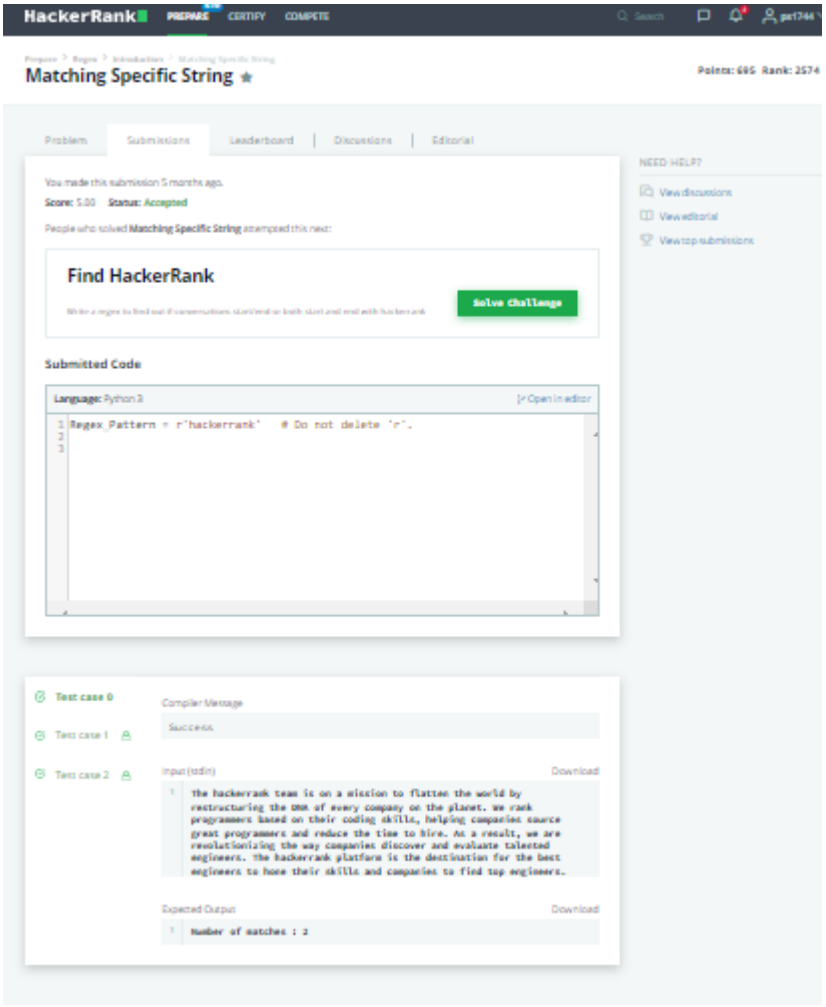
- Activation record contains the locals so that they are bound to fresh storage in each activation record.
- It works on the ~~bo~~ basis of last-in-first-out (LIFO).

Heap Storage allocation

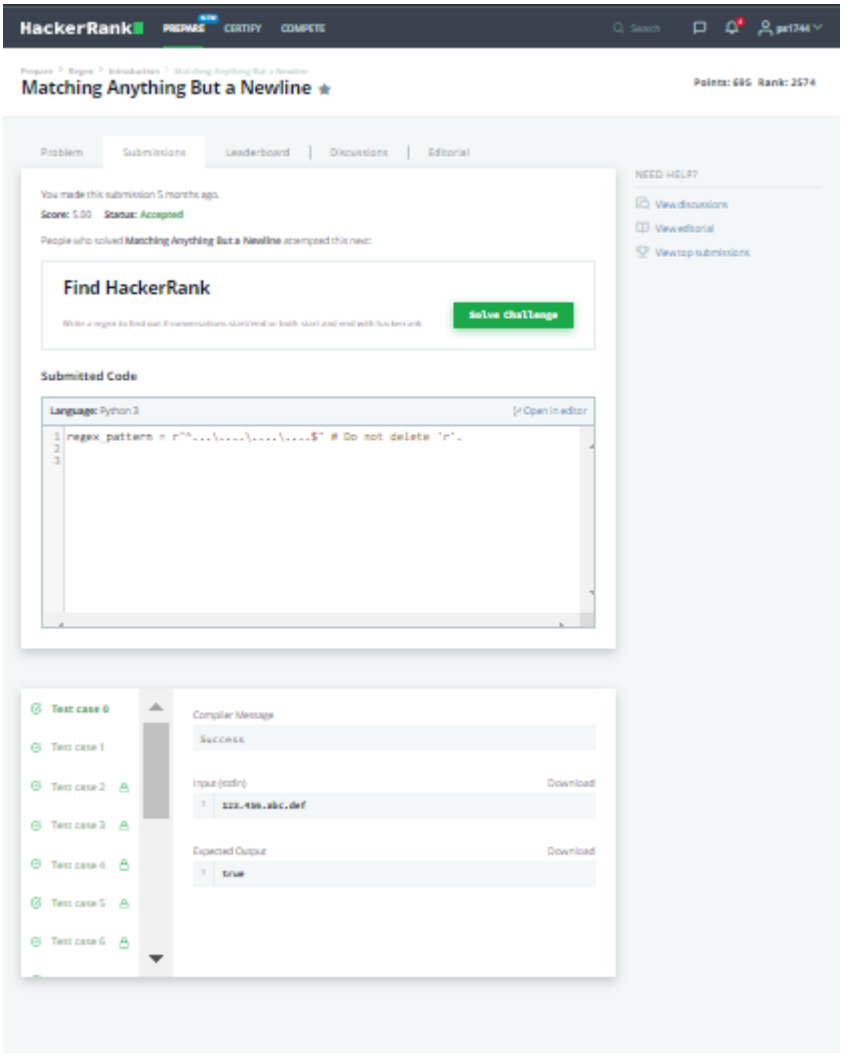
- It is most flexible allocation scheme
- Allocation and deallocation of memory can be done at any time and at any place depending upon the user's requirement.
- Heap allocation is used to allocate memory to the variables dynamically and when the variables are no more used then claim it back.
- It supports recursion process.

HackerRank Regex Problems

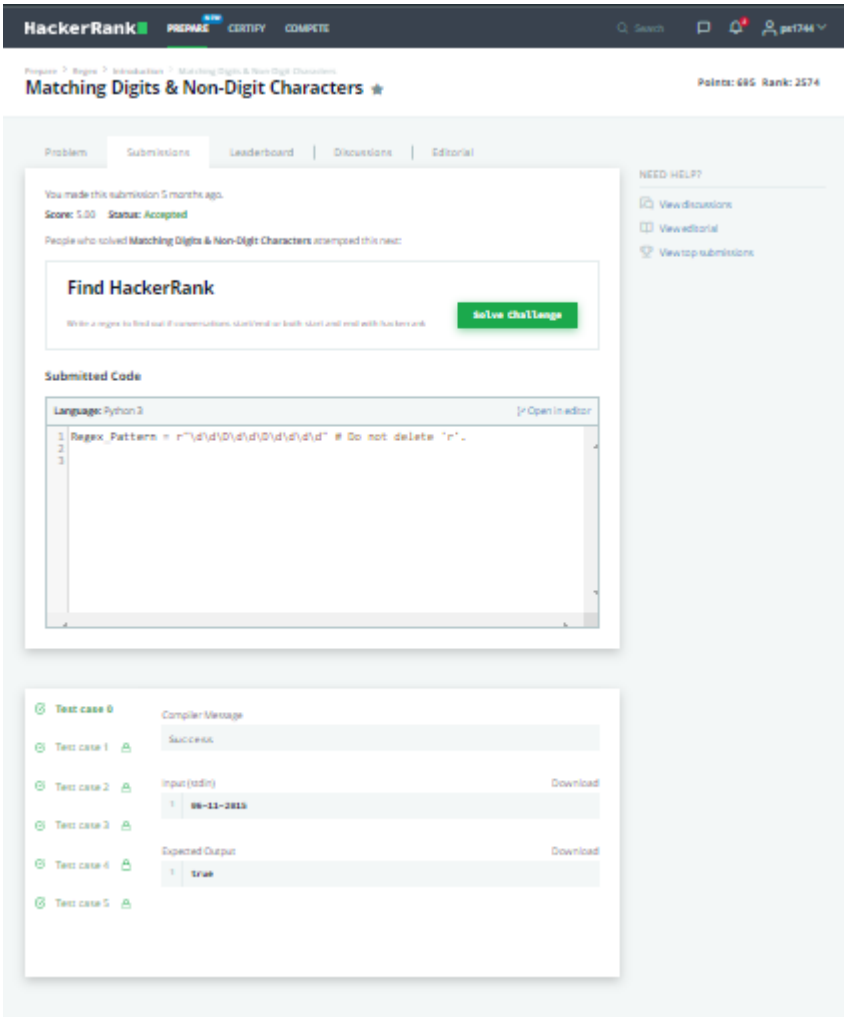
1)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Specific String
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 5.00 Status: Accepted
People who solved Matching Specific String attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
1 Regex_Pattern = r'hackerrank' # Do not delete 'r'.
2
3
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Compiler Message
Success
Input (stdin)
Download
The hackerrank team is on a mission to flatten the world by restructuring the DNA of every company on the planet. We rank programmers based on their coding skills, helping companies source great programmers and reduce the time to hire. As a result, we are revolutionizing the way companies discover and evaluate talented engineers. The hackerrank platform is the destination for the best engineers to hone their skills and companies to find top engineers.
Expected Output
Download
Number of matches : 2

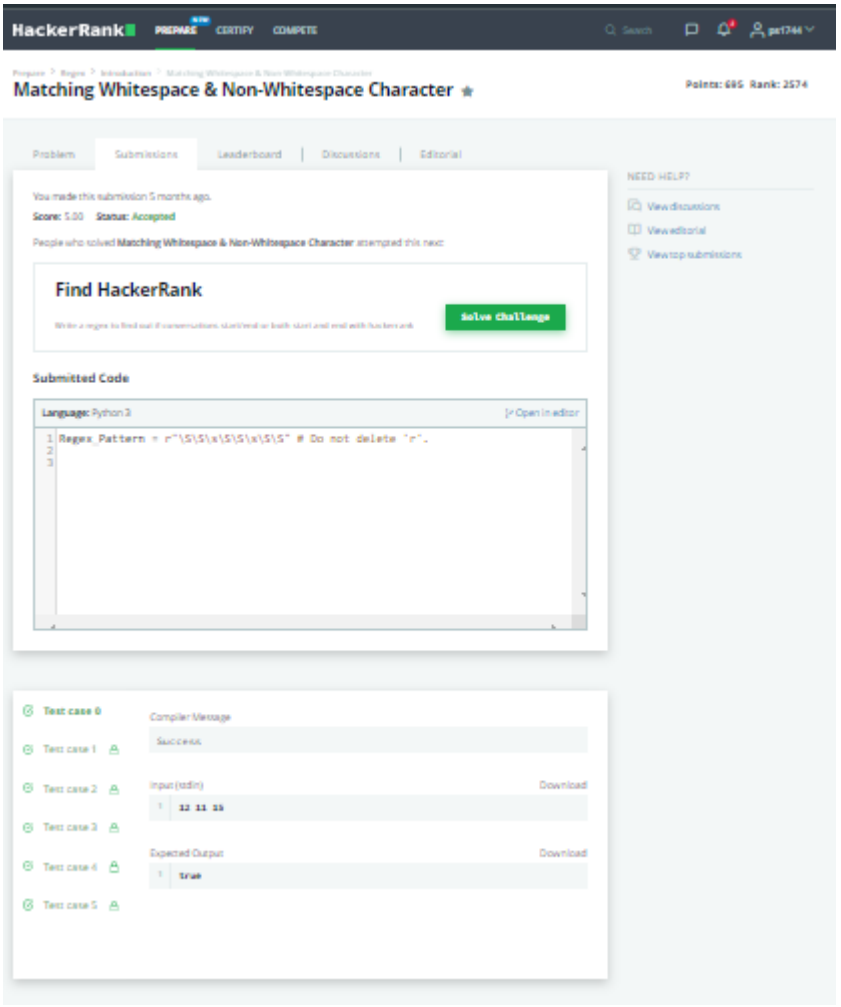
2)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Anything But a Newline
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 5.00 Status: Accepted
People who solved Matching Anything But a Newline attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
regex_pattern = r"^...\....\....\....$" # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
123.456.abc.def
Expected Output
Download
true

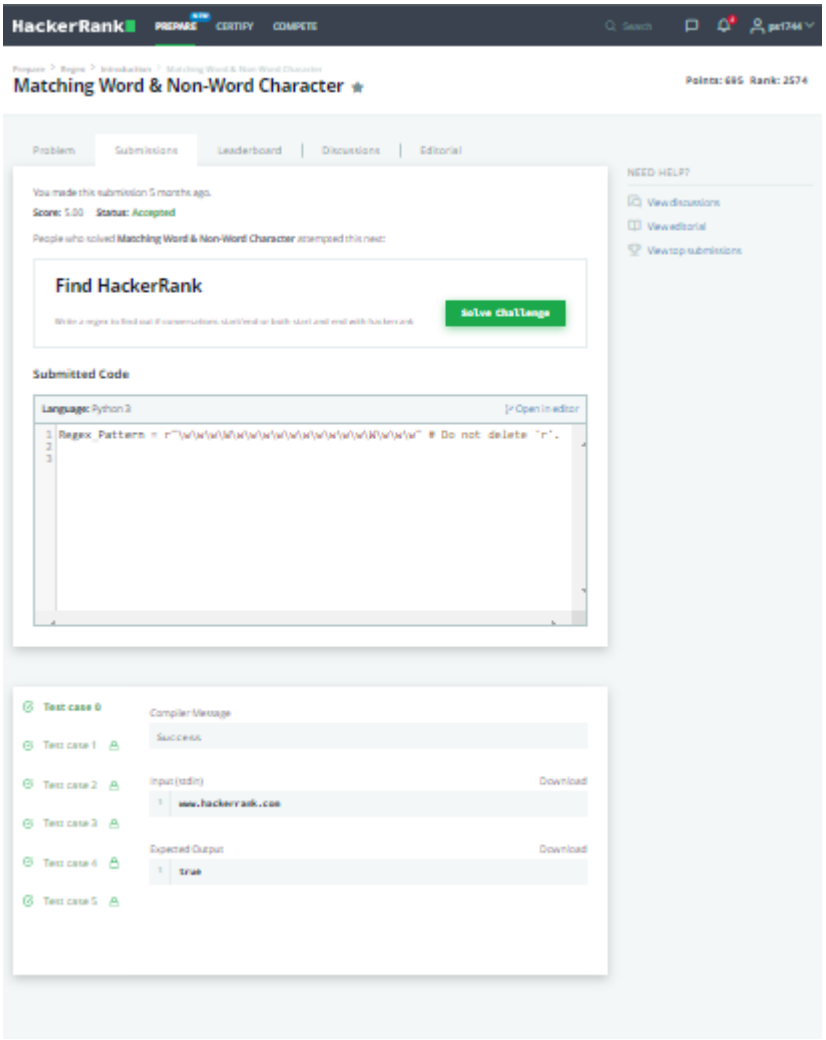
3)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Digits & Non-Digit Characters
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 5.00 Status: Accepted
People who solved Matching Digits & Non-Digit Characters attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r"\d\d\D\d\d\D\d\d\d\d" # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
Download
08-11-2015
Expected Output
Download
true

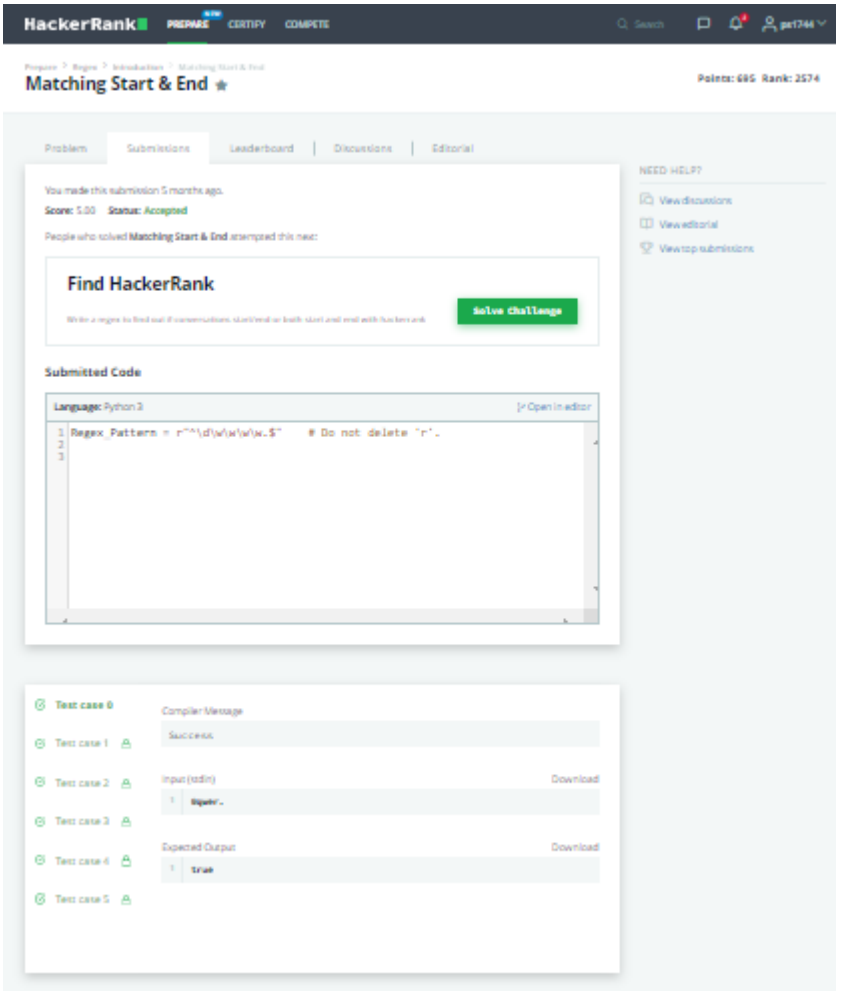
4)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Whitespace & Non-Whitespace Character
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 5.00 Status: Accepted
People who solved Matching Whitespace & Non-Whitespace Character attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
1 Regex_Pattern = r"\S\S\s\S\S\s\S\S" # Do not delete 'r'.
2
3
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
Download
1 12 11 15
Expected Output
Download
1 true

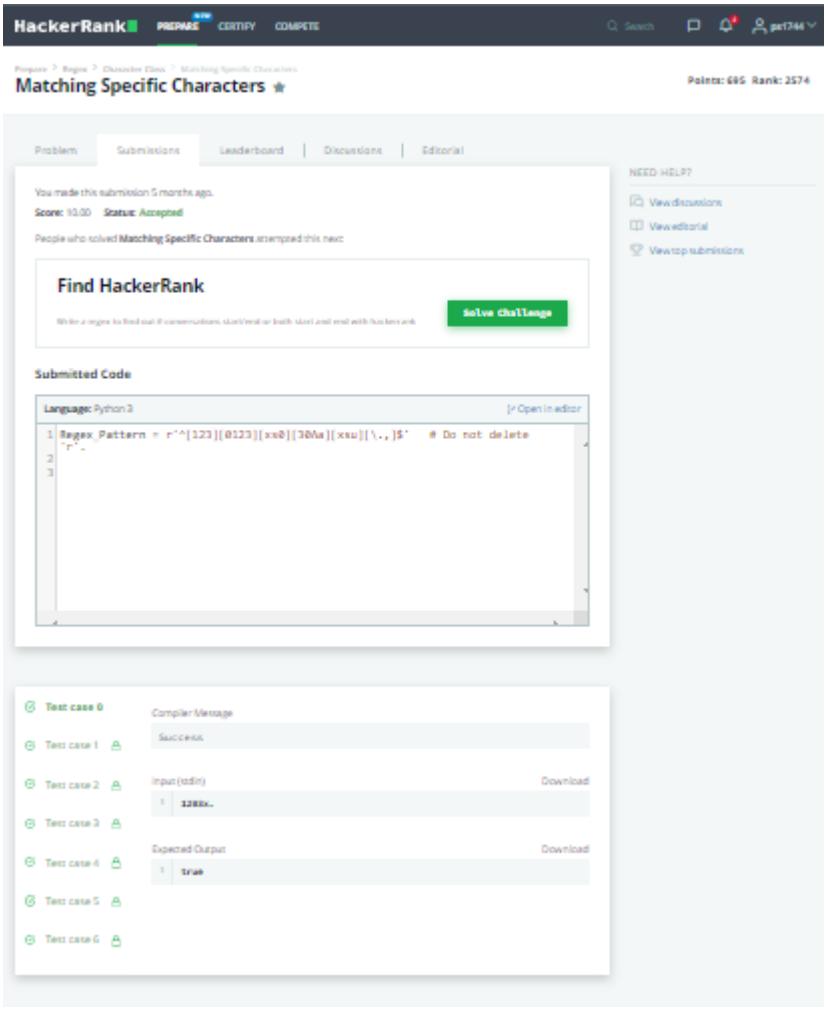
5)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Word & Non-Word Character
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 1.00 Status: Accepted
People who solved Matching Word & Non-Word Character attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r"^\w\w\w\W\w\w\w\w\w\w\w\w\w\W\w\w$" # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
Download
www.hackerrank.com
Expected Output
Download
true

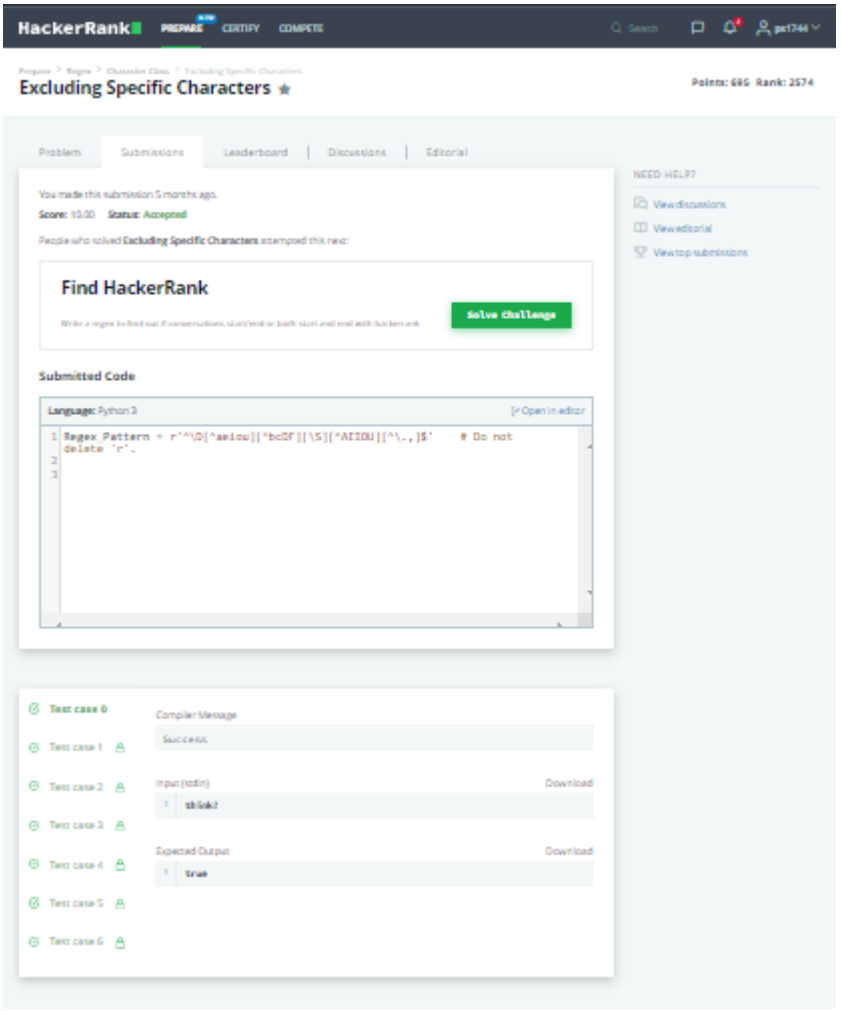
6)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Start & End
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 5.00 Status: Accepted
People who solved Matching Start & End attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r"^\d{1}[a-z]{3}$" # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
Download
Expected Output
Download
true

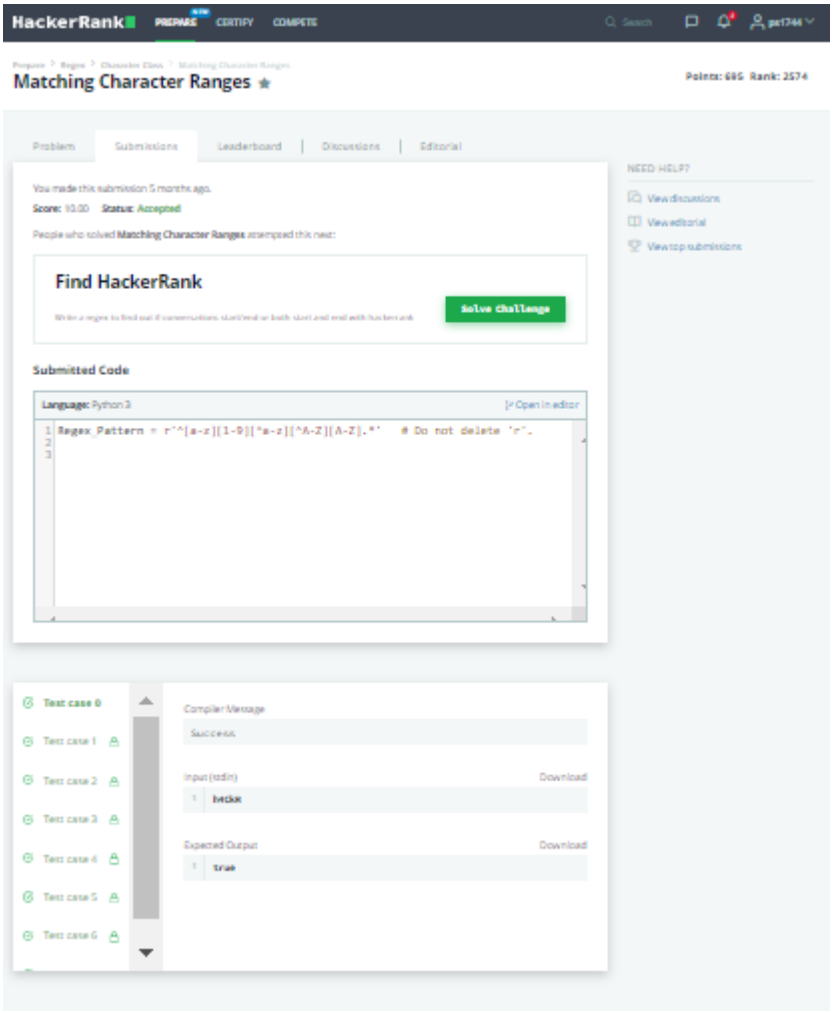
7)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Specific Characters
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 10.00 Status: Accepted
People who solved Matching Specific Characters attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r'^[123][0123][xx0][30Aa][xxu][\.,]$' # Do not delete
'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
Expected Output
Download
true

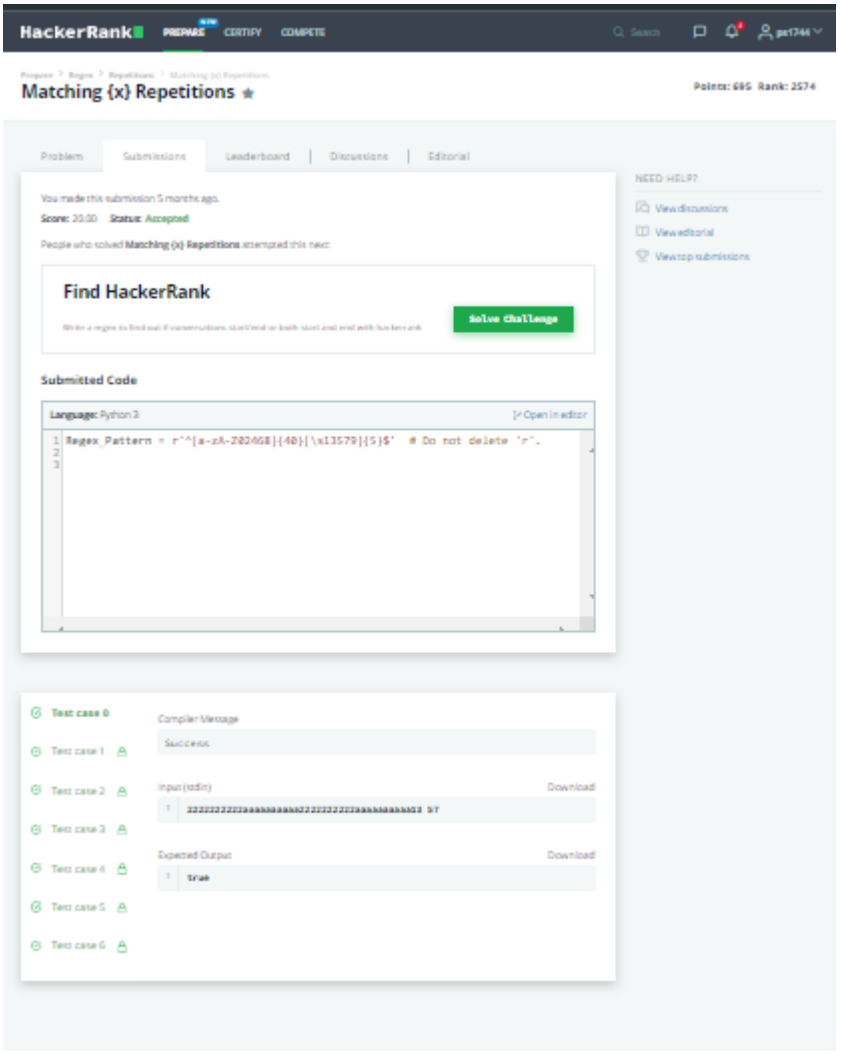
8)


HackerRank
PREPARE
CERTIFY
COMPETE
Excluding Specific Characters
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
NEED HELP?
View discussions
View editorial
View top submissions
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r'^\D[^aeiou][^bcDF][\S][^AEIOU][^\.,]$'
delete "r'_
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
think2
Expected Output
Download
true

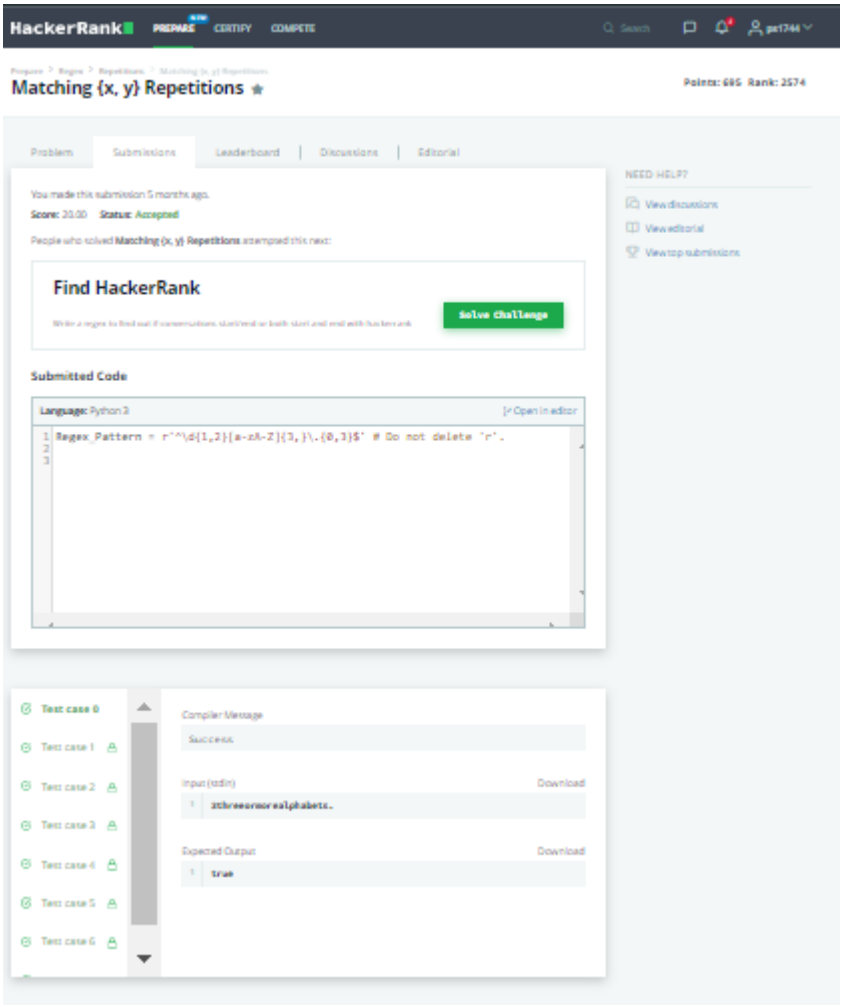
9)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Character Ranges
Points: 685 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 10.00 Status: Accepted
People who solved Matching Character Ranges attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r"^[a-z][1-9][a-z][^A-Z][A-Z].*" # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
hrckk
Expected Output
Download
true

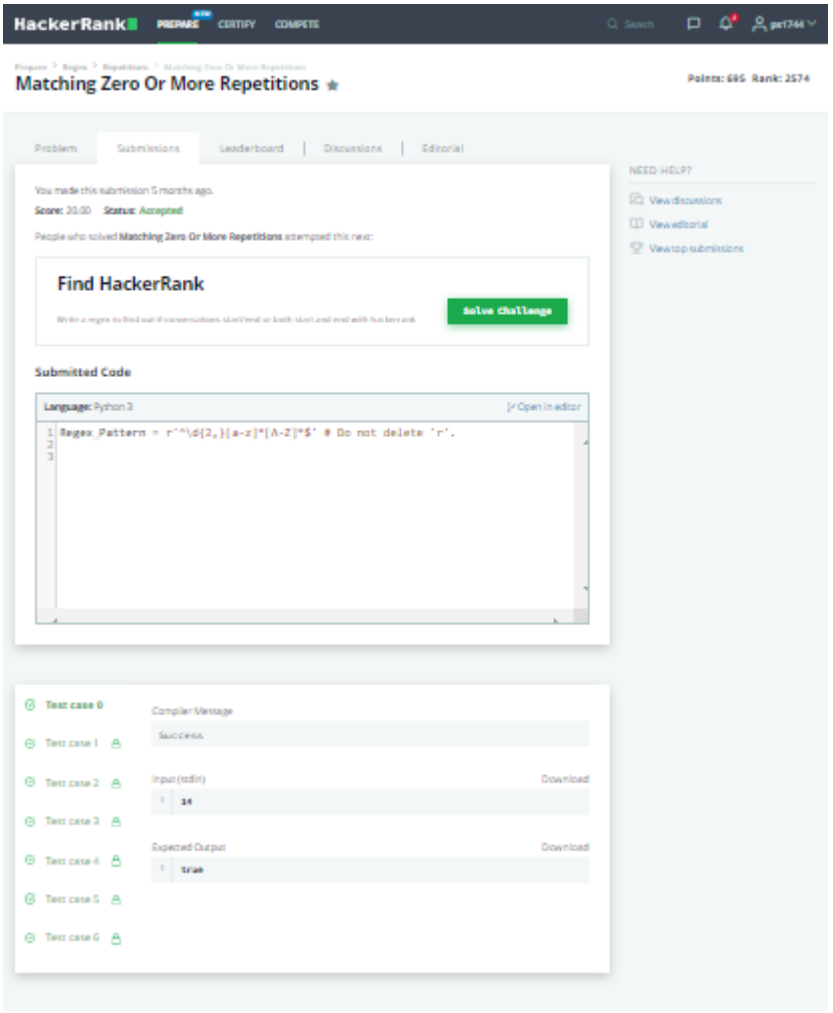
10)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching {x} Repetitions
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Matching {x} Repetitions attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r'^[a-zA-Z0246B]{40}|\x13579|{5}$' # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
Expected Output
Download
true

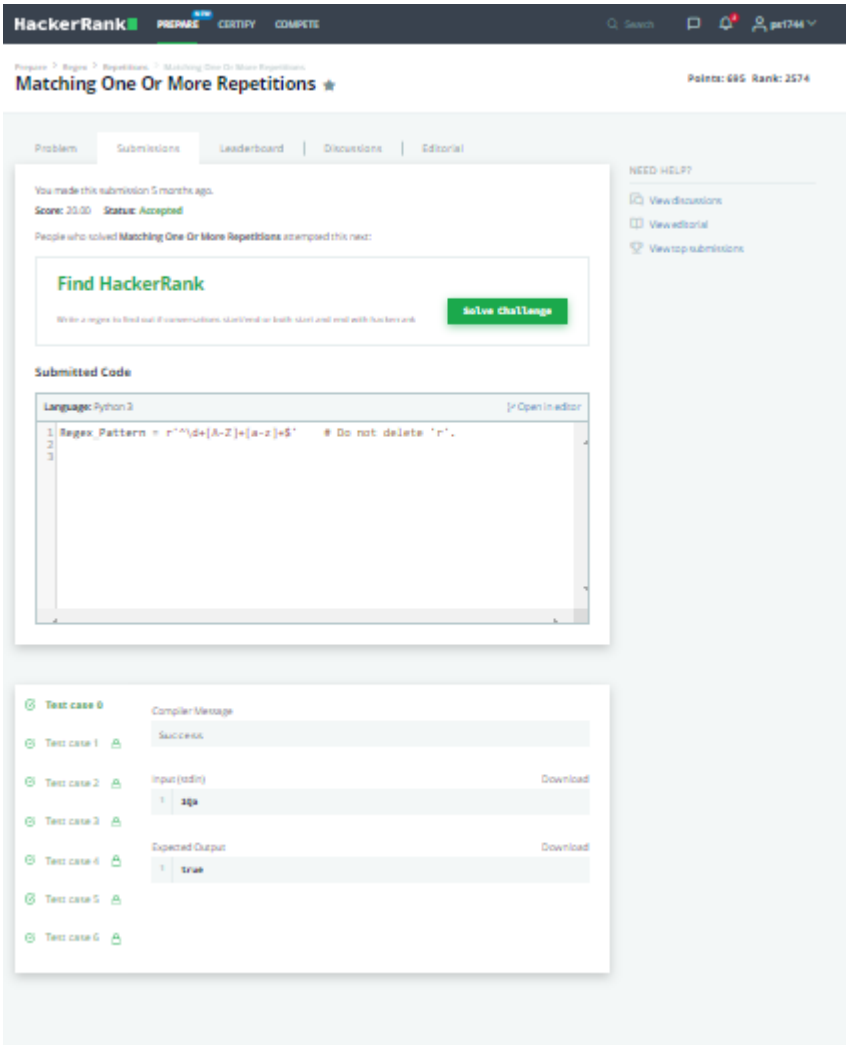
11)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching {x, y} Repetitions
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Matching {x, y} Repetitions attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r'^\d{1,2}[a-zA-Z]{3,}\.{0,3}$' # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
1 2threeormorealphabets.
Expected Output
Download
1 true

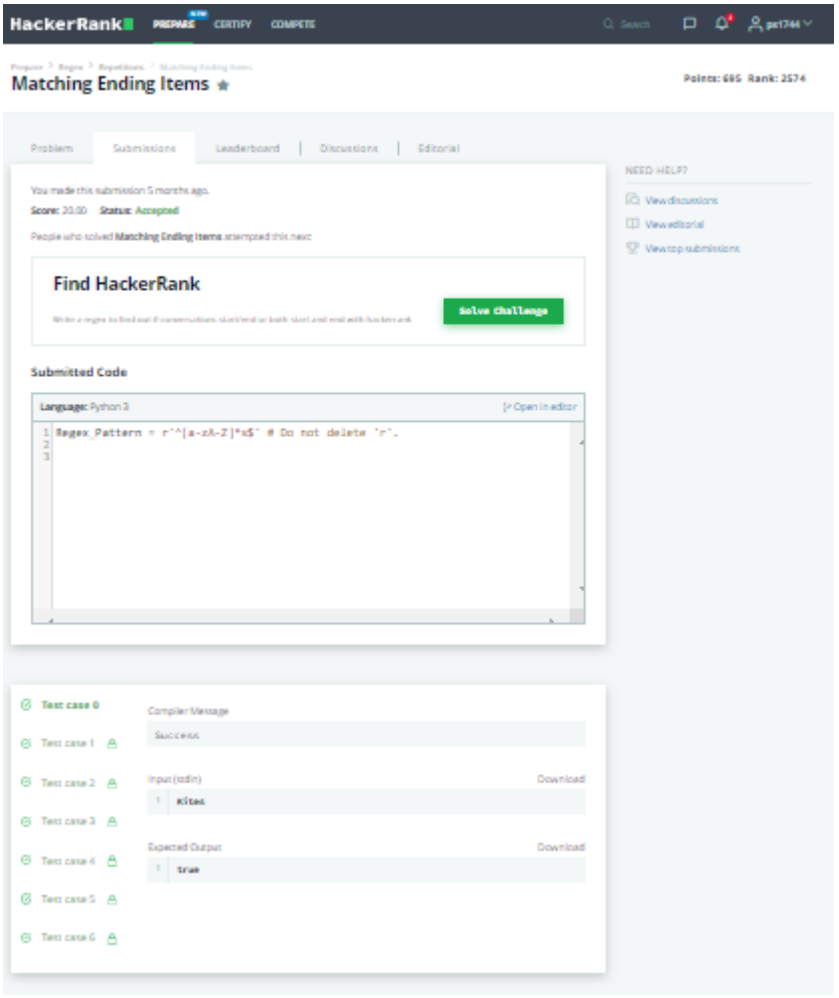
12)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Zero Or More Repetitions
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Matching Zero Or More Repetitions also solved this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r'^\d{2,}[a-z]*[A-Z]*$' # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
14
Expected Output
Download
true

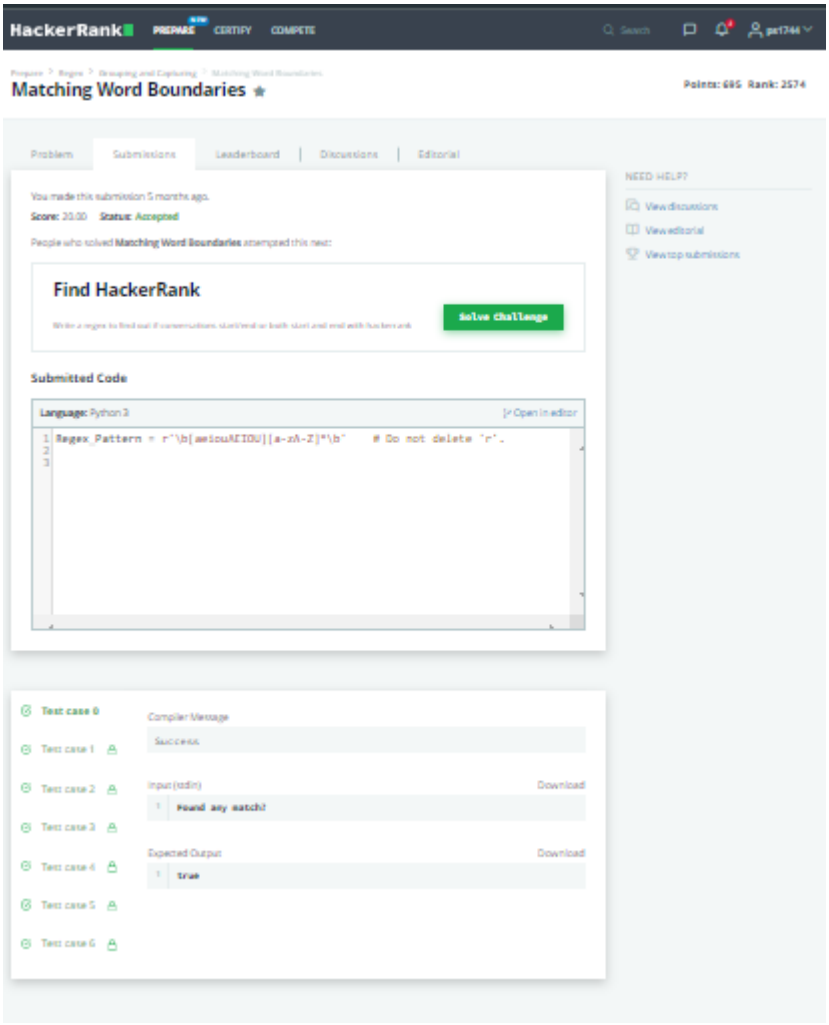
13)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching One Or More Repetitions
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Matching One Or More Repetitions attempted this next:
Find HackerRank
Solve Challenge
NEED HELP?
View discussions
View editorial
View top submissions
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r'^\d+[A-Z]+[a-z]+$' # Do not delete 'r'.
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
1 aga
Expected Output
Download
1 true

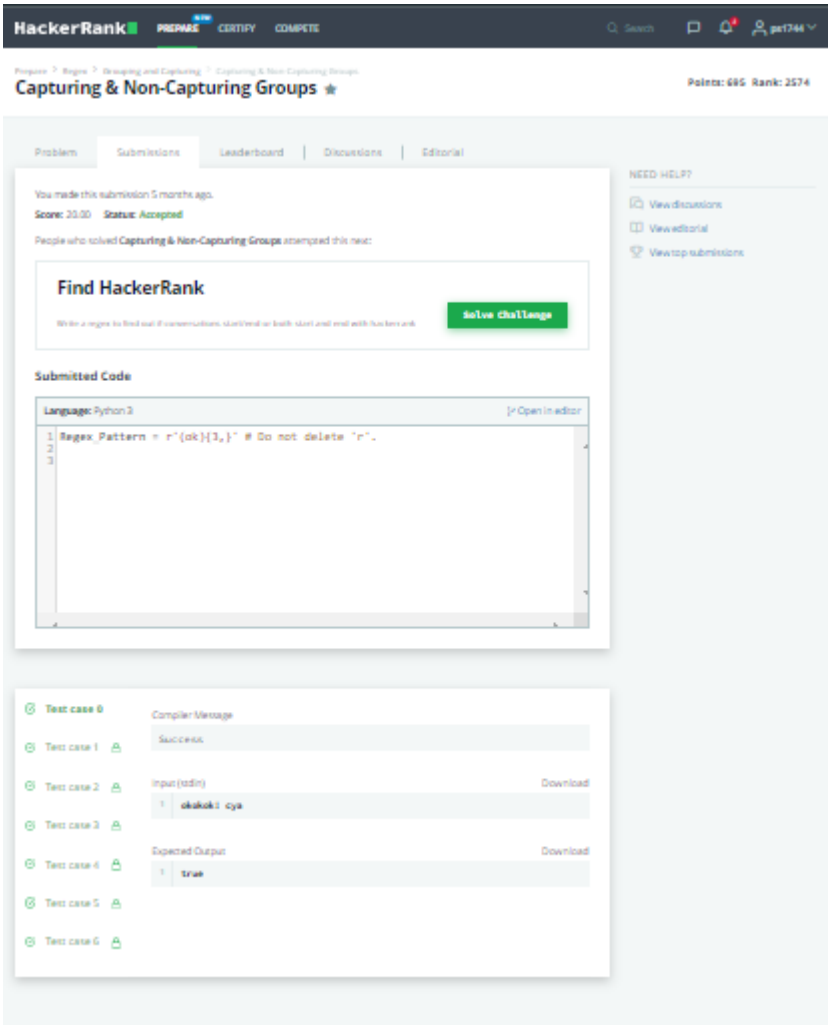
14)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Ending Items
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r"^[a-zA-Z]*s$" # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Compiler Message
Success
Input (stdin)
Download
Expected Output
Download
true

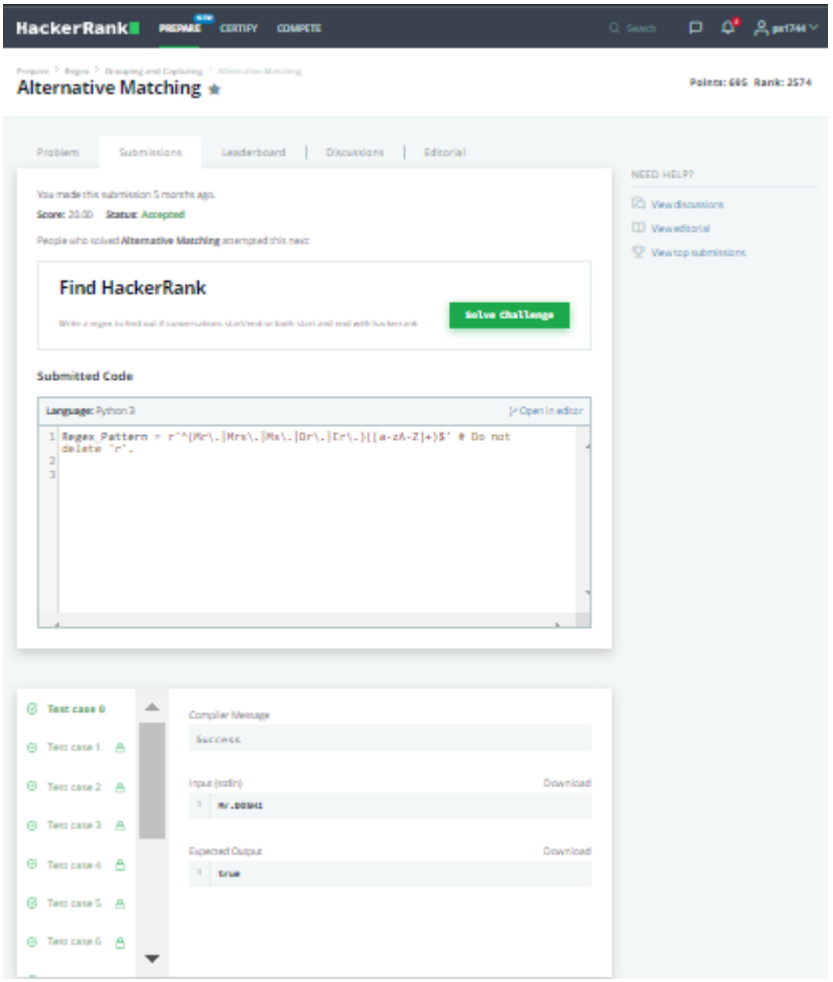
15)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Word Boundaries
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Matching Word Boundaries attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r"\b[aeiouAEIOU][a-zA-Z]*\b" # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
Found any match!
Expected Output
Download
true

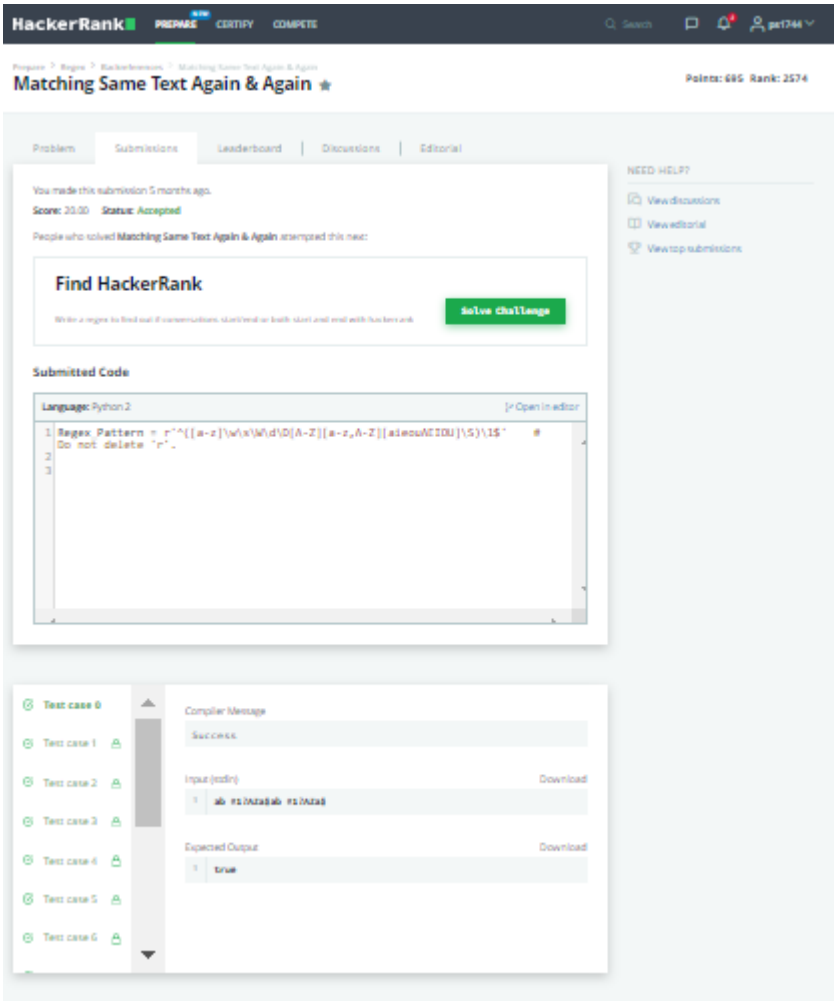
16)


HackerRank
PREPARE
CERTIFY
COMPETE
Capturing & Non-Capturing Groups
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Capturing & Non-Capturing Groups attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r'(ok){3,}' # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
okokok1 cya
Expected Output
Download
true

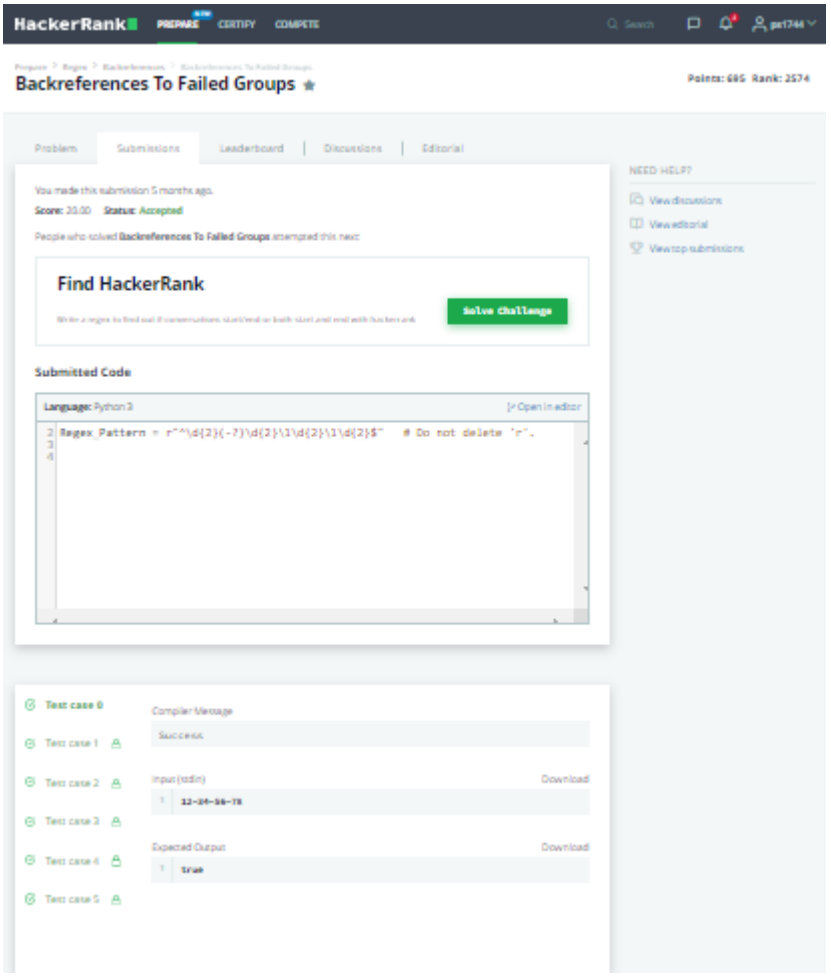
17)


HackerRank
PREPARE
CERTIFY
COMPETE
Alternative Matching
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Alternative Matching attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
1 Regex_Pattern = r'^(Mr\.|Mrs\.|Ms\.|Dr\.|Er\.){([a-zA-Z]+)$' # Do not
delete 'r'.
2
3
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
1 Mr.DUSHI
Expected Output
Download
1 true

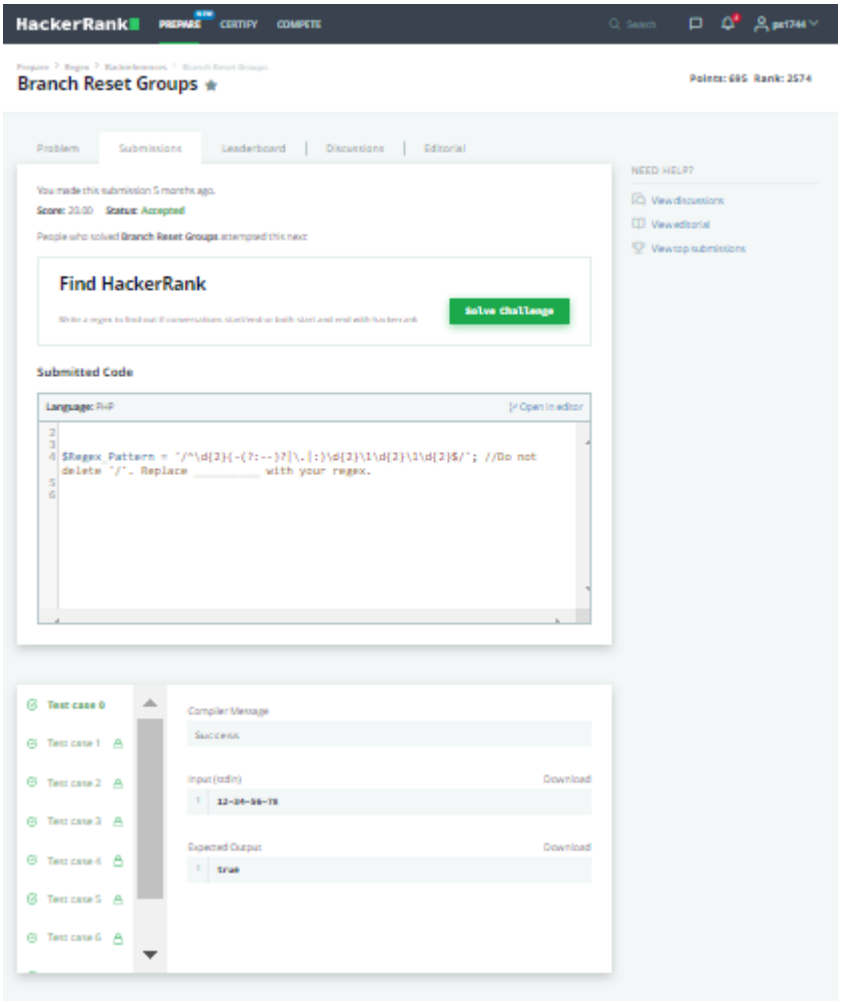
18)


HackerRank
PREPARE
CERTIFY
COMPETE
Matching Same Text Again & Again
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Matching Same Text Again & Again also solved this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 2
Open in editor
Regex_Pattern = r'^([a-z]\w\x\W\d\D[A-Z][a-z,A-Z][aieouAEIOU]\S)\1$'
Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
Expected Output
Download
true

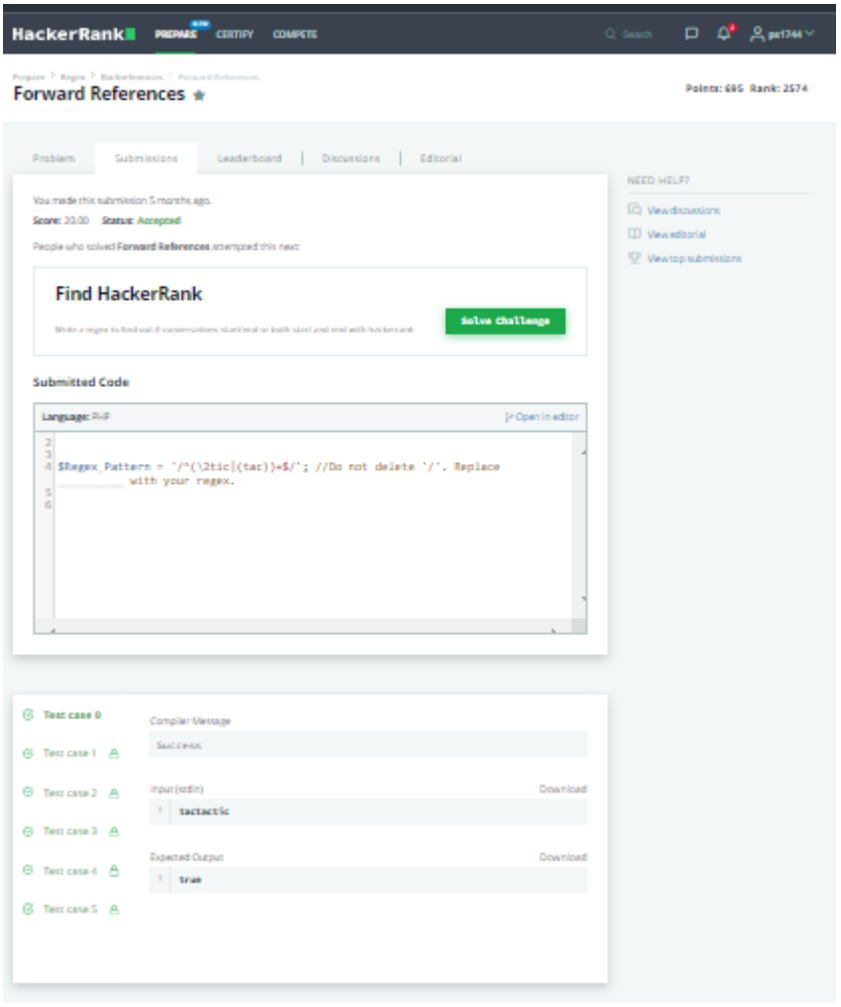
19)


HackerRank
PREPARE
CERTIFY
COMPETE
Backreferences To Failed Groups
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Backreferences To Failed Groups also solved this next:
Find HackerRank
Solve Challenge
NEED HELP?
View discussions
View editorial
View top submissions
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r"^\d{2}(-?)\d{2}\1\d{2}\1\d{2}$" # Do not delete 'r'.
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
Download
Expected Output
Download
true

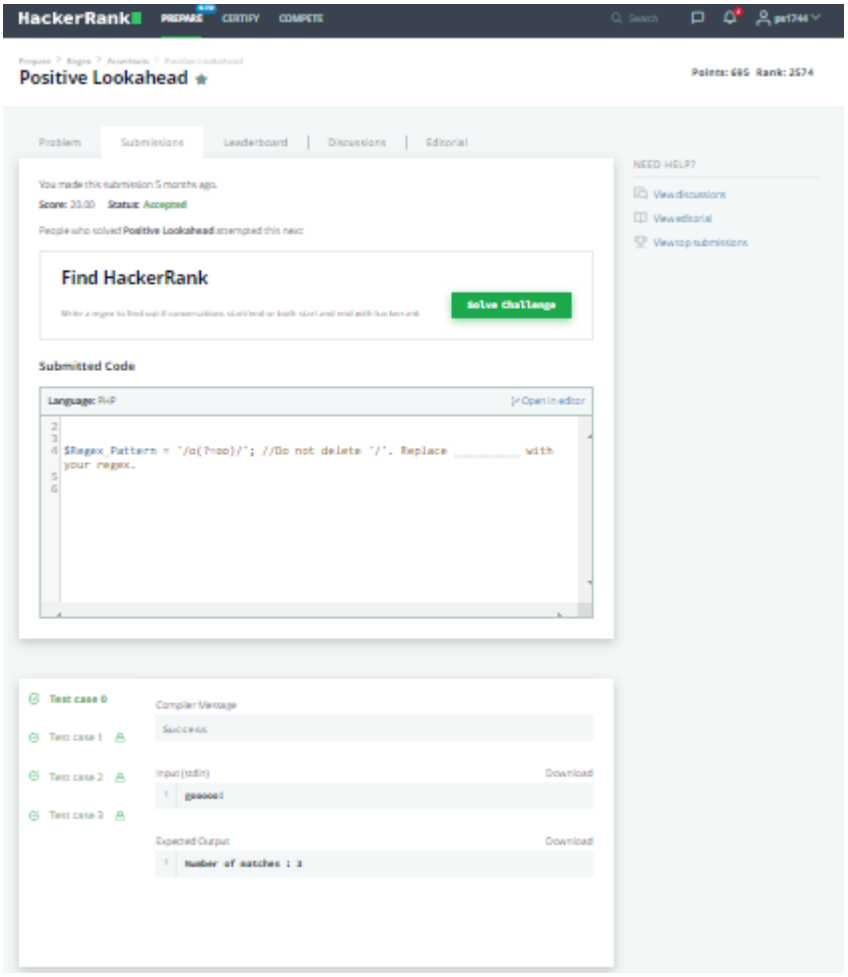
20)


HackerRank
PREPARE
CERTIFY
COMPETE
Branch Reset Groups
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Branch Reset Groups attempted this next
Find HackerRank
Solve Challenge
Submitted Code
Language: PHP
Open in editor
$Regex_Pattern = '/^\d{2}(-(?:--)?|\.|:)\d{2}\1\d{2}\1\d{2}$/'; //Do not delete '/'. Replace _____ with your regex.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
12-24-56-78
Expected Output
Download
true

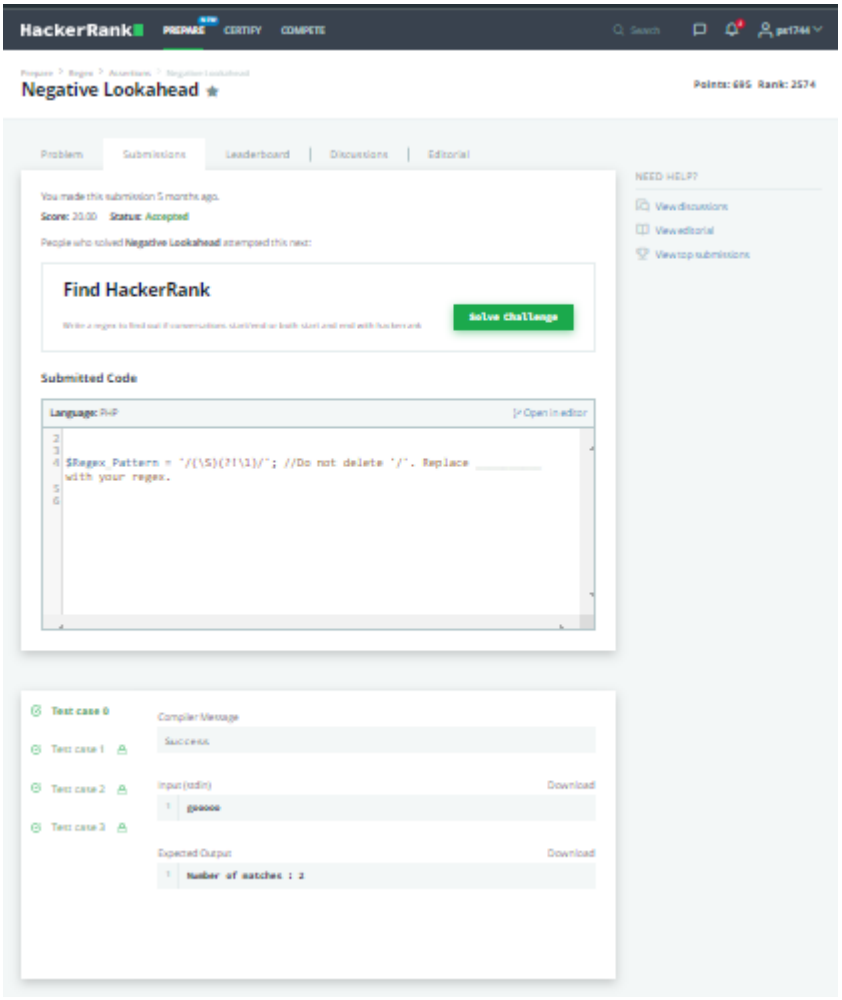
21)


HackerRank
PREPARE
CERTIFY
COMPETE
Forward References
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Forward References attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: PHP
Open in editor
$Regex_Pattern = '/^(\2tic|(tac))+$/'; //Do not delete '/'. Replace
with your regex.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
Download
tactactic
Expected Output
Download
true

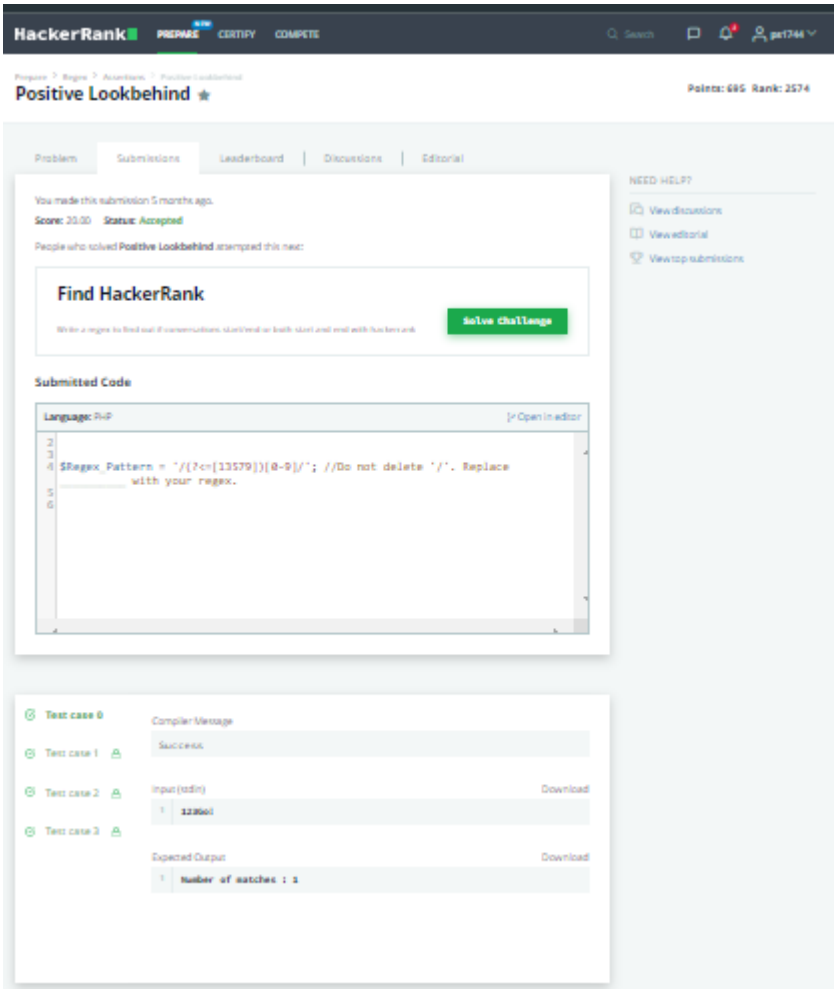
22)


HackerRank
PREPARE
CERTIFY
COMPETE
Positive Lookahead
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
Find HackerRank
Solve Challenge
Submitted Code
Language: PHP
Open in editor
$Regex_Pattern = '/a(?=oo)/'; //Do not delete '/'. Replace _____ with your regex.
Test case 0
Test case 1
Test case 2
Test case 3
Compiler Message
Success
Input (stdin)
Download
Expected Output
Download
Number of matches : 1
NEED HELP?
View discussions
View editorial
View top submissions

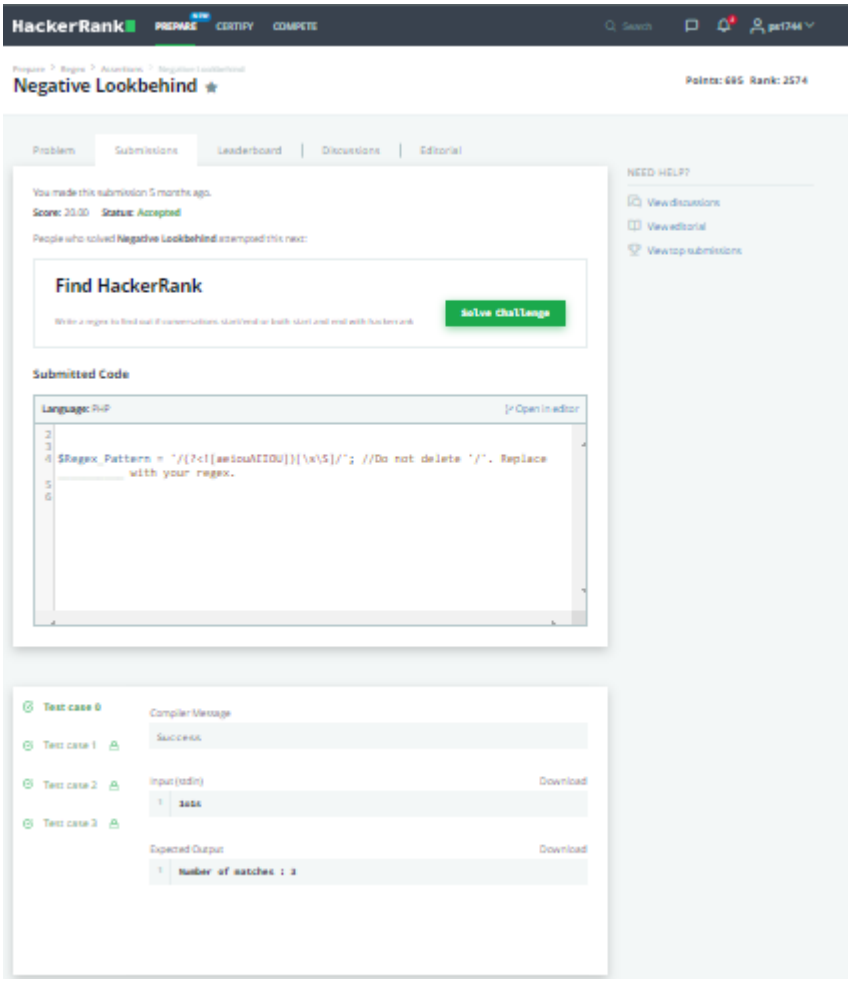
23)


HackerRank
PREPARE
CERTIFY
COMPETE
Negative Lookahead
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Negative Lookahead attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: PHP
Open in editor
$Regex_Pattern = '/(\S)(?!\1)/'; //Do not delete '/'. Replace
with your regex.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Compiler Message
Success
Input (stdin)
Download
gaaaaa
Expected Output
Download
Number of matches : 2

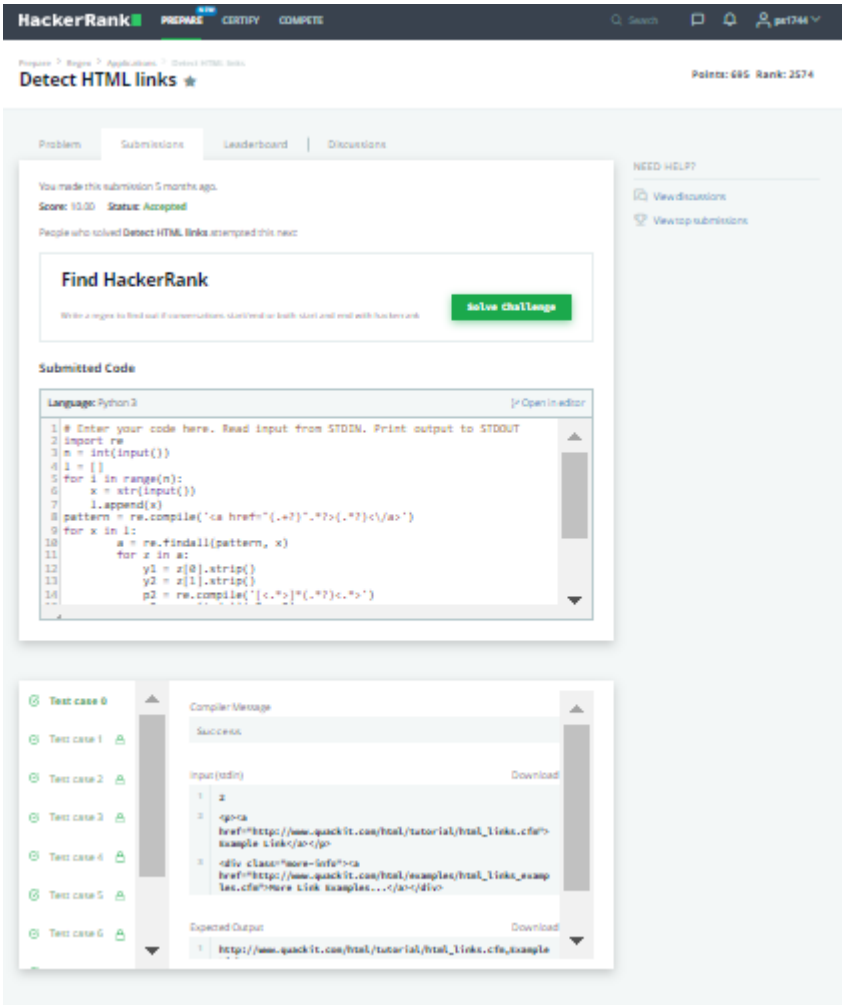
24)


HackerRank
PREPARE
CERTIFY
COMPETE
Positive Lookbehind
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Positive Lookbehind attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: PHP
Open in editor
$Regex_Pattern = '/(?<=[13579])[0-9]/'; //Do not delete '/'. Replace
with your regex.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Compiler Message
Success
Input (stdin)
Download
123bai
Expected Output
Download
Number of matches : 1

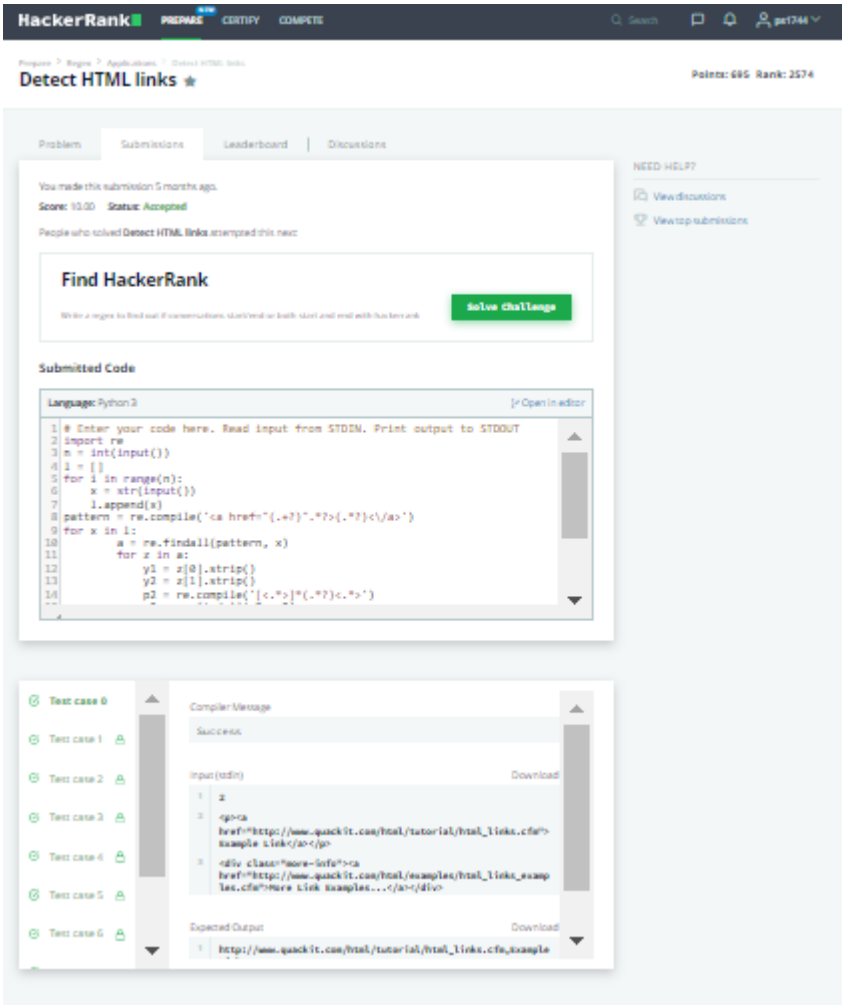
25)


HackerRank
PREPARE
CERTIFY
COMPETE
Negative Lookbehind
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Negative Lookbehind attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: PHP
Open in editor
$Regex_Pattern = '/(?<![aeiouAEIOU])[\x\S]/'; //Do not delete '/'. Replace with your regex.
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Compiler Message
Success
Input (stdin)
Download
3454
Expected Output
Download
Number of matches : 3

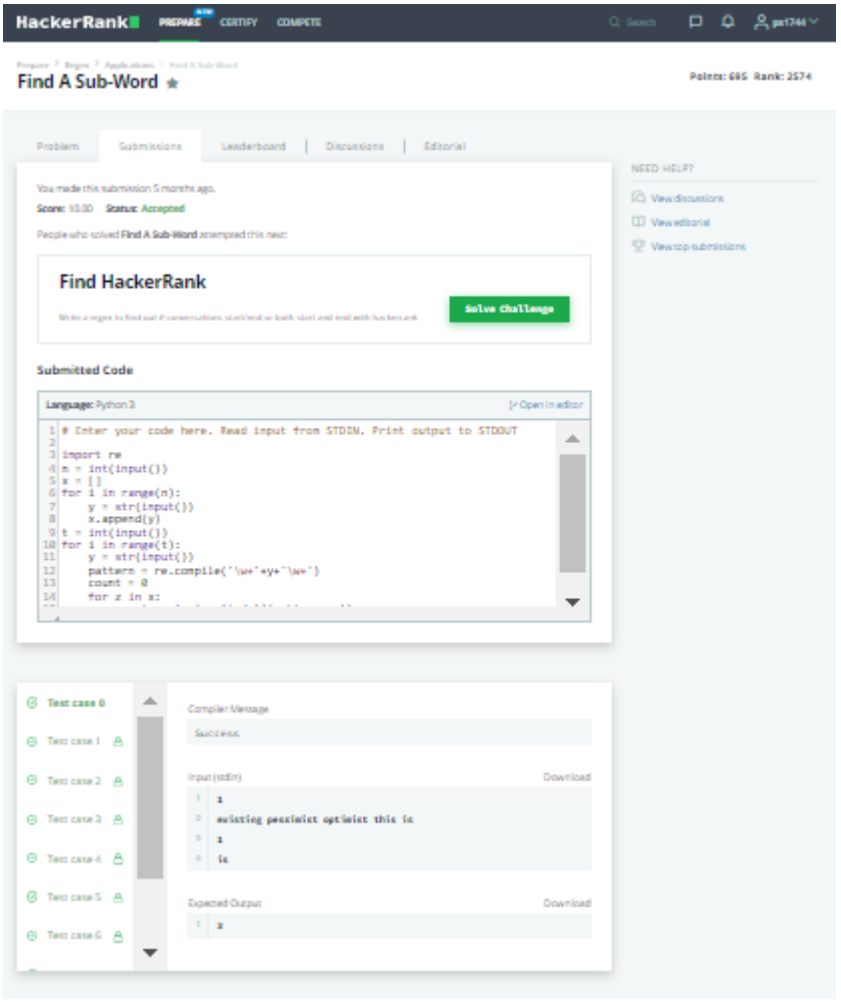
26)


HackerRank
PREPARE
CERTIFY
COMPETE
Detect HTML links
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 10.00 Status: Accepted
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
n = int(input())
l = []
for i in range(n):
x = str(input())
l.append(x)
pattern = re.compile('(.*?)<\/a>')
for x in l:
a = re.findall(pattern, x)
for z in a:
y1 = z[0].strip()
y2 = z[1].strip()
p2 = re.compile('[<.*>]*(.*?)<.*>')
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
2
<a
href="http://www.quackit.com/html/tutorial/html_links.cfm">
Example Link
<a
href="http://www.quackit.com/html/examples/html_links_examp
les.cfm">More Link Examples...
Expected Output
Download
http://www.quackit.com/html/tutorial/html_links.cfm,Example

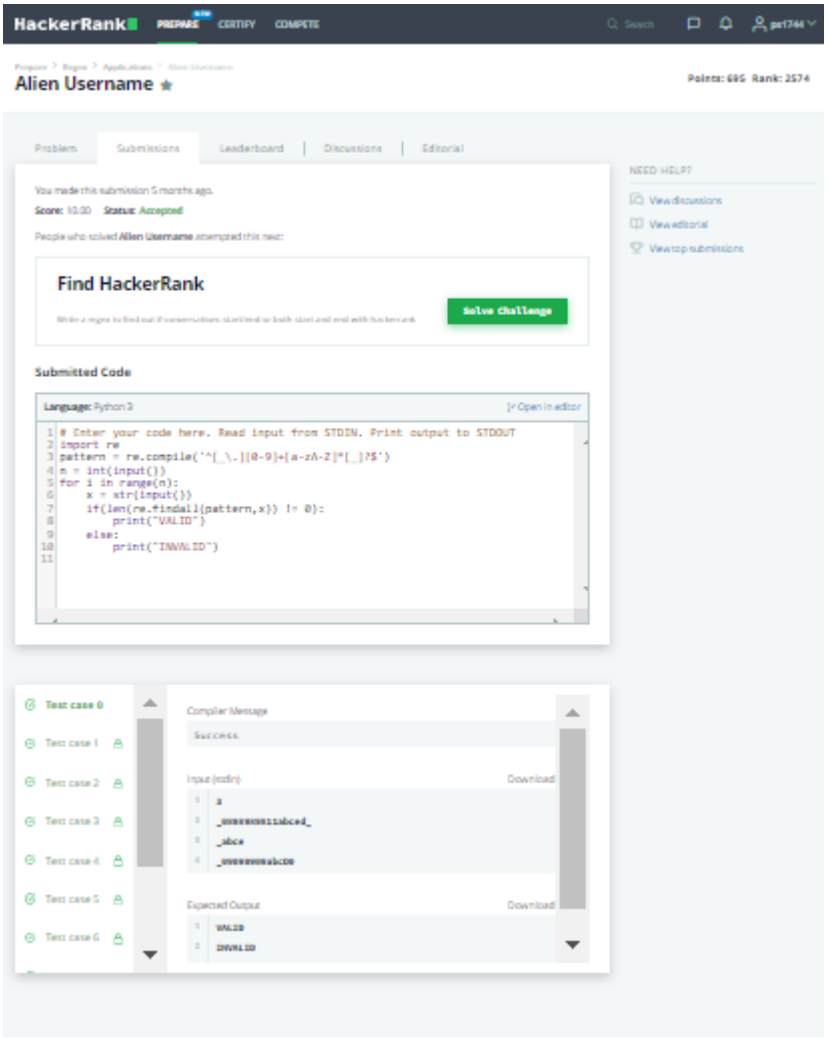
27)


HackerRank
PREPARE
CERTIFY
COMPETE
Detect HTML links
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 10.00 Status: Accepted
People who solved Detect HTML links attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
n = int(input())
l = []
for i in range(n):
x = str(input())
l.append(x)
pattern = re.compile('<a href="(.+?)".*?>(.*?)<\/a>')
for x in l:
a = re.findall(pattern, x)
for z in a:
y1 = z[0].strip()
y2 = z[1].strip()
p2 = re.compile('[<.*>]*(.*?)<.*>')
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
2
Example Link
More Link Examples...
Expected Output
Download
http://www.quackit.com/html/tutorial/html_links.cfm,Example

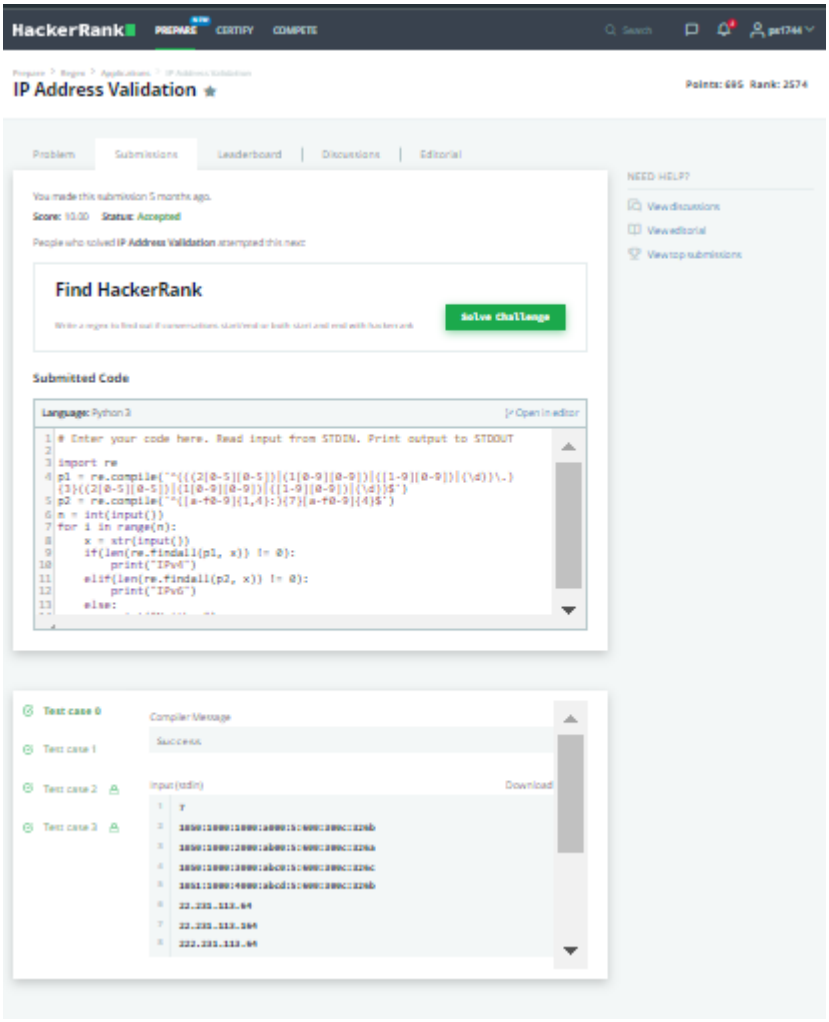
28)


HackerRank
PREPARE
CERTIFY
COMPETE
Find A Sub-Word
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 10.00 Status: Accepted
People who solved Find A Sub-Word attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
n = int(input())
x = []
for i in range(n):
y = str(input())
x.append(y)
t = int(input())
for i in range(t):
y = str(input())
pattern = re.compile(''\w+'+y+'\w+')
count = 0
for z in x:
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
1
existing pessimist optimist this is
1
is
Expected Output
Download
3

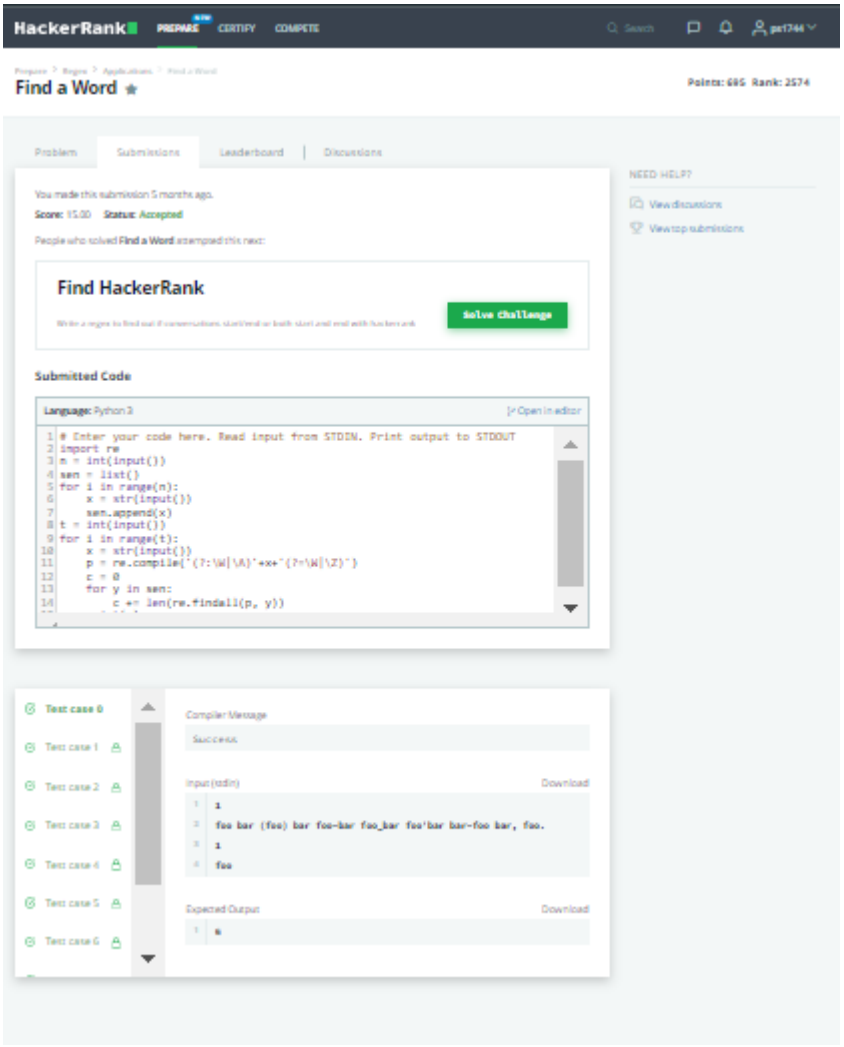
29)


HackerRank
PREPARE
CERTIFY
COMPETE
Alien Username
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 5 months ago.
Score: 10.00 Status: Accepted
People who solved Alien Username attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
pattern = re.compile('^[_\.][0-9]+[a-zA-Z]*[_]?$')
n = int(input())
for i in range(n):
x = str(input())
if(len(re.findall(pattern,x)) != 0):
print("VALID")
else:
print("INVALID")
NEED HELP?
View discussions
View editorial
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
Expected Output
Download
VALID
INVALID

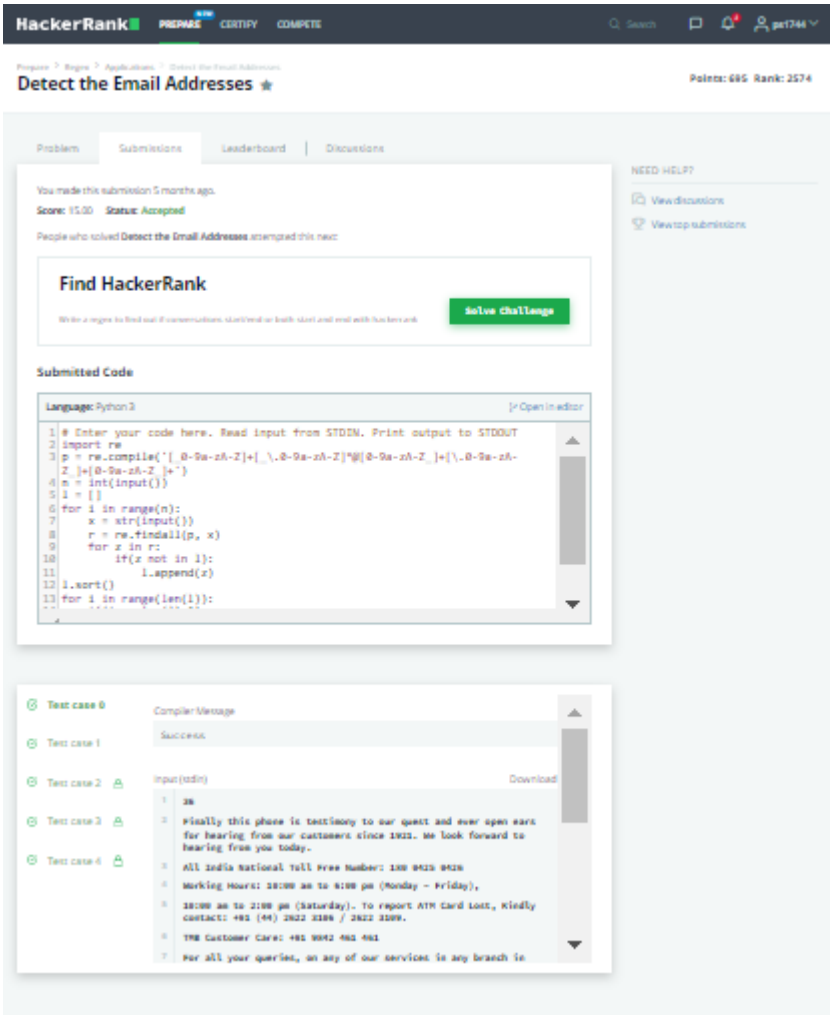
30)


HackerRank
PREPARE
CERTIFY
COMPETE
IP Address Validation
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
Editorial
NEED HELP?
View discussions
View editorial
View top submissions
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
p1 = re.compile('^(((2[0-5][0-5])|(1[0-9][0-9])|([1-9][0-9])|(\d))\.){3}((2[0-5][0-5])|(1[0-9][0-9])|([1-9][0-9])|(\d))$')
p2 = re.compile('^([a-f0-9]{1,4}:){7}[a-f0-9]{4}$')
n = int(input())
for i in range(n):
x = str(input())
if(len(re.findall(p1, x)) != 0):
print("IPv4")
elif(len(re.findall(p2, x)) != 0):
print("IPv6")
else:
Test case 0
Test case 1
Test case 2
Test case 3
Compiler Message
Success
Input (stdin)
Download
7
1050:1000:1000:1000:1:600:200c:326b
1050:1000:2000:1b00:1:600:200c:326a
1050:1000:1000:1b00:1:600:200c:326c
1051:1000:4000:1b0d:1:600:200c:326b
22.233.111.64
22.233.111.256
222.233.111.64

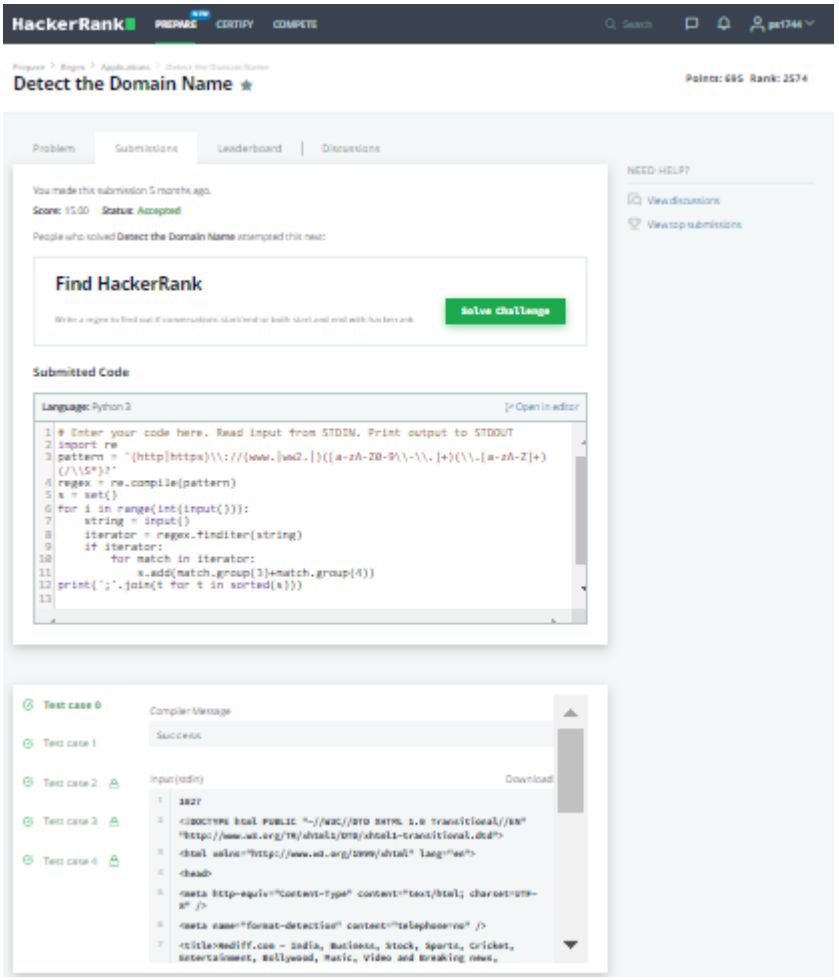
31)


HackerRank
PREPARE
CERTIFY
COMPETE
Find a Word
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 15.00 Status: Accepted
People who solved Find a Word attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
n = int(input())
sen = list()
for i in range(n):
x = str(input())
sen.append(x)
t = int(input())
for i in range(t):
x = str(input())
p = re.compile(r'(?:\W|\A)'+x+'(?=\W|\Z)')
c = 0
for y in sen:
c += len(re.findall(p, y))
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
1
foo bar (foo) bar foo-bar foo_bar foo'bar bar-foo bar, foo.
1
foo
Expected Output
Download
6

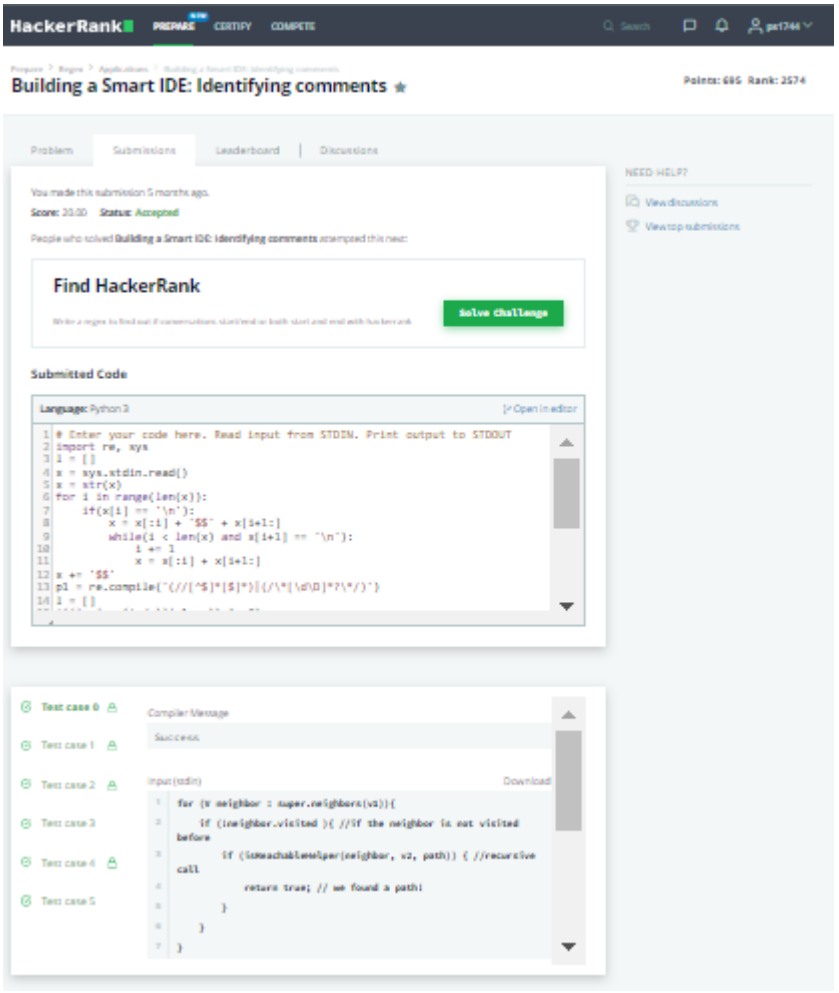
32)


HackerRank
PREPARE
CERTIFY
COMPETE
Detect the Email Addresses
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 15.00 Status: Accepted
People who solved Detect the Email Addresses attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
p = re.compile('[_0-9a-zA-Z]+|_\.0-9a-zA-Z]*@[0-9a-zA-Z_]+[\.0-9a-zA-Z_]+[0-9a-zA-Z_]+')
n = int(input())
l = []
for i in range(n):
x = str(input())
r = re.findall(p, x)
for z in r:
if(z not in l):
l.append(z)
l.sort()
for i in range(len(l)):
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Compiler Message
Success
Input (stdin)
Download
26
Finally this phone is testimony to our quest and ever open ears for hearing from our customers since 1921. We look forward to hearing from you today.
All India National Toll Free Number: 180 8425 8426
Working Hours: 10:00 am to 6:00 pm (Monday - Friday),
10:00 am to 2:00 pm (Saturday). To report ATM Card Lost, Kindly contact: +91 (44) 2822 3100 / 2822 3109.
TMB Customer Care: +91 9842 461 461
For all your queries, on any of our services in any branch in

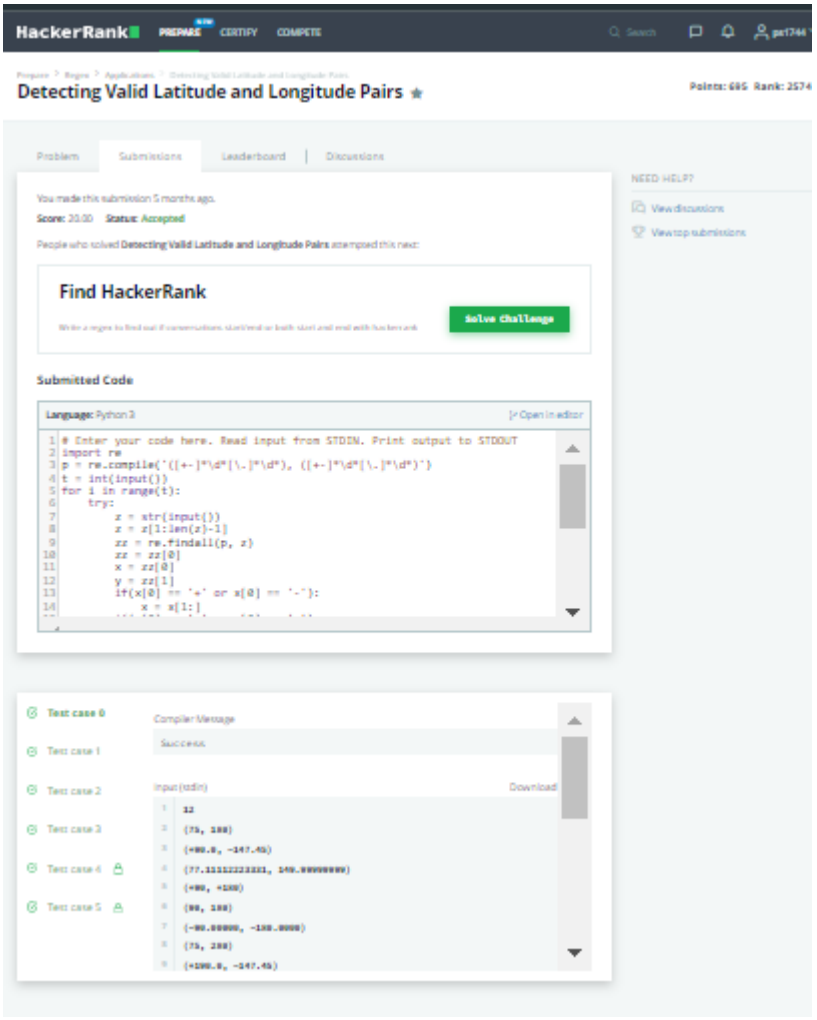
33)


HackerRank
PREPARE
CERTIFY
COMPETE
Detect the Domain Name
Points: 685 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 15.00 Status: Accepted
People who solved Detect the Domain Name attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
pattern = "(http|https)\\://(www.|ww2.|)([a-zA-Z0-9\\-\\.]+)(\\.[a-zA-Z]+)(/\\S*)?"
regex = re.compile(pattern)
s = set()
for i in range(int(input())):
string = input()
iterator = regex.finditer(string)
if iterator:
for match in iterator:
s.add(match.group(3)+match.group(4))
print(';'.join(t for t in sorted(s)))
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Compiler Message
Success
Input (stdin)
Download
1027
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="format-detection" content="telephone=no" />
<title>Rediff.com - India, Business, Stock, Sports, Cricket, Entertainment, Bollywood, Music, Video and Breaking news.</title>

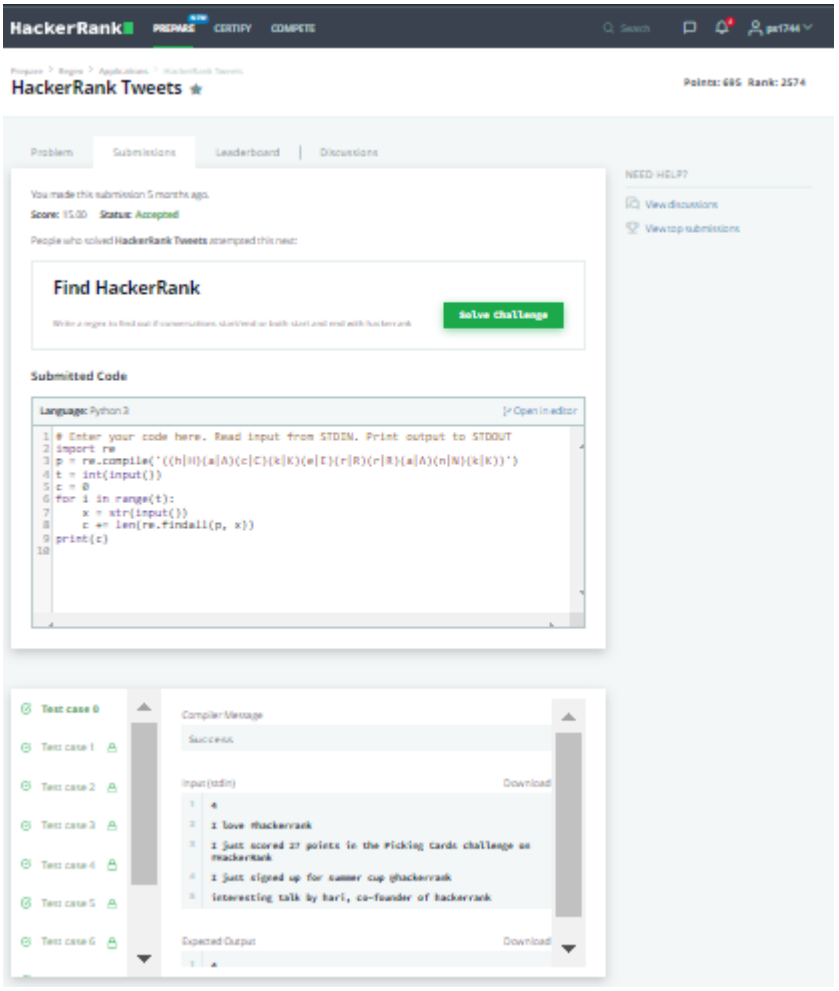
34)


HackerRank
PREPARE
CERTIFY
COMPETE
Building a Smart IDE: Identifying comments
Points: 685 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re, sys
l = []
x = sys.stdin.read()
x = str(x)
for i in range(len(x)):
if(x[i] == '\n'):
x = x[:i] + "$$" + x[i+1:]
while(i < len(x) and x[i+1] == "\n"):
i += 1
x = x[:i] + x[i+1:]
x += '$$'
p1 = re.compile("(//[^$]*|$)*|(/*[\s\S]*?*/)")
l = []
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
Download
for (V neighbor : super.neighbors(v1)){
if (!neighbor.visited){ //if the neighbor is not visited
before
if (isReachableHelper(neighbor, v2, path)) { //recursive
call
return true; // we found a path!
}
}
}

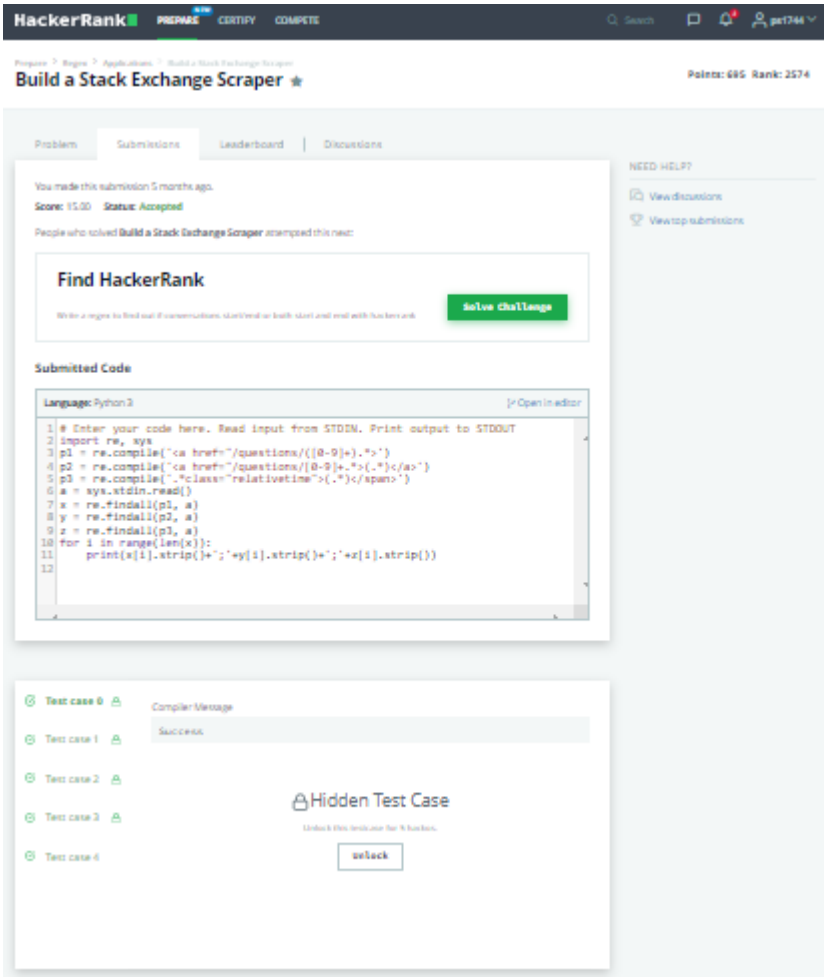
35)


HackerRank
PREPARE
CERTIFY
COMPETE
Detecting Valid Latitude and Longitude Pairs
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Detecting Valid Latitude and Longitude Pairs attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
p = re.compile('([+-]*\d*[\.]*\d*), ([+-]*\d*[\.]*\d*)')
t = int(input())
for i in range(t):
try:
x = str(input())
x = x[1:len(x)-1]
xx = re.findall(p, x)
xx = xx[0]
x = xx[0]
y = xx[1]
if(x[0] == '+' or x[0] == '-'):
x = x[1:]
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
Download
12
(75, 180)
(+90.0, -147.45)
(77.1111222331, 149.9999999)
(+90, +180)
(90, 180)
(-90.00000, -180.0000)
(75, 180)
(+190.0, -147.45)

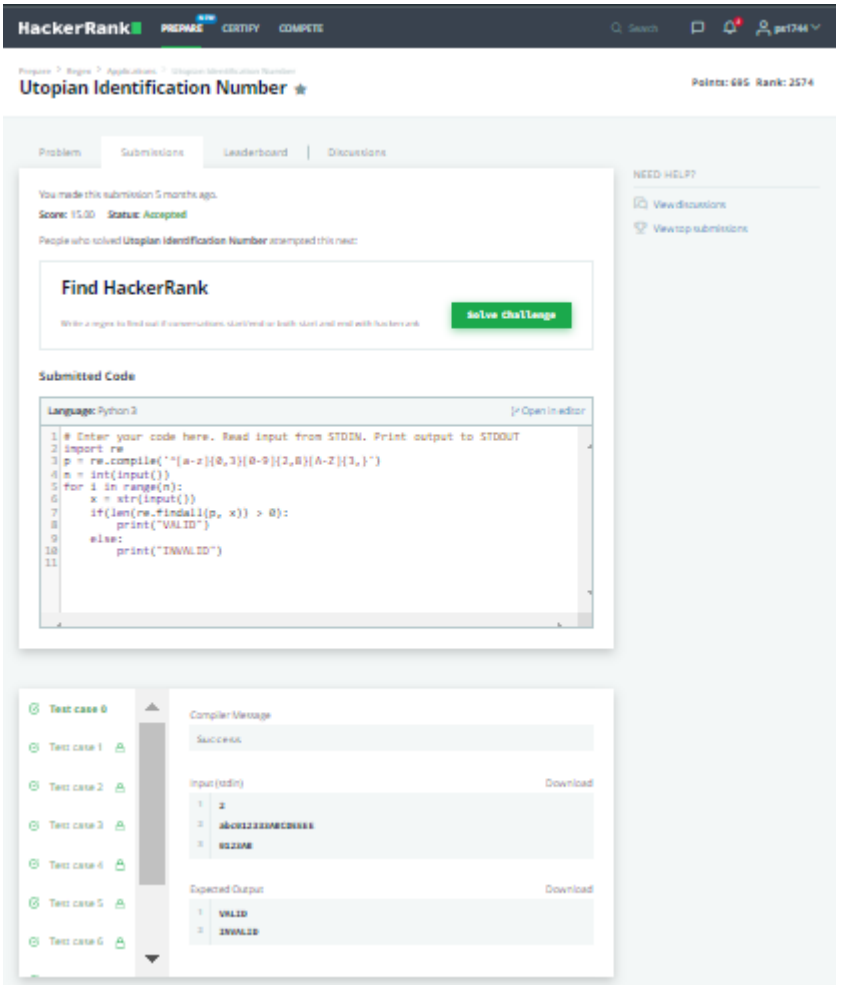
36)


HackerRank
PREPARE
CERTIFY
COMPETE
HackerRank Tweets
Points: 685 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 15.00 Status: Accepted
People who solved HackerRank Tweets attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
p = re.compile('((h|H)(a|A)(c|C)(k|K)(e|E)(r|R)(r|R)(a|A)(n|N)(k|K))')
t = int(input())
c = 0
for i in range(t):
x = str(input())
c += len(re.findall(p, x))
print(c)
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
4
I love #hackerrank
I just scored 27 points in the Picking Cards challenge on #hackerRank
I just signed up for summer cup @hackerrank
interesting talk by Hari, co-founder of hackerrank
Expected Output
Download

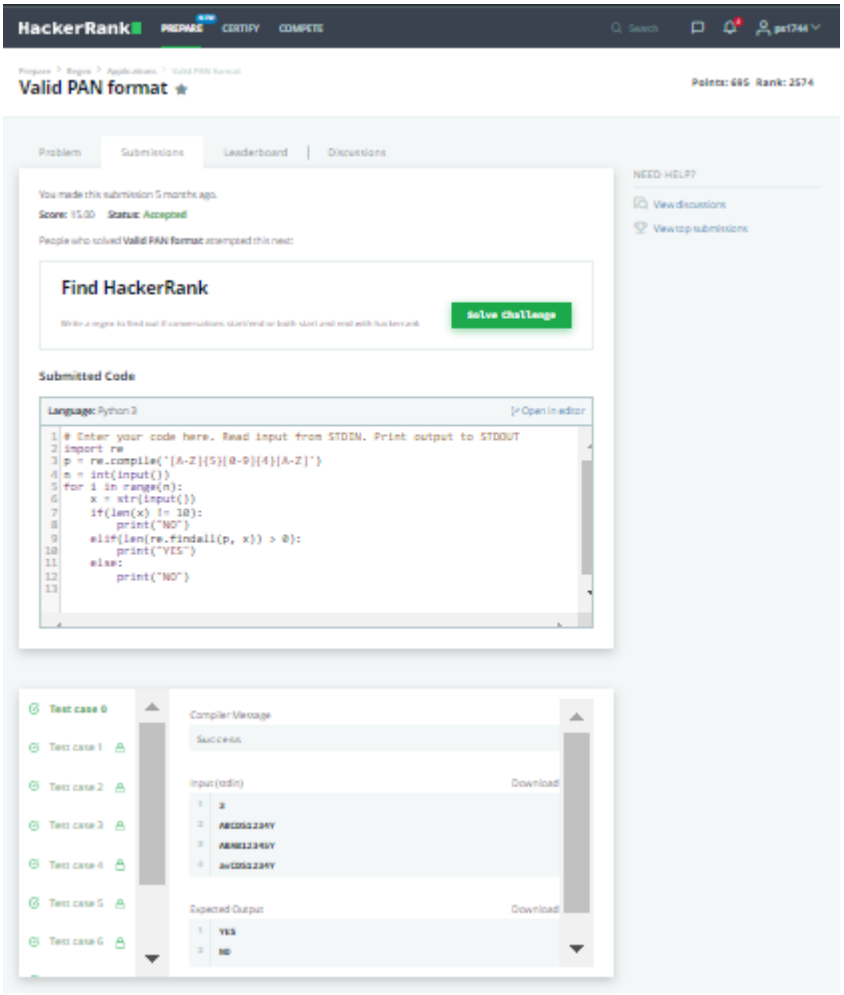
37)


HackerRank
PREPARE
CERTIFY
COMPETE
Build a Stack Exchange Scraper
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 15.00 Status: Accepted
People who solved Build a Stack Exchange Scraper attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re, sys
p1 = re.compile('<a href="/questions/([0-9]+).*>')
p2 = re.compile('(.*)')
p3 = re.compile('."class="relativetime">(.*)')
s = sys.stdin.read()
x = re.findall(p1, s)
y = re.findall(p2, s)
z = re.findall(p3, s)
for i in range(len(x)):
print(x[i].strip()+';'+y[i].strip()+';'+z[i].strip())
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Compiler Message
Success
Hidden Test Case
unlock

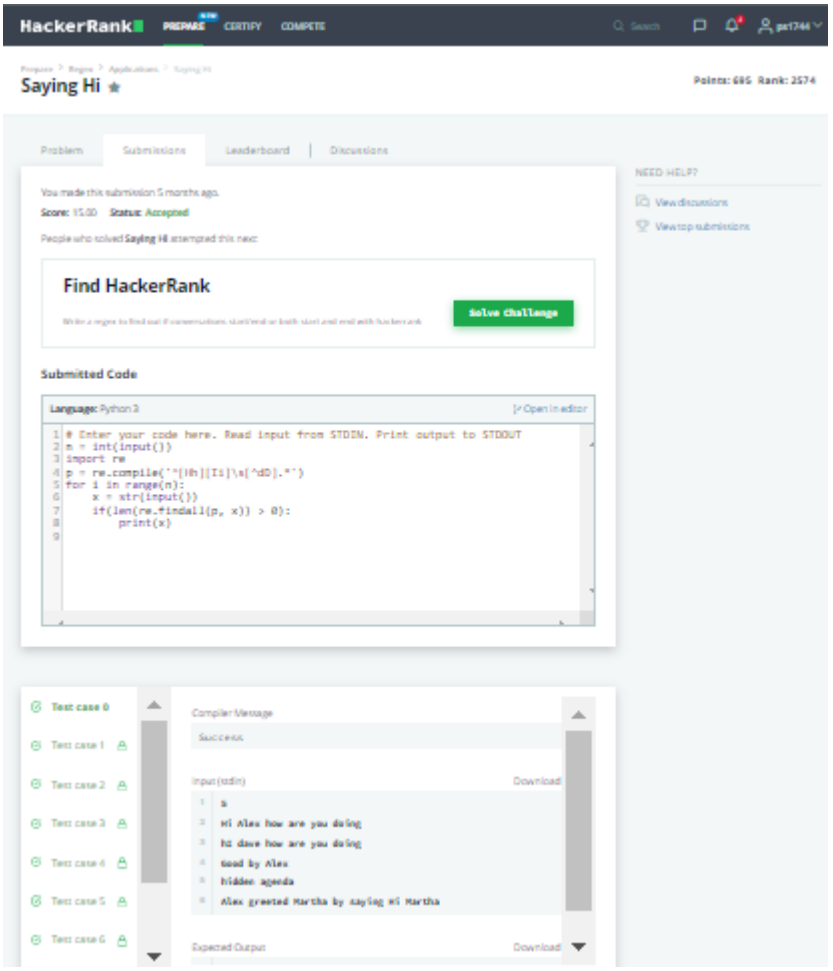
38)


HackerRank
PREPARE
CERTIFY
COMPETE
Utopian Identification Number
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 15.00 Status: Accepted
People who solved Utopian Identification Number attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
p = re.compile('^[a-z]{0,3}[0-9]{2,8}[A-Z]{3,}')
n = int(input())
for i in range(n):
x = str(input())
if(len(re.findall(p, x)) > 0):
print("VALID")
else:
print("INVALID")
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
Expected Output
Download
VALID
INVALID

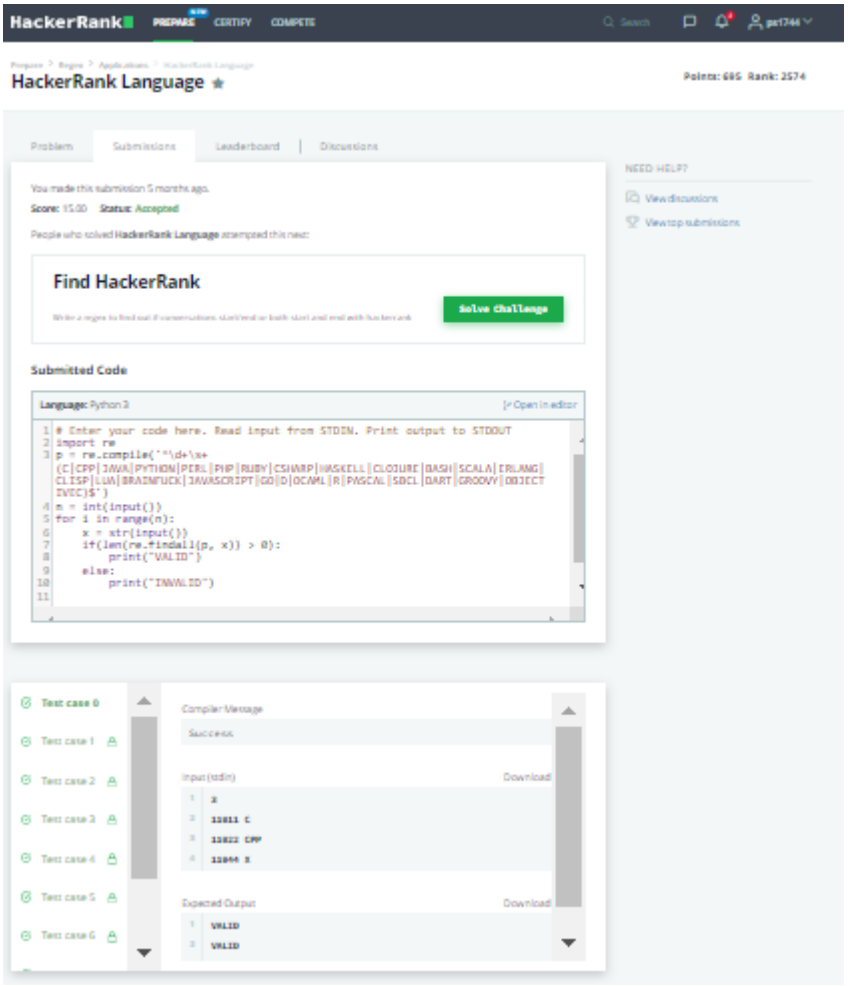
39)


HackerRank
PREPARE
CERTIFY
COMPETE
Valid PAN format
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 15.00 Status: Accepted
People who solved Valid PAN format attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
p = re.compile('[A-Z]{5}[0-9]{4}[A-Z]')
n = int(input())
for i in range(n):
x = str(input())
if(len(x) != 10):
print("NO")
elif(len(re.findall(p, x)) > 0):
print("YES")
else:
print("NO")
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
Expected Output
Download
YES
NO

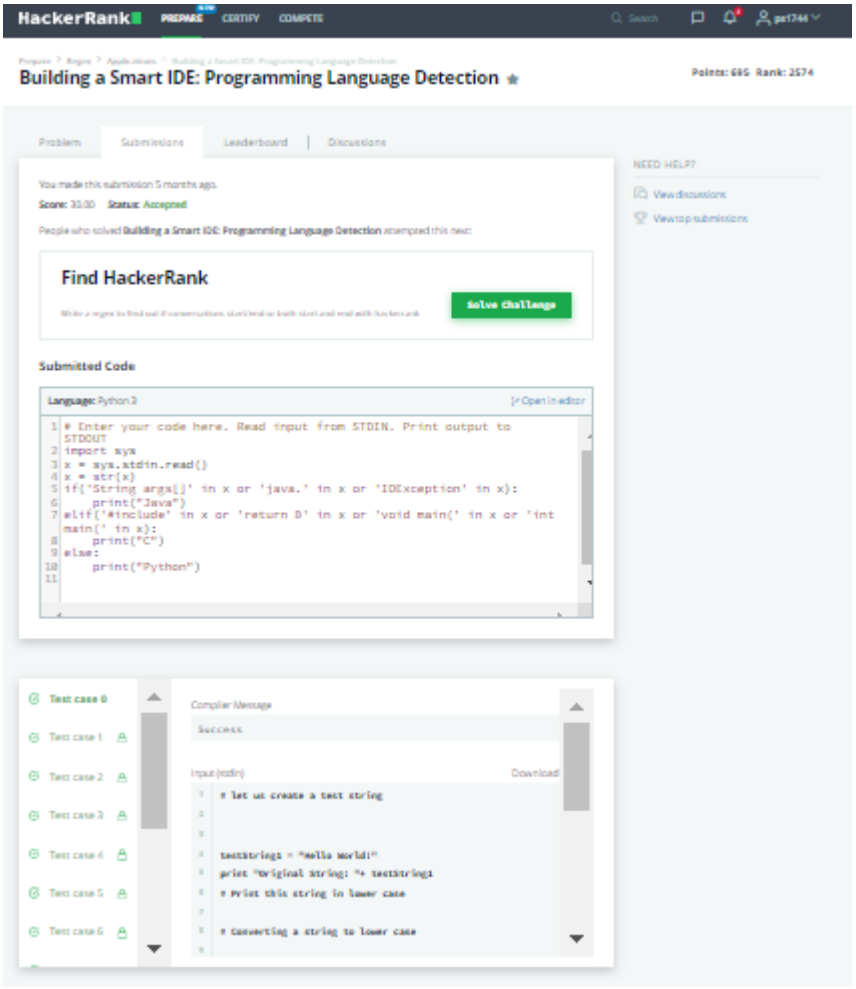
40)


HackerRank
PREPARE
CERTIFY
COMPETE
Saying Hi
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
NEED HELP?
View discussions
View top submissions
You made this submission 5 months ago.
Score: 15.00 Status: Accepted
People who solved Saying Hi attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
n = int(input())
import re
p = re.compile('^[Hh][Ii]\s[^dD].*')
for i in range(n):
x = str(input())
if(len(re.findall(p, x)) > 0):
print(x)
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
5
Hi Alex how are you doing
hi dave how are you doing
Good by Alex
hidden agenda
Alex greeted Martha by saying Hi Martha
Expected Output
Download

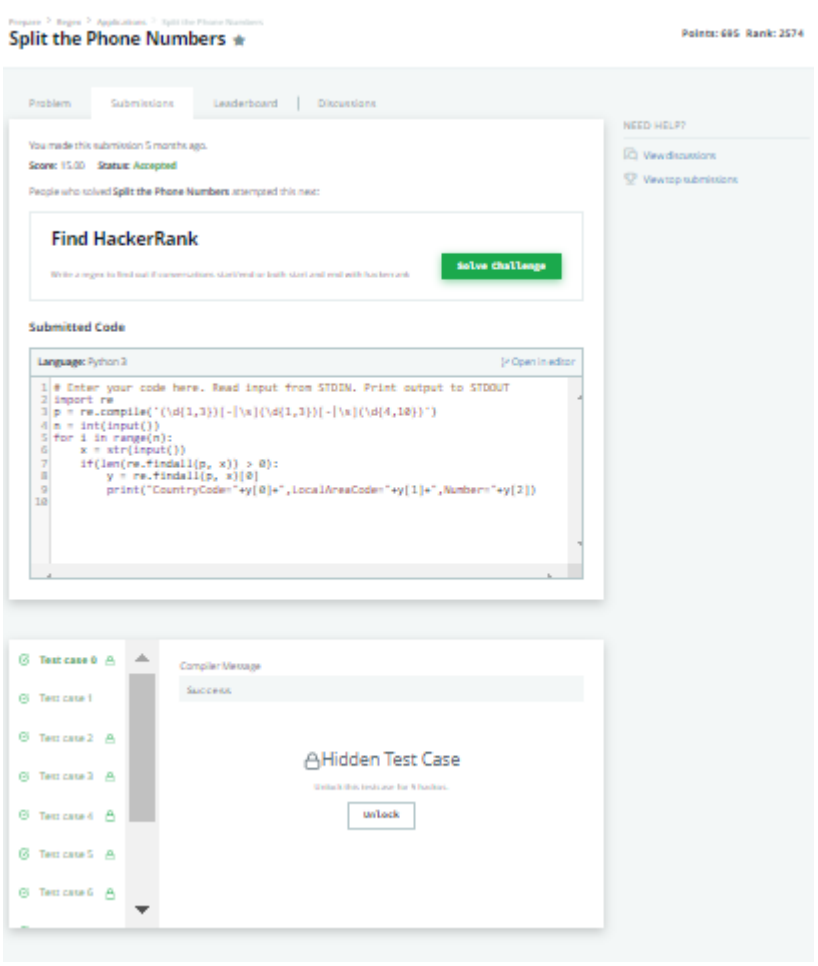
41)


HackerRank
PREPARE
CERTIFY
COMPETE
HackerRank Language
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 15.00 Status: Accepted
People who solved HackerRank Language attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
p = re.compile('^\d+\s+(C|CPP|JAVA|PYTHON|PERL|PHP|RUBY|CSHARP|HASKELL|CLOJURE|BASH|SCALA|ERLANG|CLISP|LUA|BRAINFUCK|JAVASCRIPT|GO|D|OCAML|R|PASCAL|SDCL|DART|GROOVY|OBJECT IVEC)$')
n = int(input())
for i in range(n):
x = str(input())
if(len(re.findall(p, x)) > 0):
print("VALID")
else:
print("INVALID")
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
3
11811 C
11822 CPP
11844 X
Expected Output
Download
VALID
VALID

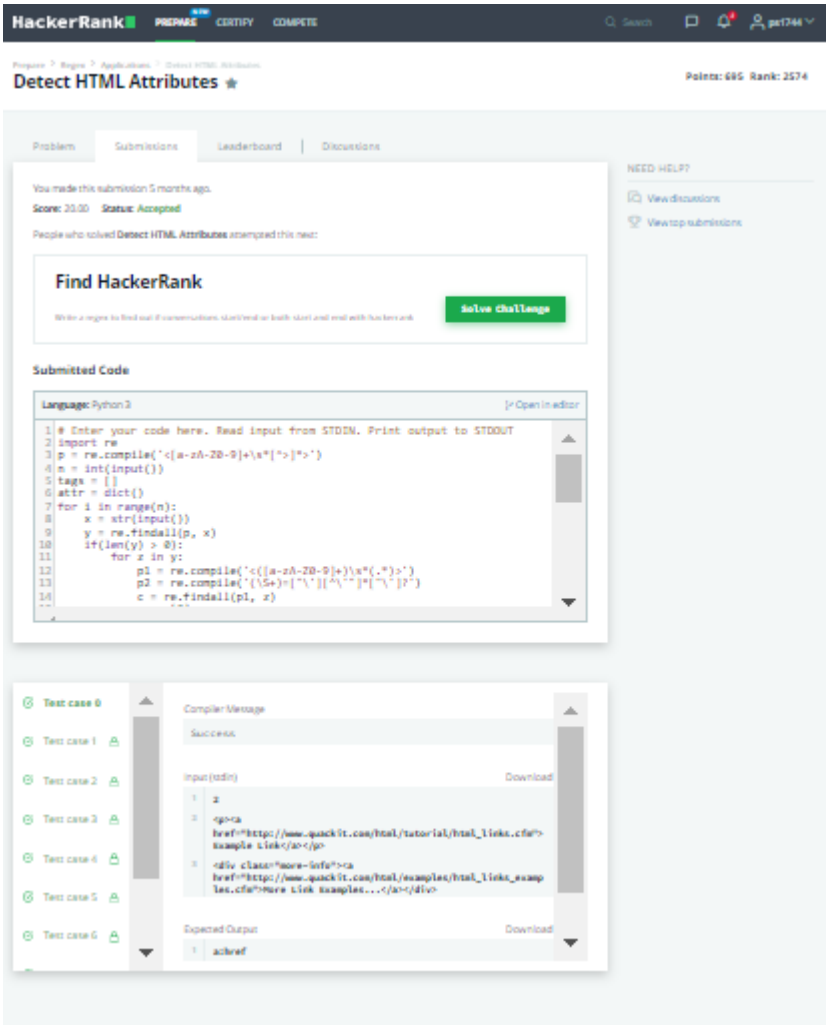
42)


HackerRank
PREPARE
CERTIFY
COMPETE
Building a Smart IDE: Programming Language Detection
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 30.00 Status: Accepted
People who solved Building a Smart IDE: Programming Language Detection attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import sys
x = sys.stdin.read()
x = str(x)
if('String args[]' in x or 'java.' in x or 'IOException' in x):
print("Java")
elif('#include' in x or 'return 0' in x or 'void main(' in x or 'int main(' in x):
print("C")
else:
print("Python")
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
# let us create a test string
teststring1 = "Hello World!"
print "Original String: "+ teststring1
Print this string in lower case
Converting a string to lower case

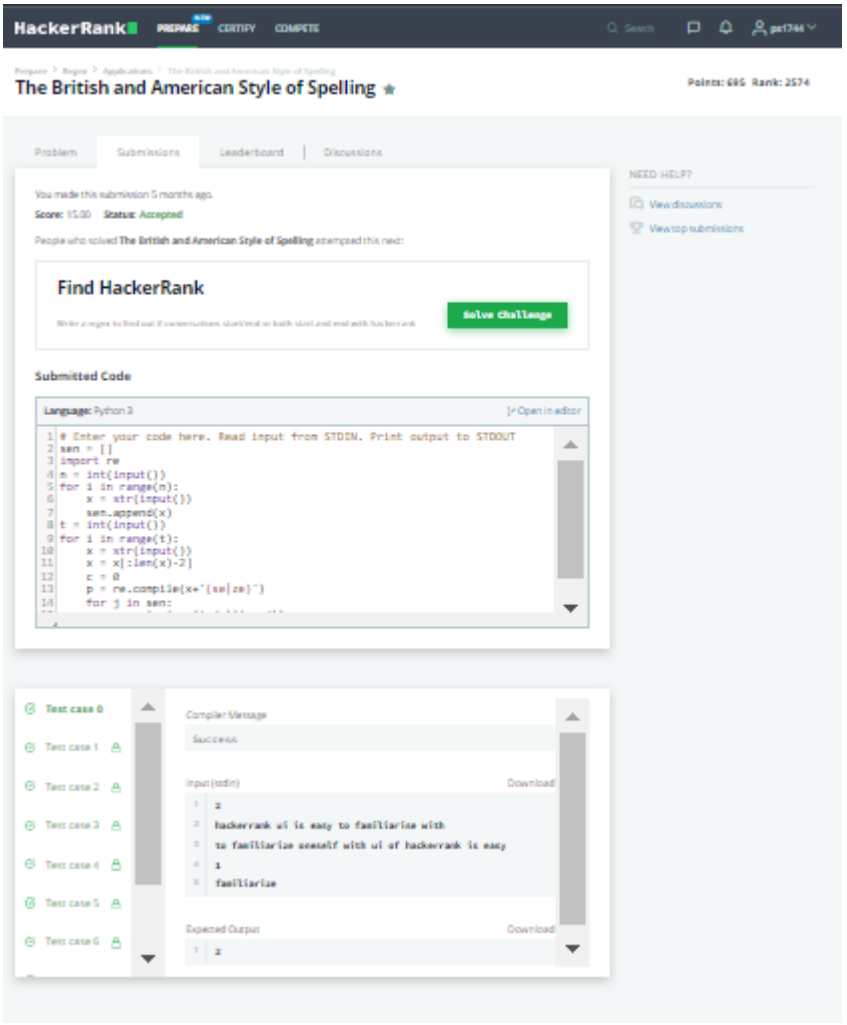
43)


Split the Phone Numbers
Points: 685 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
NEED HELP?
View discussions
View top submissions
You made this submission 5 months ago.
Score: 15.00 Status: Accepted
People who solved Split the Phone Numbers also solved this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
p = re.compile('(\d{1,3})[-|\s](\d{1,3})[-|\s](\d{4,10})')
n = int(input())
for i in range(n):
x = str(input())
if(len(re.findall(p, x)) > 0):
y = re.findall(p, x)[0]
print("CountryCode="+y[0]+",LocalAreaCode="+y[1]+",Number="+y[2])
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Hidden Test Case
Unlock

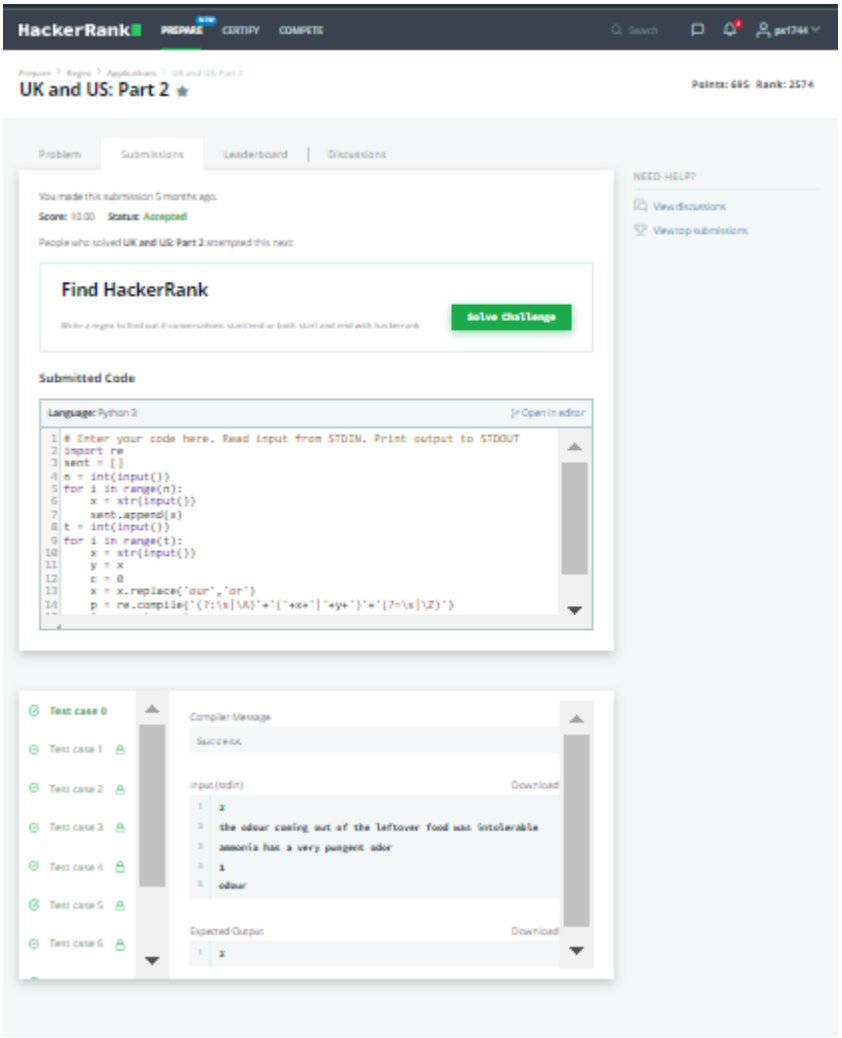
44)


HackerRank
PREPARE
CERTIFY
COMPETE
Detect HTML Attributes
Points: 685 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 20.00 Status: Accepted
People who solved Detect HTML Attributes attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
p = re.compile('<[a-zA-Z0-9]+\s*[^>]*>')
n = int(input())
tags = []
attr = dict()
for i in range(n):
x = str(input())
y = re.findall(p, x)
if(len(y) > 0):
for z in y:
p1 = re.compile('<([a-zA-Z0-9]+)\s*(.*)>')
p2 = re.compile('(\S+)=["\']?([^\"\']*)["\']?')
c = re.findall(p1, z)
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
2
Example Link
More Link Examples...
Expected Output
Download
active

45)


HackerRank
PREPARE
CERTIFY
COMPETE
The British and American Style of Spelling
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
NEED HELP?
View discussion
View top submissions
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
sen = []
import re
n = int(input())
for i in range(n):
x = str(input())
sen.append(x)
t = int(input())
for i in range(t):
x = str(input())
x = x[:len(x)-2]
c = 0
p = re.compile(x+'(se|ze)')
for j in sen:
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
2
hackerrank ui is easy to familiarize with
to familiarize oneself with ui of hackerrank is easy
1
familiarize
Expected Output
Download
2

46)


HackerRank
PREPARE
CERTIFY
COMPETE
UK and US: Part 2
Points: 695 Rank: 2574
Problem
Submissions
Leaderboard
Discussions
You made this submission 5 months ago.
Score: 10.00 Status: Accepted
People who solved UK and US: Part 2 attempted this next:
Find HackerRank
Solve Challenge
Submitted Code
Language: Python 3
Open in editor
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
sent = []
n = int(input())
for i in range(n):
x = str(input())
sent.append(x)
t = int(input())
for i in range(t):
x = str(input())
y = x
c = 0
x = x.replace('our','or')
p = re.compile('(?:\s|\A)'+'('+x+'|'+y+')'+'(?=\s|\Z)')
NEED HELP?
View discussions
View top submissions
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
2
the odour coming out of the leftover food was intolerable
ammonia has a very pungent odor
1
odour
Expected Output
Download
2

LAB EXPERIMENTS

Palak Sharma
RA1911033010112

Expt. No. 1
Implementation of Lexical Analyser

AIM :

Write a program in your preferred language for the Implementation of Lexical Analyser

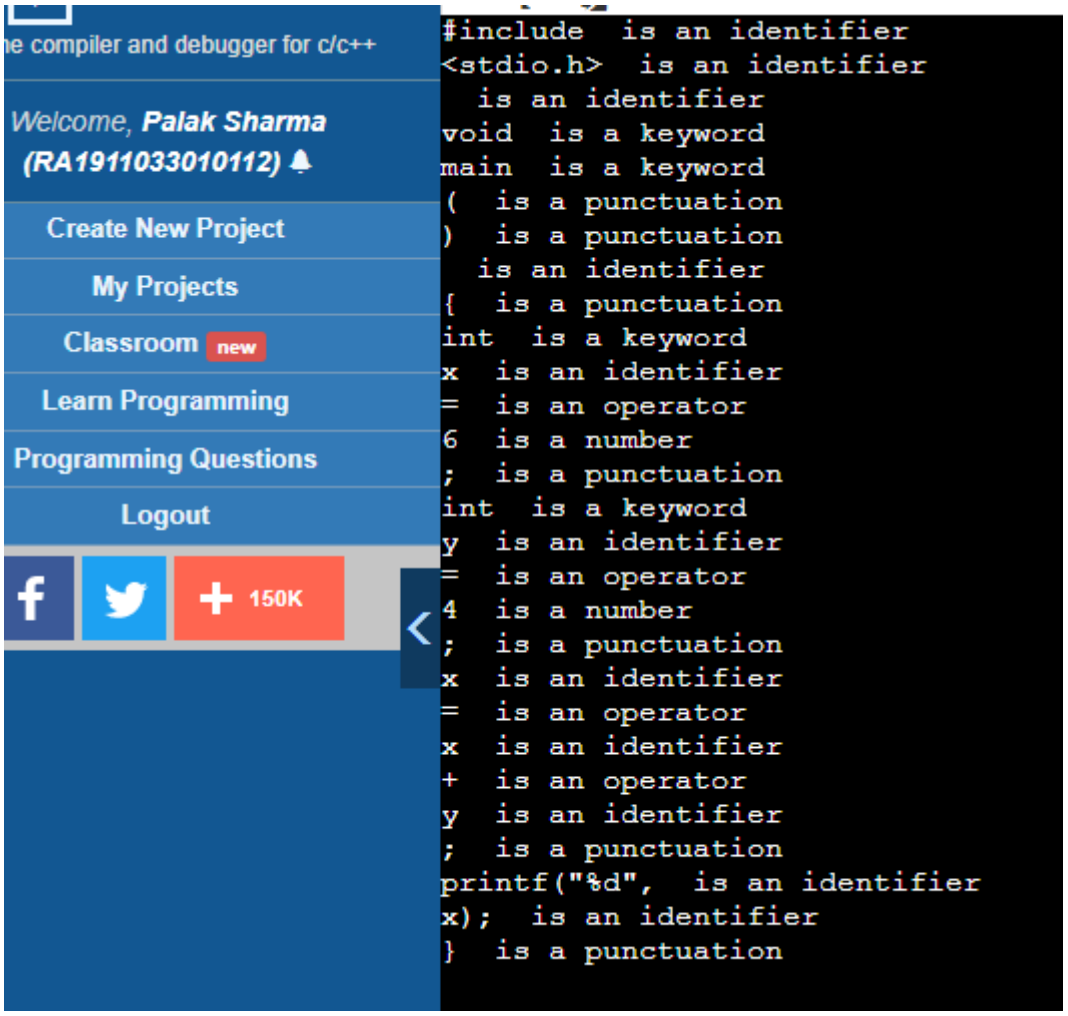
ALGORITHM :

1. Start.
2. Get the input program from the file prog.txt.
3. Read the program line by line and check if each word in a line is a keyword, identifier, constant or an operator.
4. If the word read is an identifier, assign a number to the identifier and make an entry into the symbol table stored in sybol.txt.
5. For each lexeme read, generate a token as follows: a. If the lexeme is an identifier, then the token generated is of the form b. If the lexeme is an operator, then the token generated is . c. If the lexeme is a constant, then the token generated is . d. If the lexeme is a keyword, then the token is the keyword itself.
6. The stream of tokens generated are displayed in the console output.
7. Stop.

CODE :

```
file  = open("./add.c", 'r')
lines = file.readlines()

keywords    = ["void", "main", "int", "float", "bool", "if", "for", "else", "while", "char",
"return"]
operators   = ["=", "==", "+", "-", "*", "/", "++", "--", "+=", "-=", "!=", "||", "&&"]
punctuations= [";", "(", ")", "{", "}", "[", "]"]

def is_int(x):
    try:
        int(x)
        return True
    except:
        return False

for line in lines:
    for i in line.strip().split(" "):
        if i in keywords:
            print (i, " is a keyword")
        elif i in operators:
            print (i, " is an operator")
        elif i in punctuations:
            print (i, " is a punctuation")
        elif is_int(i):
            print (i, " is a number")
        else:
            print (i, " is an identifier")
```

OUTPUT :



RESULT : Implementation of Lexical Analyser is done in python.

Palak Sharma
RA1911033010112

Expt. No. 2
Regular expression to NFA

AIM :

To write a program in any language to convert a Regular Expression (RE) to a Non-Deterministic Finite Automata (NFA).

ALGORITHM :

1.Create the mandatory classes and accept the expression from the user.
2.After taking the RE as input convert it to postfix. This is done so as to know what the operator is to be used with how many variables.
3.A tree will be constructed after converting to postfix depending on the operators and the variables. A* will have a node where one of the next state values will be itself. A|B will have a state, branching into 2 new states.
4.Then we'll evaluate the regular expression. Going from state to state, keeping a record of all the next states.
5. Finally, once all the nodes are traversed, print the list of states and the transition table.

CODE :

```
class Type:
    SYMBOL = 1
    CONCAT = 2
    UNION = 3
    KLEENE = 4

class ExpressionTree:
    def __init__(self, _type, value=None):
        self._type = _type
        self.value = value
        self.left = None
        self.right = None

def constructTree(regexp):
    stack = []
    for c in regexp:
```

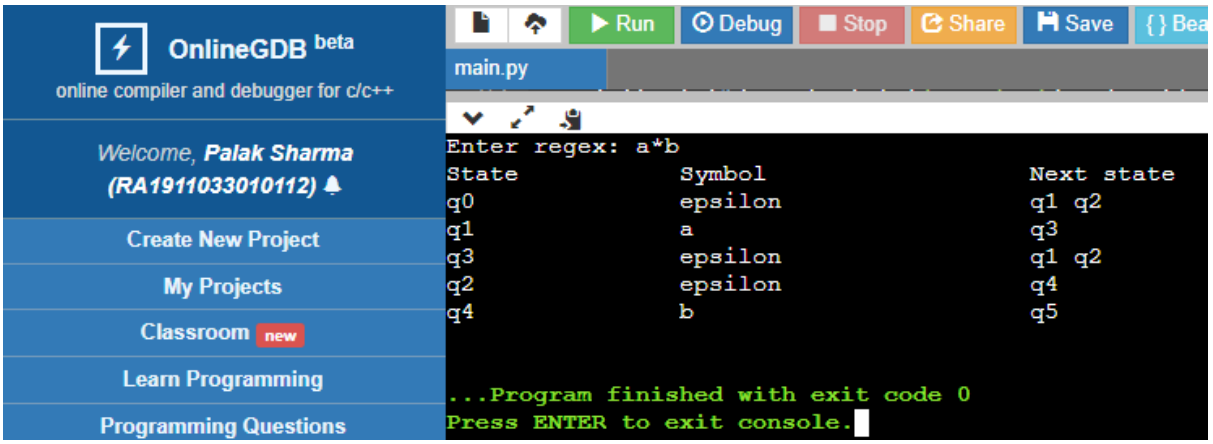
```
        if c.isalpha():
            stack.append(ExpressionTree(Type.SYMBOL, c))
        else:
            if c == "|":
                z = ExpressionTree(Type.UNION)
                z.right = stack.pop()
                z.left = stack.pop()
            elif c == ".":
                z = ExpressionTree(Type.CONCAT)
                z.right = stack.pop()
                z.left = stack.pop()
            elif c == "*":
                z = ExpressionTree(Type.KLEENE)
                z.left = stack.pop()
            stack.append(z)
    return stack[0]

def higherPrecedence(a, b):
    p = ["|", ".", "*"]
    return p.index(a) > p.index(b)

def postfix(regexp):
    temp = []
    for i in range(len(regexp)):
        if i != 0 and (regexp[i-1].isalpha() or regexp[i-1] == ")" or regexp[i-1] == "*") and
(regexp[i].isalpha() or regexp[i] == "("):
            temp.append(".")
        temp.append(regexp[i])
    regexp = temp
    stack = []
    output = ""
    for c in regexp:
        if c.isalpha():
            output = output + c
            continue
        if c == ")":
            while len(stack) != 0 and stack[-1] != "(":
```

```
            output = output + stack.pop()
        stack.pop()
    elif c == "(":
        stack.append(c)
    elif c == "*":
        output = output + c
    elif len(stack) == 0 or stack[-1] == "(" or higherPrecedence(c, stack[-1]):
        stack.append(c)
    else:
        while len(stack) != 0 and stack[-1] != "(" and not higherPrecedence(c, stack[-1]):
            output = output + stack.pop()
        stack.append(c)
while len(stack) != 0:
    output = output + stack.pop()
return output

class FiniteAutomataState:
    def __init__(self):
        self.next_state = {}

def evalRegex(et):
    if et._type == Type.SYMBOL:
        return evalRegexSymbol(et)
    elif et._type == Type.CONCAT:
        return evalRegexConcat(et)
    elif et._type == Type.UNION:
        return evalRegexUnion(et)
    elif et._type == Type.KLEENE:
        return evalRegexKleene(et)

def evalRegexSymbol(et):
    start_state = FiniteAutomataState()
    end_state = FiniteAutomataState()
    start_state.next_state[et.value] = [end_state]
    return start_state, end_state

def evalRegexConcat(et):
```

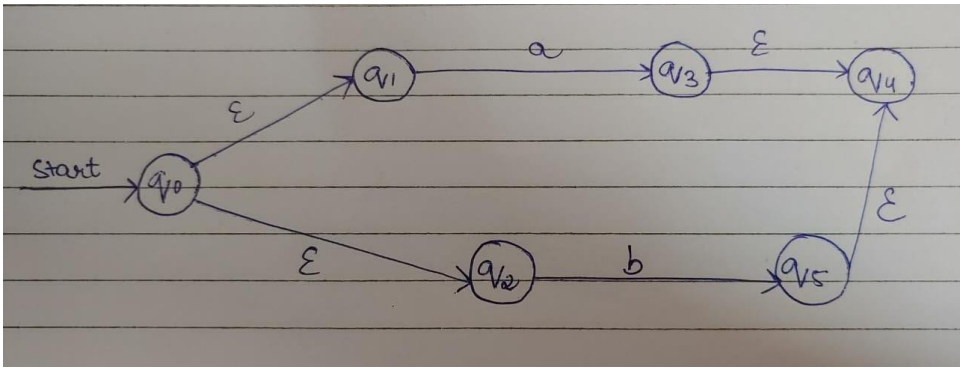
```
    left_nfa = evalRegex(et.left)
    right_nfa = evalRegex(et.right)
    left_nfa[1].next_state['epsilon'] = [right_nfa[0]]
    return left_nfa[0], right_nfa[1]

def evalRegexUnion(et):
    start_state = FiniteAutomataState()
    end_state = FiniteAutomataState()
    up_nfa = evalRegex(et.left)
    down_nfa = evalRegex(et.right)
    start_state.next_state['epsilon'] = [up_nfa[0], down_nfa[0]]
    up_nfa[1].next_state['epsilon'] = [end_state]
    down_nfa[1].next_state['epsilon'] = [end_state]
    return start_state, end_state

def evalRegexKleene(et):
    start_state = FiniteAutomataState()
    end_state = FiniteAutomataState()
    sub_nfa = evalRegex(et.left)
    start_state.next_state['epsilon'] = [sub_nfa[0], end_state]
    sub_nfa[1].next_state['epsilon'] = [sub_nfa[0], end_state]
    return start_state, end_state

def printStateTransitions(state, states_done, symbol_table):
    if state in states_done:
        return
    states_done.append(state)
    for symbol in list(state.next_state):
        line_output = "q" + str(symbol_table[state]) + "\t\t" + symbol + "\t\t\t"
        for ns in state.next_state[symbol]:
            if ns not in symbol_table:
                symbol_table[ns] = 1 + sorted(symbol_table.values())[-1]
            line_output = line_output + "q" + str(symbol_table[ns]) + " "
        print(line_output)
        for ns in state.next_state[symbol]:
            printStateTransitions(ns, states_done, symbol_table)
```

```
def printTransitionTable(finite_automata):
    print("State\t\tSymbol\t\tNext state")
    printStateTransitions(finite_automata[0], [], {finite_automata[0]: 0})

r = input("Enter regex: ")
pr = postfix(r)
et = constructTree(pr)

fa = evalRegex(et)
printTransitionTable(fa)
```

OUTPUT :



To verify the output here is a hand drawn conversion.



RESULT : Conversion of a Regular Expression (RE) to a Non-Deterministic Finite Automata (NFA).

Palak Sharma
RA1911033010112

EX. NO. 3
CONVERSION OF NFA TO DFA

AIM :

To write a program for converting NFA to DFA.

ALGORITHM :

1. Start
2. Get the input from the user
3. Set the only state in SDFA to “unmarked”.
4. while SDFA contains an unmarked state do:

a. Let T be that unmarked state

b. for each a in % do S = e-Closure(MoveNFA(T,a))

c. if S is not in SDFA already then, add S to SDFA (as an “unmarked” state)

d. Set MoveDFA(T,a) to S

5. For each S in SDFA if any s & S is a final state in the NFA then, mark S an a final state in the DFA
6. Print the result.
7. Stop the program

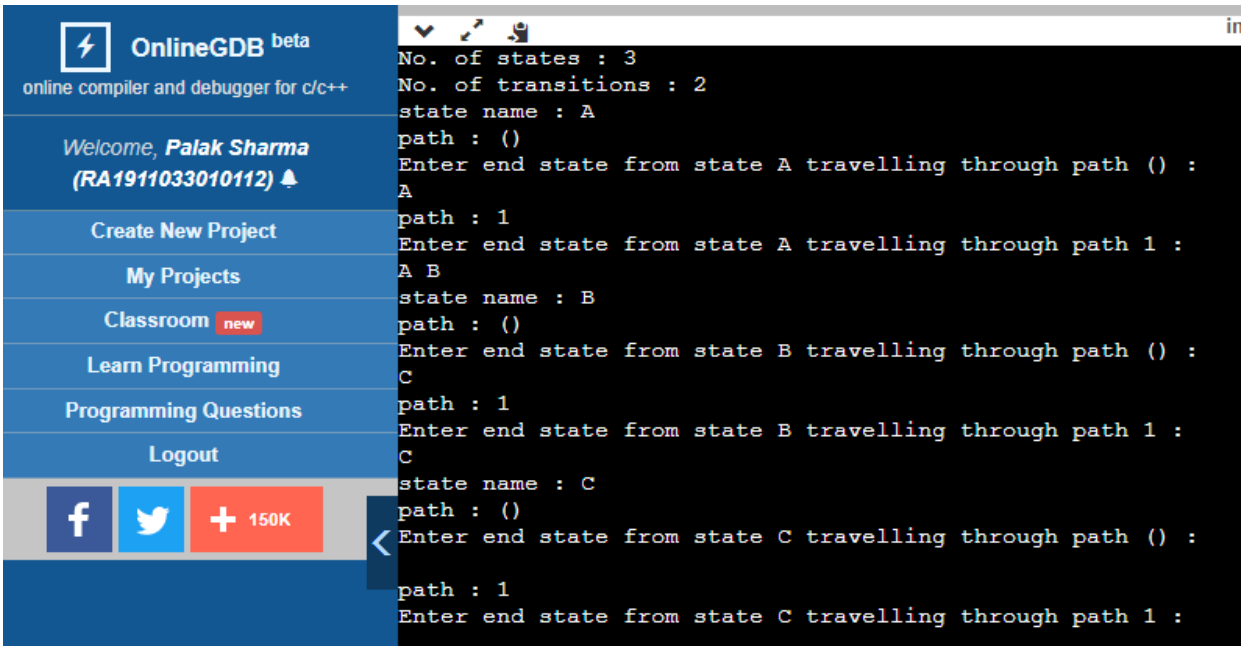
CODE :

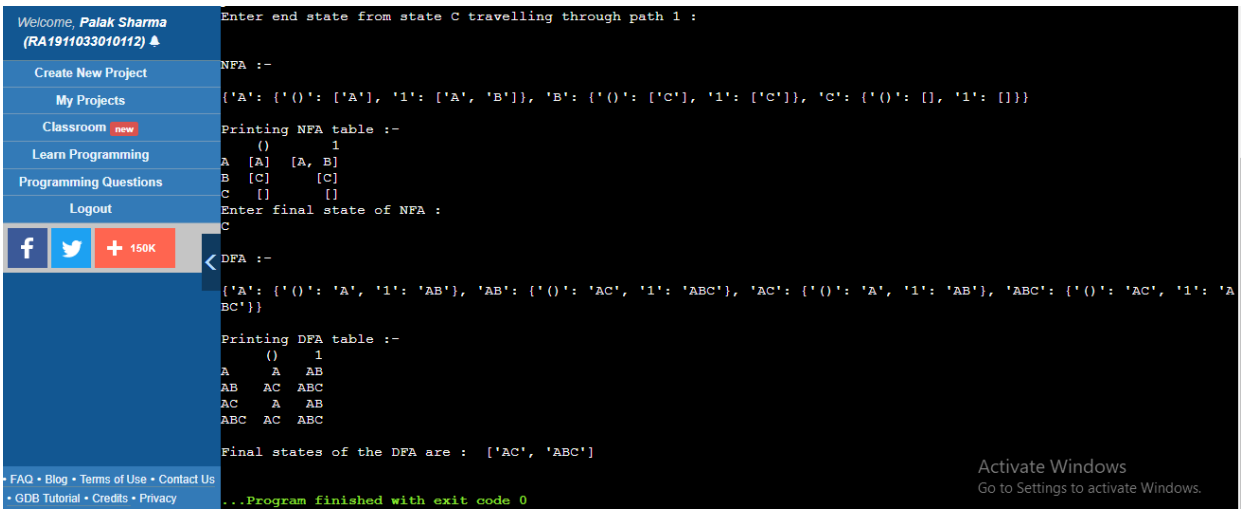
```
import pandas as pd

nfa = {}
n = int(input("No. of states : "))
t = int(input("No. of transitions : "))
for i in range(n):
    state = input("state name : ")
    nfa[state] = {}
    for j in range(t):
        path = input("path : ")
        print("Enter end state from state {} travelling through path {} : ".format(state, path))
        reaching_state = [x for x in input().split()]
        nfa[state][path] = reaching_state
```

```
print("\nNFA :- \n")
print(nfa)
print("\nPrinting NFA table :- ")
nfa_table = pd.DataFrame(nfa)
print(nfa_table.transpose())

print("Enter final state of NFA : ")
nfa_final_state = [x for x in input().split()]

new_states_list = []

#---------------------------------------------

dfa = {}
keys_list = list(
    list(nfa.keys())[0])
path_list = list(nfa[keys_list[0]].keys())

dfa[keys_list[0]] = {}
for y in range(t):
    var = "".join(nfa[keys_list[0]][
                    path_list[y]])
    dfa[keys_list[0]][path_list[y]] = var
    if var not in keys_list:
        new_states_list.append(var)
        keys_list.append(var)

while len(new_states_list) != 0:
    dfa[new_states_list[0]] = {}
    for _ in range(len(new_states_list[0])):
        for i in range(len(path_list)):
            temp = []
            for j in range(len(new_states_list[0])):
                temp += nfa[new_states_list[0][j]][path_list[i]]
            s = ""
            s = s.join(temp)
            if s not in keys_list:
                new_states_list.append(s)
                keys_list.append(s)
```

```
            dfa[new_states_list[0]][path_list[i]] = s

    new_states_list.remove(new_states_list[0])

print("\nDFA :- \n")
print(dfa)
print("\nPrinting DFA table :- ")
dfa_table = pd.DataFrame(dfa)
print(dfa_table.transpose())

dfa_states_list = list(dfa.keys())
dfa_final_states = []
for x in dfa_states_list:
    for i in x:
        if i in nfa_final_state:
            dfa_final_states.append(x)
            break

print("\nFinal states of the DFA are : ", dfa_final_states)
```

OUTPUT :





RESULT :

The given NFA was converted to a DFA using python successfully.

Palak Sharma
RA1911033010112

EX. NO. 4(a)
ELIMINATION OF LEFT RECURSION

AIM :

A program for Elimination of Left Recursion.

ALGORITHM :

1. Start the program.
2. Initialize the arrays for taking input from the user.
3. Prompt the user to input the no. of non-terminals having left recursion and no. of productions for these non-terminals.
4. Prompt the user to input the production for non-terminals.
5. Eliminate left recursion using the following rules:-

   A->Aα1| Aα2 | . . . . . |Aαm

   A->β1| β2| . . . . .| βn

   Then replace it by

   A-> βi A' i=1,2,3,…..m

   A'-> αj A' j=1,2,3,…..n

   A'-> Ɛ
6. After eliminating the left recursion by applying these rules, display the productions without left recursion.
7. Stop.

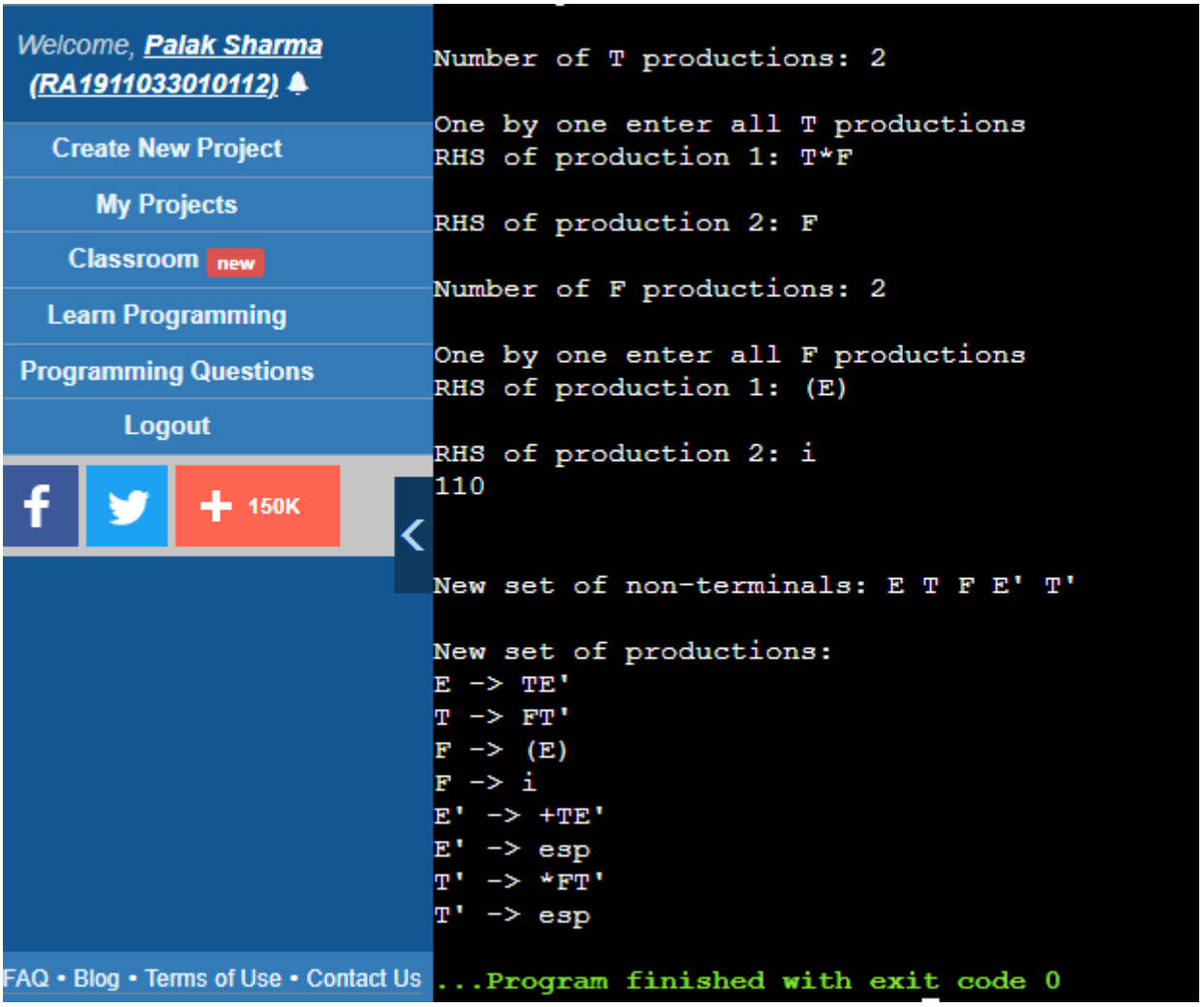
CODE :

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

int main()
{
   int n;
   cout<<"\nEnter number of non terminals: ";
   cin>>n;
   cout<<"\nEnter non terminals one by one: ";
   int i;
   vector<string> nonter(n);
   vector<int> leftrecr(n,0);
   for(i=0;i<n;++i) {
         cout<<"\nNon terminal "<<i+1<<" : ";
      cin>>nonter[i];
   }
   vector<vector<string> > prod;
   cout<<"\nEnter 'esp' for null";
   for(i=0;i<n;++i) {
      cout<<"\nNumber of "<<nonter[i]<<" productions: ";
      int k;
      cin>>k;
      int j;
      cout<<"\nOne by one enter all "<<nonter[i]<<" productions";
      vector<string> temp(k);
      for(j=0;j<k;++j) {
         cout<<"\nRHS of production "<<j+1<<": ";
         string abc;
         cin>>abc;
```

```
            temp[j]=abc;

if(nonter[i].length()<=abc.length()&&nonter[i].compare(abc.substr(0,nonter[i].length()))==0)
                leftrecr[i]=1;
        }
        prod.push_back(temp);
    }
    for(i=0;i<n;++i) {
        cout<<leftrecr[i];
    }
    for(i=0;i<n;++i) {
        if(leftrecr[i]==0)
            continue;
        int j;
        nonter.push_back(nonter[i]+"'");
        vector<string> temp;
        for(j=0;j<prod[i].size();++j) {

if(nonter[i].length()<=prod[i][j].length()&&nonter[i].compare(prod[i][j].substr(0,nonter[i].lengt
h()))==0) {
                string
abc=prod[i][j].substr(nonter[i].length(),prod[i][j].length()-nonter[i].length())+nonter[i]+"'";
                temp.push_back(abc);
                prod[i].erase(prod[i].begin()+j);
                --j;
            }
            else {
                prod[i][j]+=nonter[i]+"'";
            }
        }
        temp.push_back("esp");
        prod.push_back(temp);
    }
    cout<<"\n\n";
    cout<<"\nNew set of non-terminals: ";
    for(i=0;i<nonter.size();++i)
        cout<<nonter[i]<<" ";
    cout<<"\n\nNew set of productions: ";
```

```
    for(i=0;i<nonter.size();++i) {
        int j;
        for(j=0;j<prod[i].size();++j) {
            cout<<"\n"<<nonter[i]<<" -> "<<prod[i][j];
        }
    }
    return 0;
}
```

OUTPUT :



RESULT : A program for Elimination of Left Recursion was run successfully

Palak Sharma
RA1911033010112

EX. NO. 4(b)
LEFT FACTORING

AIM :

A program for implementation Of Left Factoring

ALGORITHM :

1. Start
2. Ask the user to enter the set of productions
3. Check for common symbols in the given set of productions by comparing
   with: A->aB1|aB2
4. If found, replace the particular productions with:
   A->aA'
   A'->B1 | B2|ε
5. Display the output
6. Exit

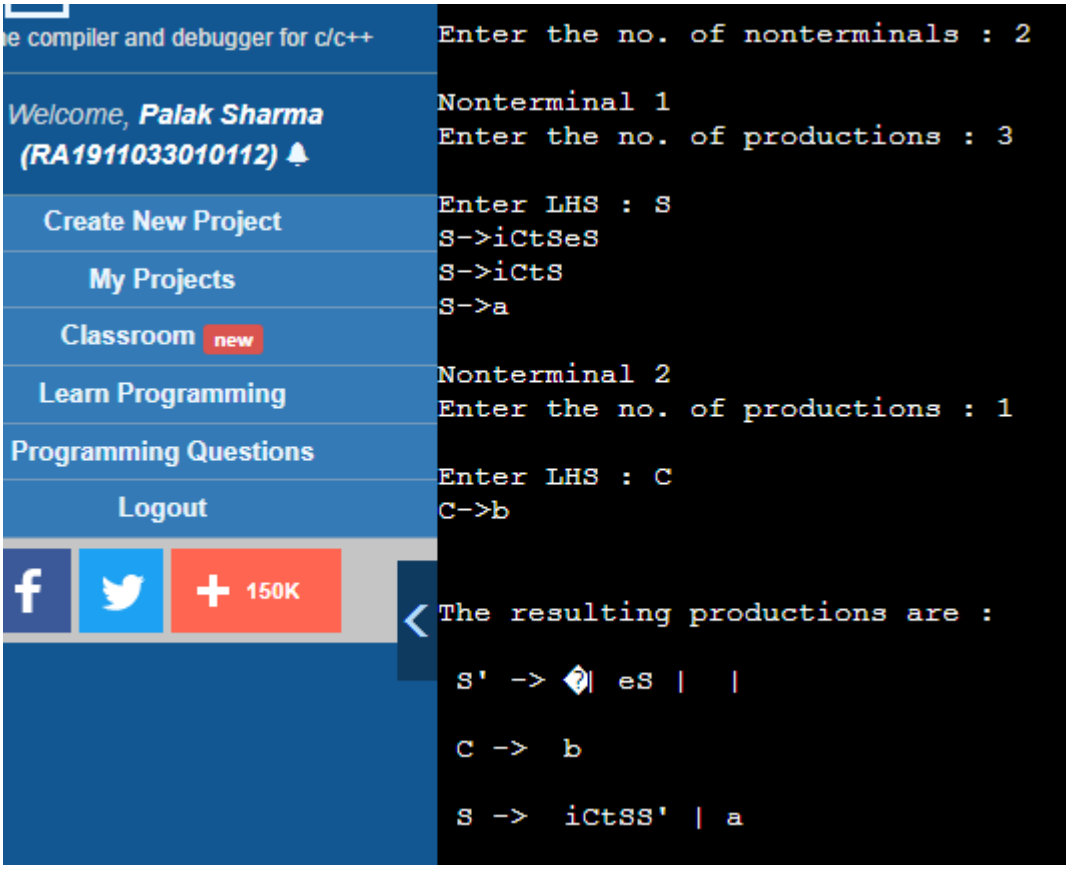
CODE :

```
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
void main()
{
    char ch,lhs[20][20],rhs[20][20][20],temp[20],temp1[20];
    int n,n1,count[20],x,y,i,j,k,c[20];
    printf("\nEnter the no. of nonterminals : ");
```

```
scanf("%d",&n);
n1=n;
for(i=0;i<n;i++)
{
	printf("\nNonterminal %d \nEnter the no. of productions : ",i+1);
	scanf("%d",&c[i]);
	printf("\nEnter LHS : ");
	scanf("%s",lhs[i]);
	for(j=0;j<c[i];j++)
	{
		printf("%s->",lhs[i]);
		scanf("%s",rhs[i][j]);
	}
}
for(i=0;i<n;i++)
{
	count[i]=1;
	while(memcmp(rhs[i][0],rhs[i][1],count[i])==0)
		count[i]++;
}
for(i=0;i<n;i++)
{
	count[i]--;
	if(count[i]>0)
	{
		strcpy(lhs[n1],lhs[i]);
		strcat(lhs[i],"'");
		for(k=0;k<count[i];k++)
			temp1[k] = rhs[i][0][k];
		temp1[k++] = '\0';
		for(j=0;j<c[i];j++)
		{
			for(k=count[i],x=0;k<strlen(rhs[i][j]);x++,k++)
```

```
                                        temp[x] = rhs[i][j][k];
                                temp[x++] = '\0';
                                if(strlen(rhs[i][j])==1)
                                        strcpy(rhs[n1][1],rhs[i][j]);
                                strcpy(rhs[i][j],temp);
                        }
                        c[n1]=2;
                        strcpy(rhs[n1][0],temp1);
                        strcat(rhs[n1][0],lhs[n1]);
                        strcat(rhs[n1][0],"'");
                        n1++;
                }
        }
        printf("\n\nThe resulting productions are : \n");
        for(i=0;i<n1;i++)
        {
                if(i==0)
                        printf("\n %s -> %c|",lhs[i],(char)238);
                else
                        printf("\n %s -> ",lhs[i]);
                for(j=0;j<c[i];j++)
                {
                        printf(" %s ",rhs[i][j]);
                        if((j+1)!=c[i])
                                printf("|");
                }
                printf("\b\b\b\n");
        }
}
```

OUTPUT :



RESULT : A program for implementation Of Left Factoring was compiled and run successful

Palak Sharma
RA1911033010112

Exp 5
Computation of First and Follow

AIM :

To study and implement Computation of First and Follow.

ALGORITHM :

> If x is a terminal, then FIRST(x) = { 'x' }
> If x-> Є, is a production rule, then add Є to FIRST(x). ➢ If X->Y1 Y2 Y3….Yn is a production,
- FIRST(X) = FIRST(Y1)
- If FIRST(Y1) contains Є then FIRST(X) = { FIRST(Y1) – Є } U { FIRST(Y2) }
- If FIRST (Yi) contains Є for all i = 1 to n, then add Є to FIRST(X).

> Always check the right side of the productions for a non-terminal, whose FOLLOW set is being found. (never see the left side). ➢ Follow =>
- If that non-terminal (S,A,B…) is followed by any terminal (a,b…,*,+,(,)…) , then add that "terminal" into FOLLOW set. • If that non-terminal is followed by any other non-terminal then add "FIRST of other nonterminal" into FOLLOW set.

CODE :

```
// C program to calculate the First and
// Follow sets of a given grammar
#include<stdio.h>
#include<ctype.h>
#include<string.h>

// Functions to calculate Follow
void followfirst(char, int, int);
void follow(char c);

// Function to calculate First
void findfirst(char, int, int);
```

```
int count, n = 0;

// Stores the final result
// of the First Sets
char calc_first[10][100];

// Stores the final result
// of the Follow Sets
char calc_follow[10][100];
int m = 0;

// Stores the production rules
char production[10][10];
char f[10], first[10];
int k;
char ck;
int e;

int main(int argc, char **argv)
{
        int jm = 0;
        int km = 0;
        int i, choice;
        char c, ch;
        count = 8;

        // The Input grammar
        strcpy(production[0], "E=TR");
        strcpy(production[1], "R=+TR");
        strcpy(production[2], "R=#");
        strcpy(production[3], "T=FY");
        strcpy(production[4], "Y=*FY");
        strcpy(production[5], "Y=#");
        strcpy(production[6], "F=(E)");
        strcpy(production[7], "F=i");
```

```
int kay;
char done[count];
int ptr = -1;

// Initializing the calc_first array
for(k = 0; k < count; k++) {
        for(kay = 0; kay < 100; kay++) {
                calc_first[k][kay] = '!';
        }
}
int point1 = 0, point2, xxx;

for(k = 0; k < count; k++)
{
        c = production[k][0];
        point2 = 0;
        xxx = 0;

        // Checking if First of c has
        // already been calculated
        for(kay = 0; kay <= ptr; kay++)
                if(c == done[kay])
                        xxx = 1;

        if (xxx == 1)
                continue;

        // Function call
        findfirst(c, 0, 0);
        ptr += 1;

        // Adding c to the calculated list
        done[ptr] = c;
        printf("\n First(%c) = { ", c);
```

```
		calc_first[point1][point2++] = c;

		// Printing the First Sets of the grammar
		for(i = 0 + jm; i < n; i++) {
			int lark = 0, chk = 0;

			for(lark = 0; lark < point2; lark++) {

				if (first[i] == calc_first[point1][lark])
				{
					chk = 1;
					break;
				}
			}
			if(chk == 0)
			{
				printf("%c, ", first[i]);
				calc_first[point1][point2++] = first[i];
			}
		}
		printf("}\n");
		jm = n;
		point1++;
	}
	printf("\n");
	printf("----------------------------------------------\n\n");
	char donee[count];
	ptr = -1;

	// Initializing the calc_follow array
	for(k = 0; k < count; k++) {
		for(kay = 0; kay < 100; kay++) {
			calc_follow[k][kay] = '!';
		}
	}
```

```
point1 = 0;
int land = 0;
for(e = 0; e < count; e++)
{
        ck = production[e][0];
        point2 = 0;
        xxx = 0;

        // Checking if Follow of ck
        // has alredy been calculated
        for(kay = 0; kay <= ptr; kay++)
                if(ck == donee[kay])
                        xxx = 1;

        if (xxx == 1)
                continue;
        land += 1;

        // Function call
        follow(ck);
        ptr += 1;

        // Adding ck to the calculated list
        donee[ptr] = ck;
        printf(" Follow(%c) = { ", ck);
        calc_follow[point1][point2++] = ck;

        // Printing the Follow Sets of the grammar
        for(i = 0 + km; i < m; i++) {
                int lark = 0, chk = 0;
                for(lark = 0; lark < point2; lark++)
                {
                        if (f[i] == calc_follow[point1][lark])
                        {
                                chk = 1;
```

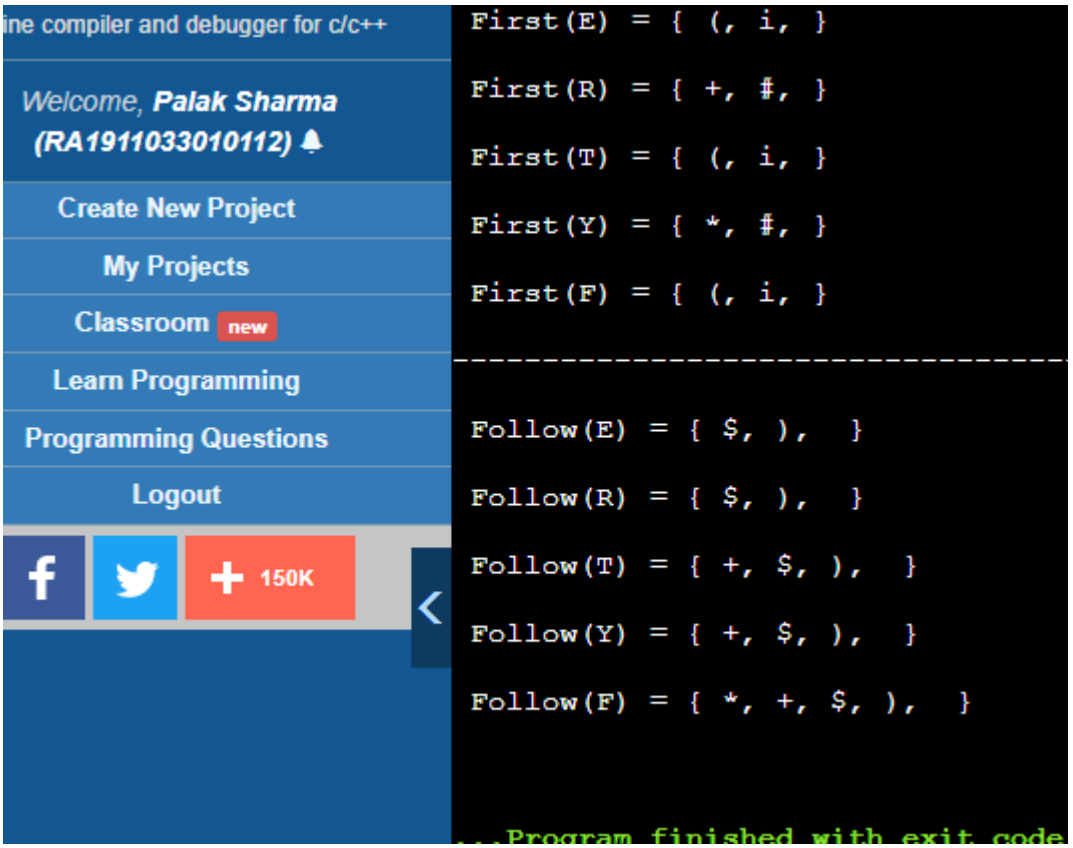
```
                                        break;
                                }
                        }
                        if(chk == 0)
                        {
                                printf("%c, ", f[i]);
                                calc_follow[point1][point2++] = f[i];
                        }
                }
                printf(" }\n\n");
                km = m;
                point1++;
        }
}

void follow(char c)
{
        int i, j;

        // Adding "$" to the follow
        // set of the start symbol
        if(production[0][0] == c) {
                f[m++] = '$';
        }
        for(i = 0; i < 10; i++)
        {
                for(j = 2;j < 10; j++)
                {
                        if(production[i][j] == c)
                        {
                                if(production[i][j+1] != '\0')
                                {
                                        // Calculate the first of the next
                                        // Non-Terminal in the production
                                        followfirst(production[i][j+1], i, (j+2));
```

```
                                }

                                if(production[i][j+1]=='\0' && c!=production[i][0])
                                {
                                        // Calculate the follow of the Non-Terminal
                                        // in the L.H.S. of the production
                                        follow(production[i][0]);
                                }
                        }
                }
        }
}

void findfirst(char c, int q1, int q2)
{
        int j;

        // The case where we
        // encounter a Terminal
        if(!(isupper(c))) {
                first[n++] = c;
        }
        for(j = 0; j < count; j++)
        {
                if(production[j][0] == c)
                {
                        if(production[j][2] == '#')
                        {
                                if(production[q1][q2] == '\0')
                                        first[n++] = '#';
                                else if(production[q1][q2] != '\0'
                                                && (q1 != 0 || q2 != 0))
                                {
                                        // Recursion to calculate First of New
                                        // Non-Terminal we encounter after epsilon
```

```
                                findfirst(production[q1][q2], q1, (q2+1));
                        }
                        else
                                first[n++] = '#';
                }
                else if(!isupper(production[j][2]))
                {
                        first[n++] = production[j][2];
                }
                else
                {
                        // Recursion to calculate First of
                        // New Non-Terminal we encounter
                        // at the beginning
                        findfirst(production[j][2], j, 3);
                }
        }
    }
}

void followfirst(char c, int c1, int c2)
{
    int k;

    // The case where we encounter
    // a Terminal
    if(!(isupper(c)))
        f[m++] = c;
    else
    {
        int i = 0, j = 1;
        for(i = 0; i < count; i++)
        {
            if(calc_first[i][0] == c)
                break;
```

```
                }

                //Including the First set of the
                // Non-Terminal in the Follow of
                // the original query
                while(calc_first[i][j] != '!')
                {
                        if(calc_first[i][j] != '#')
                        {
                                f[m++] = calc_first[i][j];
                        }
                        else
                        {
                                if(production[c1][c2] == '\0')
                                {
                                        // Case where we reach the
                                        // end of a production
                                        follow(production[c1][0]);
                                }
                                else
                                {
                                        // Recursion to the next symbol
                                        // in case we encounter a "#"
                                        followfirst(production[c1][c2], c1, c2+1);
                                }
                        }
                        j++;
                }
        }
}
```

OUTPUT :



RESULT :

The code was successfully implemented using c++ and output was recorded.

Palak Sharma
RA1911033010112

EXPERIMENT 6
PREDICTIVE PARSING

AIM :

A program for Predictive Parsing

ALGORITHM :

1. Start the program.
2. Initialize the required variables.
3. Get the number of coordinates and productions from the user. 4. Perform the following

for (each production A → α in G) {
for (each terminal a in FIRST(α))
add A → α to M[A, a];
if (ε is in FIRST(α))
for (each symbol b in FOLLOW(A))
add A → α to M[A, b];

5. Print the resulting stack.
6. Print if the grammar is accepted or not.
7. Exit the program.

CODE :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   char fin[10][20],st[10][20],ft[20][20],fol[20][20];
   int a=0,e,i,t,b,c,n,k,l=0,j,s,m,p;

   printf("enter the no. of nonterminals\n");
   scanf("%d",&n);
   printf("enter the productions in a grammar\n");
   for(i=0;i<n;i++)
      scanf("%s",st[i]);
   for(i=0;i<n;i++)
      fol[i][0]='\0';
   for(s=0;s<n;s++)
   {
      for(i=0;i<n;i++)
      {
         j=3;
         l=0;
```

```
        a=0;
        l1:if(!((st[i][j]>64)&&(st[i][j]<91)))
        {
           for(m=0;m<l;m++)
           {
              if(ft[i][m]==st[i][j])
              goto s1;
           }
           ft[i][l]=st[i][j];
           l=l+1;
           s1:j=j+1;
        }
        else
        {
           if(s>0)
           {
              while(st[i][j]!=st[a][0])
              {
                 a++;
              }
              b=0;
              while(ft[a][b]!='\0')
              {
                 for(m=0;m<l;m++)
                 {
                    if(ft[i][m]==ft[a][b])
                    goto s2;
                 }
                 ft[i][l]=ft[a][b];
                 l=l+1;
                 s2:b=b+1;
              }
           }
        }
        while(st[i][j]!='\0')
        {
           if(st[i][j]=='|')
           {
              j=j+1;
              goto l1;
           }
           j=j+1;
        }

        ft[i][l]='\0';
     }
  }
```

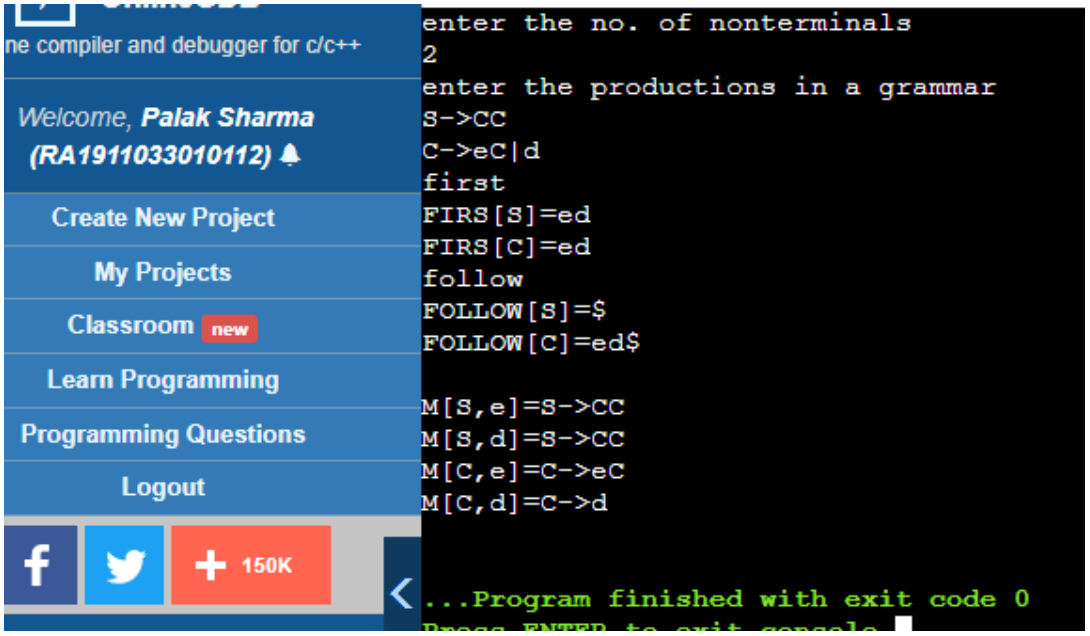
```
printf("first \n");
for(i=0;i<n;i++)
   printf("FIRS[%c]=%s\n",st[i][0],ft[i]);
fol[0][0]='$';
for(i=0;i<n;i++)
{
   k=0;
   j=3;
   if(i==0)
      l=1;
   else
      l=0;
   k1:while((st[i][0]!=st[k][j])&&(k<n))
   {
      if(st[k][j]=='\0')
      {
         k++;
         j=2;
      }
      j++;
   }

   j=j+1;
   if(st[i][0]==st[k][j-1])
   {
      if((st[k][j]!='|')&&(st[k][j]!='\0'))
      {
         a=0;
         if(!((st[k][j]>64)&&(st[k][j]<91)))
         {
            for(m=0;m<l;m++)
            {
               if(fol[i][m]==st[k][j])
               goto q3;
            }
            fol[i][l]=st[k][j];
            l++;
            q3:;
         }
         else
         {
            while(st[k][j]!=st[a][0])
            {
               a++;
            }
            p=0;
            while(ft[a][p]!='\0')
```

```
            {
               if(ft[a][p]!='@')
               {
                  for(m=0;m<l;m++)
                  {
                     if(fol[i][m]==ft[a][p])
                     goto q2;
                  }
                  fol[i][l]=ft[a][p];
                  l=l+1;
               }
               else
               e=1;
               q2:p++;
            }
            if(e==1)
            {
               e=0;
               goto a1;
            }
         }
      }
      else
      {
         a1:c=0;
         a=0;
         while(st[k][0]!=st[a][0])
         {
            a++;
         }
         while((fol[a][c]!='\0')&&(st[a][0]!=st[i][0]))
         {
            for(m=0;m<l;m++)
            {
               if(fol[i][m]==fol[a][c])
               goto q1;
            }
            fol[i][l]=fol[a][c];
            l++;
            q1:c++;
         }
      }
      goto k1;
   }
   fol[i][l]='\0';
}
printf("follow \n");
```

```
for(i=0;i<n;i++)
   printf("FOLLOW[%c]=%s\n",st[i][0],fol[i]);
printf("\n");
s=0;
for(i=0;i<n;i++)
{
   j=3;
   while(st[i][j]!='\0')
   {
      if((st[i][j-1]=='|')||(j==3))
      {
         for(p=0;p<=2;p++)
         {
            fin[s][p]=st[i][p];
         }
         t=j;
         for(p=3;((st[i][j]!='|')&&(st[i][j]!='\0'));p++)
         {
            fin[s][p]=st[i][j];
            j++;
         }
         fin[s][p]='\0';
         if(st[i][k]=='@')
         {
            b=0;
            a=0;
            while(st[a][0]!=st[i][0])
            {
               a++;
            }
            while(fol[a][b]!='\0')
            {
               printf("M[%c,%c]=%s\n",st[i][0],fol[a][b],fin[s]);
               b++;
            }
         }
         else if(!((st[i][t]>64)&&(st[i][t]<91)))
            printf("M[%c,%c]=%s\n",st[i][0],st[i][t],fin[s]);
         else
         {
            b=0;
            a=0;
            while(st[a][0]!=st[i][3])
            {
               a++;
            }
            while(ft[a][b]!='\0')
```

```
                {
                    printf("M[%c,%c]=%s\n",st[i][0],ft[a][b],fin[s]);
                    b++;
                }
            }
            s++;
        }
        if(st[i][j]=='|')
        j++;
    }
  }

}
```

OUTPUT :



RESULT : The program was successfully compiled and run.

Palak Sharma
RA1911033010112

Exp 7
SHIFT REDUCE PARSING

AIM :

To perform shift reduce parsing for the given input string.

ALGORITHM :

1. Start
2. First we take the number of production rules as well the production rules as an input form the user. We also sanitize the input from spaces.
3. For a given production rule =>
    a. We split the lexemes on the left and the right side of the "->" symbol. The left side contains a non-terminal.
    b. We again split the right side lexemes on basis of '|' character and store it in list.
    c. We store the list in a dictionary with right side (non terminal) as the key.
4. We also take the input string to test from the user and append '$' to it.
5. Set up a stack having initial symbol '$' and set up an input cursor to 0 pointing to the first character of the input.
6. We compare the top of stack with the production rules. If a found is match we reduce the elements in the stack to the corresponding non-terminal of the production rule. This is the reduce operation.
7. If no match we perform shift operation by removing a character pointed out by the input cursor and push it into the stack.
8. If operation 6 and 7 cannot be carried out we display string is rejected in the output.
9. We repeat from step 6 again until the input cursor reaches the end of the input string (i.e, upto the '$' symbol in test) and there are only 1 element in stack including the '$' symbol.
10. Print string is accepted.
11. Stop.

CODE :

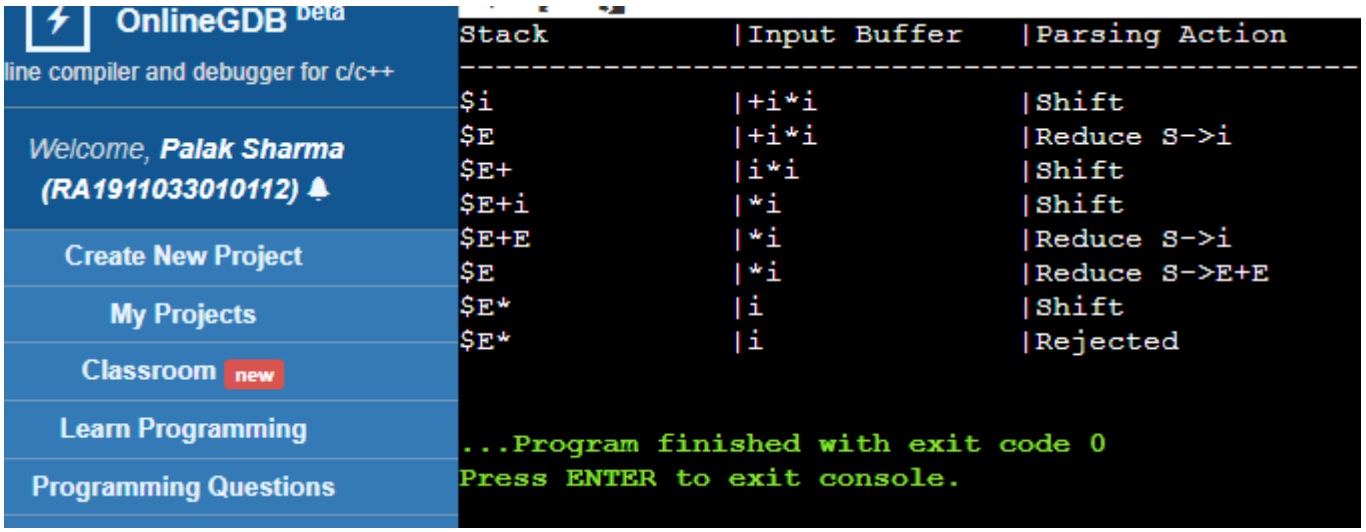
```
gram = {

    "E":["E*E","E+E","i"]
```

```
}

starting_terminal = "E"

inp = "i+i*i"



stack = "$"

print(f'{"Stack": <15}'+"|"+f'{"Input Buffer": <15}'+"|"+f'Parsing Action')

print(f'{"-":-<50}')



while True:

    action = True

    i = 0

    while i<len(gram[starting_terminal]):

        if gram[starting_terminal][i] in stack:

            stack = stack.replace(gram[starting_terminal][i],starting_terminal)
```

```
                    print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Reduce
S->{gram[starting_terminal][i]}')

                    i=-1

                    action = False

            i+=1

        if len(inp)>1:

            stack+=inp[0]

            inp=inp[1:]

            print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Shift')

            action = False

if inp == "$" and stack == ("$"+starting_terminal):

            print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Accepted')

            break

if action:

            print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Rejected')
```

break

OUTPUT :



RESULT :

Implementation of shift reduce parsing for the given input string is done.

Palak Sharma
RA1911033010112

EXP 8
LEADING AND TRAILING

AIM :

A program to implement Leading and Trailing

ALGORITHM :

1. For Leading, check for the first non-terminal.
2. If found, print it.
3. Look for next production for the same non-terminal.
4. If not found, recursively call the procedure for the single non-terminal present before the comma or End Of Production String.
5. Include it's results in the result of this non-terminal.
6. For trailing, we compute same as leading but we start from the end of the production to the beginning.
7. Stop

CODE :

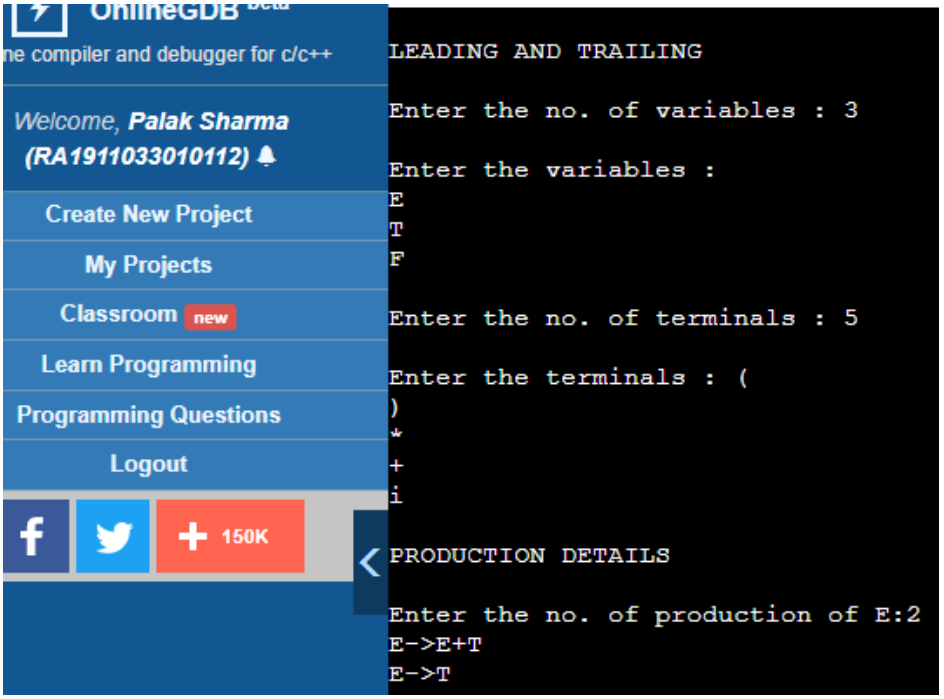
```
#include<iostream>
#include<conio.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
using namespace std;

int vars,terms,i,j,k,m,rep,count,temp=-1;
char var[10],term[10],lead[10][10],trail[10][10];
struct grammar
{
        int prodno;
        char lhs,rhs[20][20];
}gram[50];
void get()
{
        cout<<"\nLEADING AND TRAILING\n";
        cout<<"\nEnter the no. of variables : ";
        cin>>vars;
        cout<<"\nEnter the variables : \n";
```

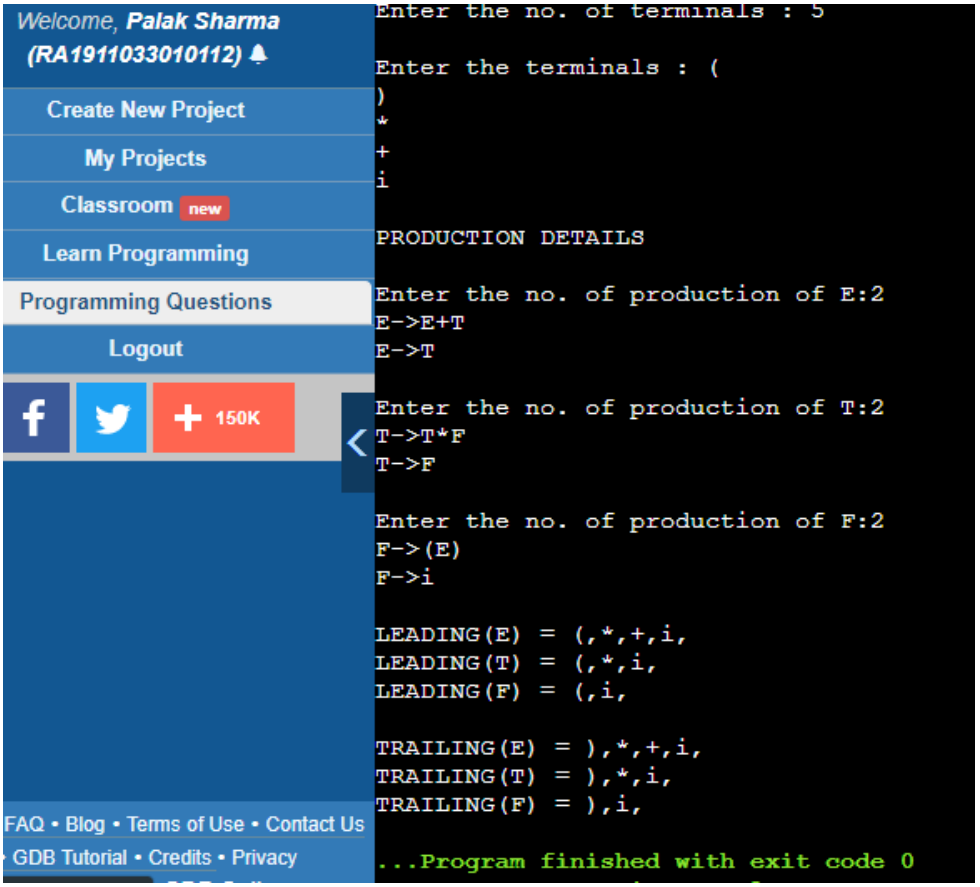
```
	for(i=0;i<vars;i++)
	{
		cin>>gram[i].lhs;
		var[i]=gram[i].lhs;
	}
	cout<<"\nEnter the no. of terminals : ";
	cin>>terms;
	cout<<"\nEnter the terminals : ";
	for(j=0;j<terms;j++)
		cin>>term[j];
	cout<<"\nPRODUCTION DETAILS\n";
	for(i=0;i<vars;i++)
	{
		cout<<"\nEnter the no. of production of "<<gram[i].lhs<<":";
		cin>>gram[i].prodno;
		for(j=0;j<gram[i].prodno;j++)
		{
			cout<<gram[i].lhs<<"->";
			cin>>gram[i].rhs[j];
		}
	}
}
void leading()
{
	for(i=0;i<vars;i++)
	{
		for(j=0;j<gram[i].prodno;j++)
		{
			for(k=0;k<terms;k++)
			{
				if(gram[i].rhs[j][0]==term[k])
					lead[i][k]=1;
				else
				{
					if(gram[i].rhs[j][1]==term[k])
						lead[i][k]=1;
				}
			}
		}
	}
	for(rep=0;rep<vars;rep++)
	{
		for(i=0;i<vars;i++)
		{
			for(j=0;j<gram[i].prodno;j++)
			{
				for(m=1;m<vars;m++)
				{
					if(gram[i].rhs[j][0]==var[m])
					{
						temp=m;
						goto out;
					}
				}
```

```
                                out:
                                for(k=0;k<terms;k++)
                                {
                                        if(lead[temp][k]==1)
                                                lead[i][k]=1;
                                }
                        }
                }
        }
}
void trailing()
{
        for(i=0;i<vars;i++)
        {
                for(j=0;j<gram[i].prodno;j++)
                {
                        count=0;
                        while(gram[i].rhs[j][count]!='\x0')
                                count++;
                        for(k=0;k<terms;k++)
                        {
                                if(gram[i].rhs[j][count-1]==term[k])
                                        trail[i][k]=1;
                                else
                                {
                                        if(gram[i].rhs[j][count-2]==term[k])
                                                trail[i][k]=1;
                                }
                        }
                }
        }
        for(rep=0;rep<vars;rep++)
        {
                for(i=0;i<vars;i++)
                {
                        for(j=0;j<gram[i].prodno;j++)
                        {
                                count=0;
                                while(gram[i].rhs[j][count]!='\x0')
                                        count++;
                                for(m=1;m<vars;m++)
                                {
                                        if(gram[i].rhs[j][count-1]==var[m])
                                                temp=m;
                                }
                                for(k=0;k<terms;k++)
                                {
                                        if(trail[temp][k]==1)
                                                trail[i][k]=1;
                                }
                        }
                }
        }
}
```

```
void display()
{
	for(i=0;i<vars;i++)
	{
		cout<<"\nLEADING("<<gram[i].lhs<<") = ";
		for(j=0;j<terms;j++)
		{
			if(lead[i][j]==1)
				cout<<term[j]<<",";
		}
	}
	cout<<endl;
	for(i=0;i<vars;i++)
	{
		cout<<"\nTRAILING("<<gram[i].lhs<<") = ";
		for(j=0;j<terms;j++)
		{
			if(trail[i][j]==1)
				cout<<term[j]<<",";
		}
	}
}
int main()
{

	get();
	leading();
	trailing();
	display();

}
```

OUTPUT :





RESULT :

The program was successfully compiled and run.

Palak Sharma
RA1911033010112

EXP 9

Computation of LR(0) Items

AIM : A program to implement LR(0) items

ALGORITHM :

1. Start.
2. Create structure for production with LHS and RHS.
3. Open file and read input from file.
4. Build state 0 from extra grammar Law S' -> S $ that is all start symbol of grammar and one Dot (.) before S symbol.
5. If Dot symbol is before a non-terminal, add grammar laws that this non-terminal is in Left Hand Side of that Law and set Dot in before of first part of Right Hand Side.
6. If state exists (a state with this Laws and same Dot position), use that instead.
7. Now find set of terminals and non-terminals in which Dot exist in before.
8. If step 7 Set is non-empty go to 9, else go to 10.
9. For each terminal/non-terminal in set step 7 create new state by using all grammar law that Dot position is before of that terminal/non-terminal in reference state by increasing Dot point to next part in Right Hand Side of that laws.
10. Go to step 5.
11. End of state building.
12. Display the output.
13. End.

Program:

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
char
prod[20][20],listofvar[26]="ABCDEFGHIJKLMNOPQR";
int novar=1,i=0,j=0,k=0,n=0,m=0,arr[30];
```

```
int noitem=0;
struct Grammar
{
	char lhs;
	char rhs[8];
}g[20],item[20],clos[20][10];

int isvariable(char variable)
{
	for(int i=0;i<novar;i++)
		if(g[i].lhs==variable)
			return i+1;
	return 0;
}
void findclosure(int z, char a)
{
	int n=0,i=0,j=0,k=0,l=0;
	for(i=0;i<arr[z];i++)
	{
		for(j=0;j<strlen(clos[z][i].rhs);j++)
		{
			if(clos[z][i].rhs[j]=='.' && clos[z][i].rhs[j+1]==a)
			{
				clos[noitem][n].lhs=clos[z][i].lhs;
				strcpy(clos[noitem][n].rhs,clos[z][i].rhs);
				char temp=clos[noitem][n].rhs[j];
				clos[noitem][n].rhs[j]=clos[noitem][n].rhs[j+1];
				clos[noitem][n].rhs[j+1]=temp;
				n=n+1;
			}
		}
```

```
	}
	for(i=0;i<n;i++)
	{
		for(j=0;j<strlen(clos[noitem][i].rhs);j++)
		{
			if(clos[noitem][i].rhs[j]=='.' && isvariable(clos[noitem][i].rhs[j+1])>0)
			{
				for(k=0;k<novar;k++)
				{
					if(clos[noitem][i].rhs[j+1]==clos[0][k].lhs)
					{
						for(l=0;l<n;l++)
							if(clos[noitem][l].lhs==clos[0][k].lhs
							&&
strcmp(clos[noitem][l].rhs,clos[0][k].rhs)==0)
								break;
						if(l==n)
						{
							clos[noitem][n].lhs=clos[0][k].lhs;
						strcpy(clos[noitem][n].rhs,clos[0][k].rhs);
							n=n+1;
						}
					}
				}
			}
		}
	}
	arr[noitem]=n;
	int flag=0;
	for(i=0;i<noitem;i++)
	{
		if(arr[i]==n)
```

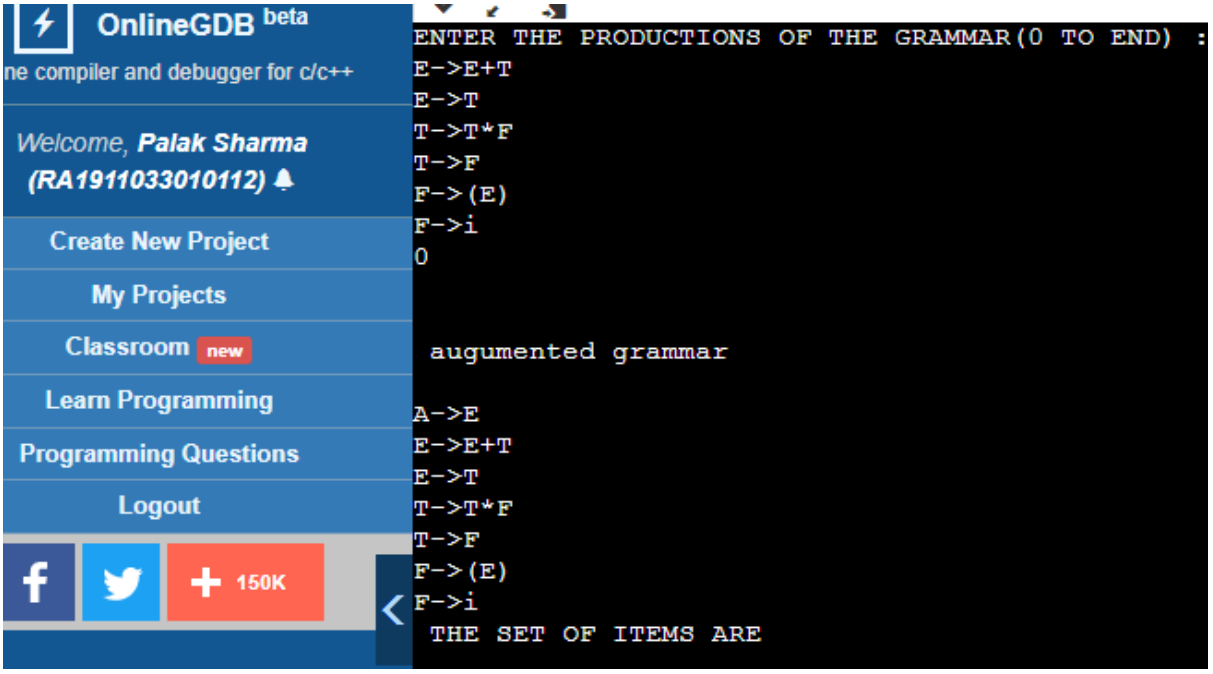
```
		{
			for(j=0;j<arr[i];j++)
			{
				int c=0;
				for(k=0;k<arr[i];k++)
					if(clos[noitem][k].lhs==clos[i][k].lhs &&
strcmp(clos[noitem][k].rhs,clos[i][k].rhs)==0)
						c=c+1;
				if(c==arr[i])
				{
					flag=1;
					goto exit;
				}
			}
		}
	}
	exit:;
	if(flag==0)
		arr[noitem++]=n;
}

void main()
{
	clrscr();
	cout<<"ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :\n";
	do
	{
		cin>>prod[i++];
	}while(strcmp(prod[i-1],"0")!=0);
	for(n=0;n<i-1;n++)
	{
```

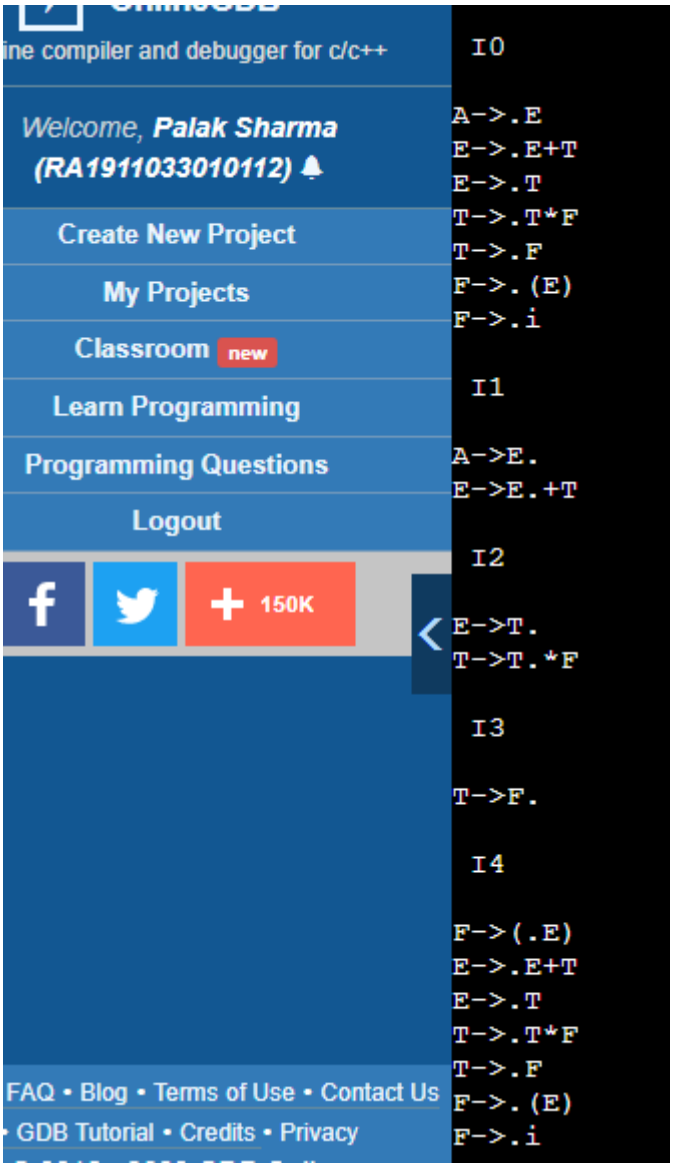
```
            m=0;
            j=novar;
            g[novar++].lhs=prod[n][0];
            for(k=3;k<strlen(prod[n]);k++)
            {
                  if(prod[n][k] != '|')
                  g[j].rhs[m++]=prod[n][k];
                  if(prod[n][k]=='|')
                  {
                        g[j].rhs[m]='\0';
                        m=0;
                        j=novar;
                        g[novar++].lhs=prod[n][0];
                  }
            }
      }
      for(i=0;i<26;i++)
            if(!isvariable(listofvar[i]))
                  break;
g[0].lhs=listofvar[i];
      char temp[2]={g[1].lhs,'\0'};
      strcat(g[0].rhs,temp);
      cout<<"\n\n augumented grammar \n";
      for(i=0;i<novar;i++)
            cout<<endl<<g[i].lhs<<"->"<<g[i].rhs<<" ";
      getch();
      for(i=0;i<novar;i++)
      {
            clos[noitem][i].lhs=g[i].lhs;
            strcpy(clos[noitem][i].rhs,g[i].rhs);
            if(strcmp(clos[noitem][i].rhs,"ε")==0)
```

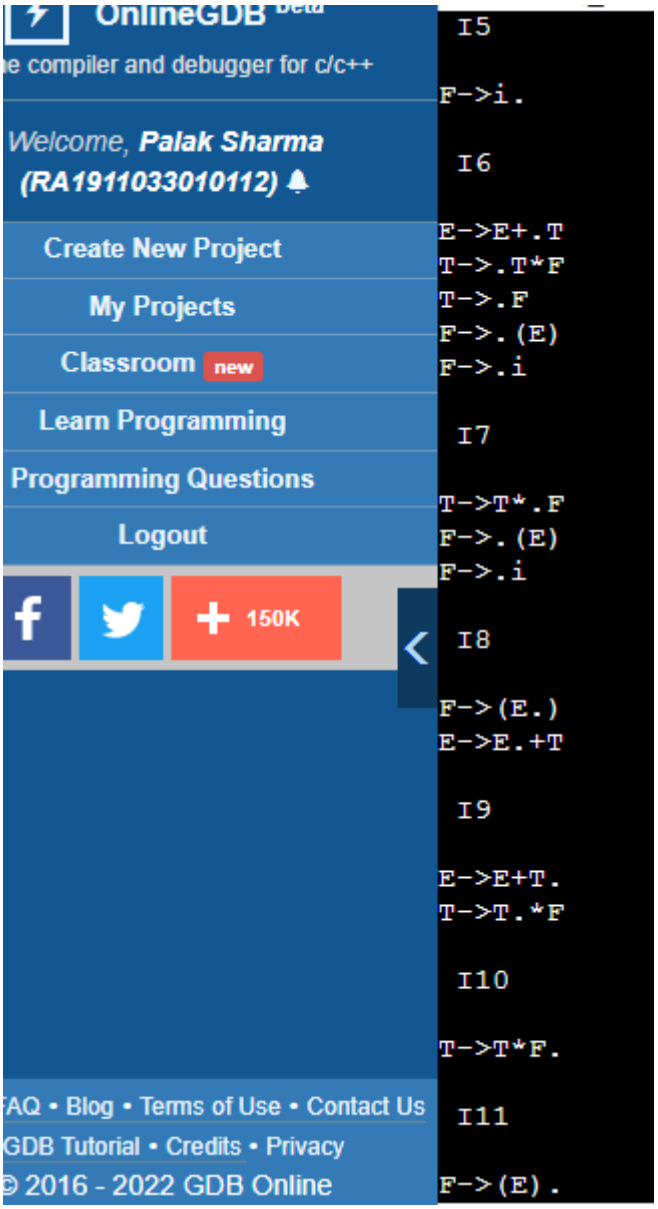
```
                strcpy(clos[noitem][i].rhs,".");
        else
        {
                for(int j=strlen(clos[noitem][i].rhs)+1;j>=0;j--)
                        clos[noitem][i].rhs[j]=clos[noitem][i].rhs[j
                        -1];
                clos[noitem][i].rhs[0]='.';
        }
}
arr[noitem++]=novar;
for(int z=0;z<noitem;z++)
{
        char list[10];
        int l=0;
        for(j=0;j<arr[z];j++)
        {
                for(k=0;k<strlen(clos[z][j].rhs)-1;k++)
                {
                        if(clos[z][j].rhs[k]=='.')
                        {
                                for(m=0;m<l;m++)
                                        if(list[m]==clos[z][j].rhs[k+1])
                                                break;
                                if(m==l)
                                        list[l++]=clos[z][j].rhs[k+1];
                        }
                }
        }
        for(int x=0;x<l;x++)
                findclosure(z,list[x]);
}
cout<<"\n THE SET OF ITEMS ARE \n\n";
```

```
        for(z=0;z<noitem;z++)
        {
                cout<<"\n I"<<z<<"\n\n";
                for(j=0;j<arr[z];j++)
                        cout<<clos[z][j].lhs<<"->"<<clos[z][j].rhs<<"
                \n"; getch();
        }
        getch();
}
```

OUTPUT :




ine compiler and debugger for c/c++
Welcome, Palak Sharma
(RA1911033010112)
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Logout
150K
FAQ • Blog • Terms of Use • Contact Us
• GDB Tutorial • Credits • Privacy
I0
A->.E
E->.E+T
E->.T
T->.T*F
T->.F
F->.(E)
F->.i
I1
A->E.
E->E.+T
I2
E->T.
T->T.*F
I3
T->F.
I4
F->(.E)
E->.E+T
E->.T
T->.T*F
T->.F
F->.(E)
F->.i



RESULT : The program was successfully compiled and run.

Palak Sharma
RA1911033010112

EXP 10

Intermediate code generation – Postfix, Prefix

AIM :

A program to implement Intermediate code generation – Postfix, Prefix.

ALGORITHM :

1. Declare a set of operators.
2. Initialize an empty stack.
3. To convert INFIX to POSTFIX follow the following steps
4. Scan the infix expression from left to right.
5. If the scanned character is an operand, output it.
6. Else, If the precedence of the scanned operator is greater than the precedence of the operator in the stack(or the stack is empty or the stack contains a '('), push it.
7. Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack.
8. If the scanned character is an '(', push it to the stack.
9. If the scanned character is an ')', pop the stack and output it until a '(' is encountered, and discard both the parenthesis.
10. Pop and output from the stack until it is not empty.
11. To convert INFIX to PREFIX follow the following steps
12. First, reverse the infix expression given in the problem.
13. Scan the expression from left to right.
14. Whenever the operands arrive, print them.
15. If the operator arrives and the stack is found to be empty, then simply push the operator into the stack.
16. Repeat steps 6 to 9 until the stack is empty

CODE :

```
OPERATORS = set(['+', '-', '*', '/', '(', ')'])

PRI = {'+': 1, '-': 1, '*': 2, '/': 2}


### INFIX ===> POSTFIX ###

def infix_to_postfix(formula):
    stack = []  # only pop when the coming op has priority

    output = ''

    for ch in formula:

        if ch not in OPERATORS:

            output += ch

        elif ch == '(':

            stack.append('(')

        elif ch == ')':

            while stack and stack[-1] != '(':
                output += stack.pop()

            stack.pop()  # pop '('

        else:

            while stack and stack[-1] != '(' and PRI[ch] <= PRI[stack[-1]]:
                output += stack.pop()
```

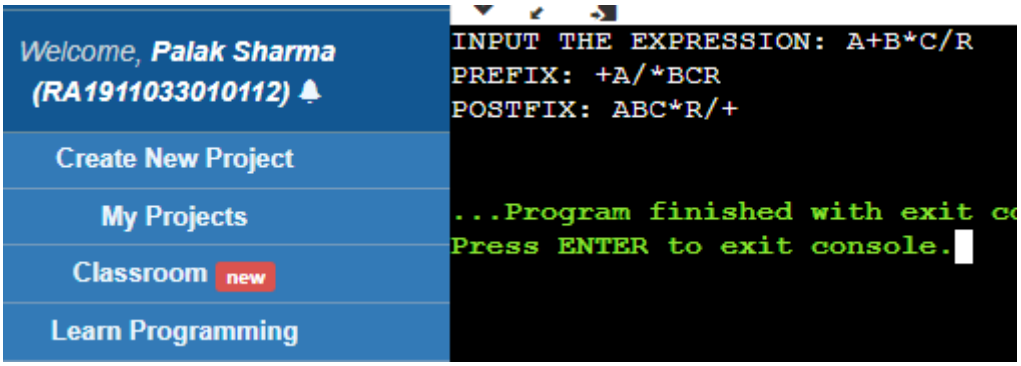
```
            stack.append(ch)

            # leftover

    while stack:
        output += stack.pop()

    print(f'POSTFIX: {output}')

    return output


### INFIX ===> PREFIX ###

def infix_to_prefix(formula):
    op_stack = []

    exp_stack = []

    for ch in formula:

        if not ch in OPERATORS:

            exp_stack.append(ch)

        elif ch == '(':

            op_stack.append(ch)

        elif ch == ')':

            while op_stack[-1] != '(':
                op = op_stack.pop()

                a = exp_stack.pop()
```

```
            b = exp_stack.pop()

            exp_stack.append(op + b + a)

        op_stack.pop()  # pop '('

    else:

        while op_stack and op_stack[-1] != '(' and PRI[ch] <= PRI[op_stack[-1]]:
            op = op_stack.pop()

            a = exp_stack.pop()

            b = exp_stack.pop()

            exp_stack.append(op + b + a)

        op_stack.append(ch)

        # leftover

while op_stack:
    op = op_stack.pop()

    a = exp_stack.pop()

    b = exp_stack.pop()

    exp_stack.append(op + b + a)

print(f'PREFIX: {exp_stack[-1]}')

return exp_stack[-1]

expres = input("INPUT THE EXPRESSION: ")
```

pre = infix_to_prefix(expres)

pos = infix_to_postfix(expres)

OUTPUT :



RESULT :

The program was successfully compiled and run.

Palak Sharma
RA1911033010112

EXP 11
Intermediate code generation – Quadruple, Triple, Indirect triple

AIM : Intermediate code generation – Quadruple, Triple, Indirect triple

ALGORITHM :

The algorithm takes a sequence of three-address statements as input. For each three address statements of the form a:= b op c perform the various actions. These are as follows:

1. Invoke a function getreg to find out the location L where the result of computation b op c should be stored.
2. Consult the address description for y to determine y'. If the value of y currently in memory and register both then prefer the register y' . If the value of y is not already in L then generate the instruction MOV y' , L to place a copy of y in L.
3. Generate the instruction OP z' , L where z' is used to show the current location of z. if z is in both then prefer a register to a memory location. Update the address descriptor of x to indicate that x is in location L. If x is in L then update its descriptor and remove x from all other descriptors.
4. If the current value of y or z have no next uses or not live on exit from the block or in register then alter the register descriptor to indicate that after execution of x : = y op z those register will no longer contain y or z.

CODE :

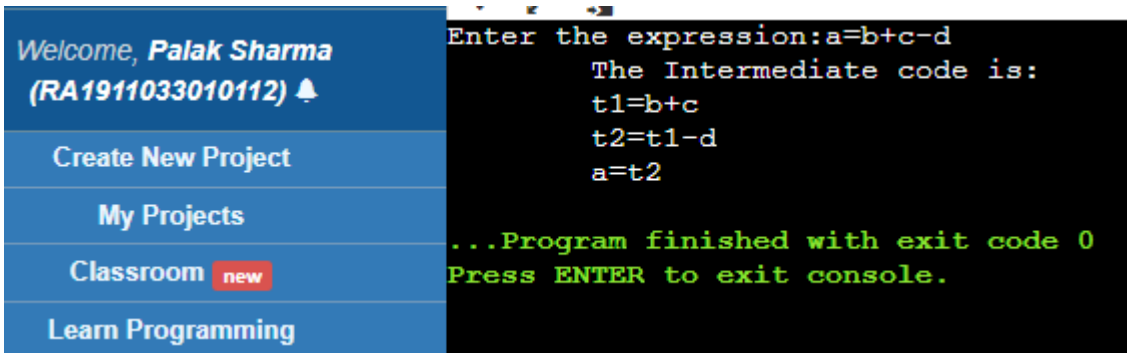
```
#include<stdio.h>

#include<ctype.h>

#include<stdlib.h>

#include<string.h>

void small();

void dove(int i);

int p[5]={0,1,2,3,4},c=1,i,k,l,m,pi;

char sw[5]={'=','-','+','/','*'},j[20],a[5],b[5],ch[2];

void main()
```

```
{
    printf("Enter the expression:");
    scanf("%s",j);
    printf("\tThe Intermediate code is:\n");
    small();
}
void dove(int i)
{
    a[0]=b[0]='\0';
    if(!isdigit(j[i+2])&&!isdigit(j[i-2]))
    {
        a[0]=j[i-1];
        b[0]=j[i+1];
    }
    if(isdigit(j[i+2]))
    {
        a[0]=j[i-1];
        b[0]='t';
        b[1]=j[i+2];
    }
    if(isdigit(j[i-2]))
    {
        b[0]=j[i+1];
        a[0]='t';
        a[1]=j[i-2];
        b[1]='\0';
    }
```

```
    if(isdigit(j[i+2]) &&isdigit(j[i-2]))
    {
       a[0]='t';
       b[0]='t';
       a[1]=j[i-2];
       b[1]=j[i+2];
       sprintf(ch,"%d",c);
       j[i+2]=j[i-2]=ch[0];
    }
    if(j[i]=='*')
       printf("\tt%d=%s*%s\n",c,a,b);
    if(j[i]=='/')
       printf("\tt%d=%s/%s\n",c,a,b);
    if(j[i]=='+')
       printf("\tt%d=%s+%s\n",c,a,b);if(j[i]=='-')
    printf("\tt%d=%s-%s\n",c,a,b);
    if(j[i]=='=')
       printf("\t%c=t%d",j[i-1],--c);
    sprintf(ch,"%d",c);
    j[i]=ch[0];
    c++;
    small();
}
void small()
{
    pi=0;l=0;
    for(i=0;i<strlen(j);i++)
```

```
    {
       for(m=0;m<5;m++)
       if(j[i]==sw[m])
       if(pi<=p[m])
       {
          pi=p[m];
          l=1;
          k=i;
       }
    }
    if(l==1)
       dove(k);
    else
       exit(0);
}
```

OUTPUT :



RESULT : The program was successfully compiled and run.

Palak Sharma
RA1911033010112

EXP 12
Implementation of DAG

AIM : A program to implement DAG.

ALGORITHM :

1. The leaves of a graph are labeled by a unique identifier and that identifier can be variablenames or constants.
2. Interior nodes of the graph are labeled by an operator symbol.
3. Nodes are also given a sequence of identifiers for labels to store the computed value.
4. If y operand is undefined then create node(y).
5. If z operand is undefined then for case(i) create node(z).
6. For case(i), create node(OP) whose right child is node(z) and left child is node(y).
7. For case(ii), check whether there is node(OP) with one child node(y).
8. For case(iii), node n will be node(y).
9. For node(x) delete x from the list of identifiers. Append x to attached identifiers list forthe node n found in step 2. Finally set node(x) to n.

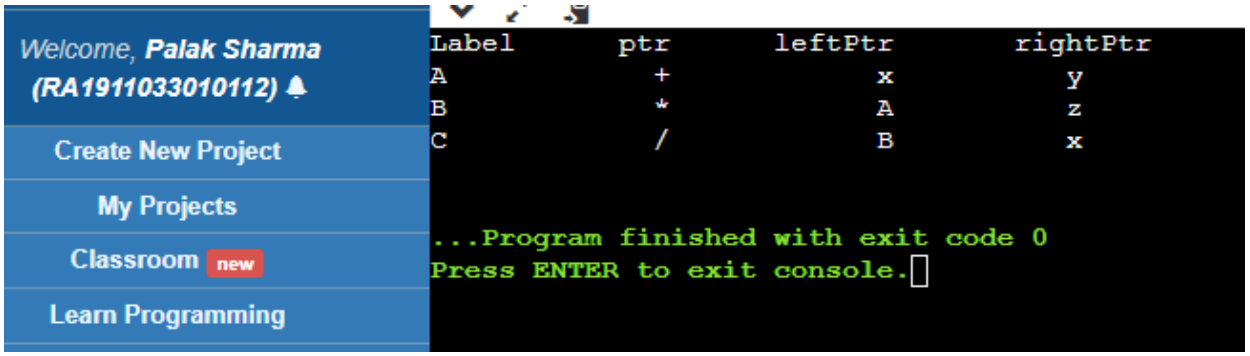
CODE :

```
#include<iostream>
#include<string>
#include<unordered_map>
using namespace std;

class DAG
{
        public:
        char label;
        char data;
        DAG* left;
        DAG* right;
        DAG(char x)
        {
                label='_';
                data=x;
                left=NULL;
                right=NULL;
        }
        DAG(char lb, char x, DAG* lt, DAG* rt)
        {
                label=lb;
```

```
                data=x;
                left=lt;
                right=rt;
        }
};
int main()
{
        int n;
        n=3;
        string st[n];
        st[0]="A=x+y";
        st[1]="B=A*z";
        st[2]="C=B/x";
        unordered_map<char, DAG*> labelDAGNode;
        for(int i=0;i<3;i++)
        {
                string stTemp=st[i];
                for(int j=0;j<5;j++)
                {
                        char tempLabel = stTemp[0];
                        char tempLeft = stTemp[2];
                        char tempData = stTemp[3];
                        char tempRight = stTemp[4];
                        DAG* leftPtr;
                        DAG* rightPtr;
                        if(labelDAGNode.count(tempLeft) == 0)
                        {
                                leftPtr = new DAG(tempLeft);
                        }
                        else
                        {
                                leftPtr = labelDAGNode[tempLeft];
                        }
                                if(labelDAGNode.count(tempRight) == 0)
                        {
                                rightPtr = new DAG(tempRight);
                        }
                        else
                        {
                                rightPtr = labelDAGNode[tempRight];
                        }
                        DAG* nn = new DAG(tempLabel,tempData,leftPtr,rightPtr);
                        labelDAGNode.insert(make_pair(tempLabel,nn));
                        }
                }
                cout<<"Label ptr leftPtr rightPtr"<<endl;
                for(int i=0;i<n;i++)
                {
                        DAG* x=labelDAGNode[st[i][0]];
                        cout<<st[i][0]<<" "<<x->data<<" ";
                        if(x->left->label=='_')cout<<x->left->data;
                        else cout<<x->left->label;
                        cout<<" ";
                        if(x->right->label=='_')cout<<x->right->data;
```

```
                else cout<<x->right->label;
                cout<<endl;
        }
        return 0;
}
```

OUTPUT :



RESULT : The program was successfully compiled and run.

Palak Sharma
RA1911033010112

EXP 13
Implementation of Anyone Storage SLO-2 Allocation Strategies (heap,stack,static)

AIM :

To implement various storage allocation techniques algorithms

ALGORITHM :

Static storage allocation

● In static allocation, names are bound to storage locations.

● If memory is created at compile time then the memory will be created instaticarea and only once.

● Static allocation supports the dynamic data structure that means memory is created only at compile time and deallocated after program completion.

● The drawback with static storage allocation is that the size and position of dataobjects should be known at compile time.

● Another drawback is restriction of the recursion procedure. Stack Storage Allocation

● In static storage allocation, storage is organized as a stack.

● An activation record is pushed into the stack when activation begins and it is popped when the activation ends.

● Activation record contains the locals so that they are bound to fresh storage for each activation record. The value of locals is deleted when the activation ends.

● It works on the basis of last-in-first-out (LIFO) and this allocation supports the recursion process. Heap Storage Allocation.

● Heap allocation is the most flexible allocation scheme.

● Allocation and deallocation of memory can be done at any time and at any place depending upon the user's requirement.

● Heap allocation is used to allocate memory to the variables dynamically and when the variables are no more used then claim it back.

● Heap storage allocation supports the recursion process.

CODE :

```
class memory_allocation:
    def create_instance_memory_blocks(self,l):
        self.i=0
        self.block=l
        self.pdict ={}
        for self.y in self.block:
            print("Memory block :{0} | Size: {1}".format(self.i,self.y))
            self.i+=1
            self.pdict[self.y]=0
        print("Total {0} Memory blocks Created!".format(self.i))

    def process_to_be_allocated(self,list):
        self.total_processlist=list
        print()
        self.i=0
        self.bdict={}
        for self.x in self.total_processlist:
            print("Process :{0} | Size: {1}".format(self.i,self.x))
            self.bdict[self.x]=0
            self.i+=1
        print("Total {0} Processes needs to be allocated".format(self.i))

    def first_fit(self):
        print()
        print("Using First Fit Algorithm")
```

```
        self.i=0
        for self.x in self.total_processlist:
            self.j=0
            for self.y in self.block:
                if (self.x <= self.y) and (self.pdict[self.y]!=1):
                    self.pdict[self.y]=1
                    self.bdict[self.x]=1
                    print("Process :{0} |Size :{1} allocated into Memory block :{2} |Size
:{3}".format(self.i,self.x,self.j,self.y))
                    break
                self.j+=1
            self.i+=1


    def worst_fit(self):
        print()
        print("Using worst Fit Algorithm")
        self.i=0
        for self.x in self.total_processlist:
            self.j=0
            for self.y in self.block:
                if (self.x <= self.y) and (self.pdict[self.y]!=1):
                    if(self.isbiggest(self.y)):
                        self.pdict[self.y]=1
                        self.bdict[self.x]=1
                        print("Process :{0} |Size :{1} allocated into Memory block :{2} |Size
:{3}".format(self.i,self.x,self.j,self.y))
                        break
```

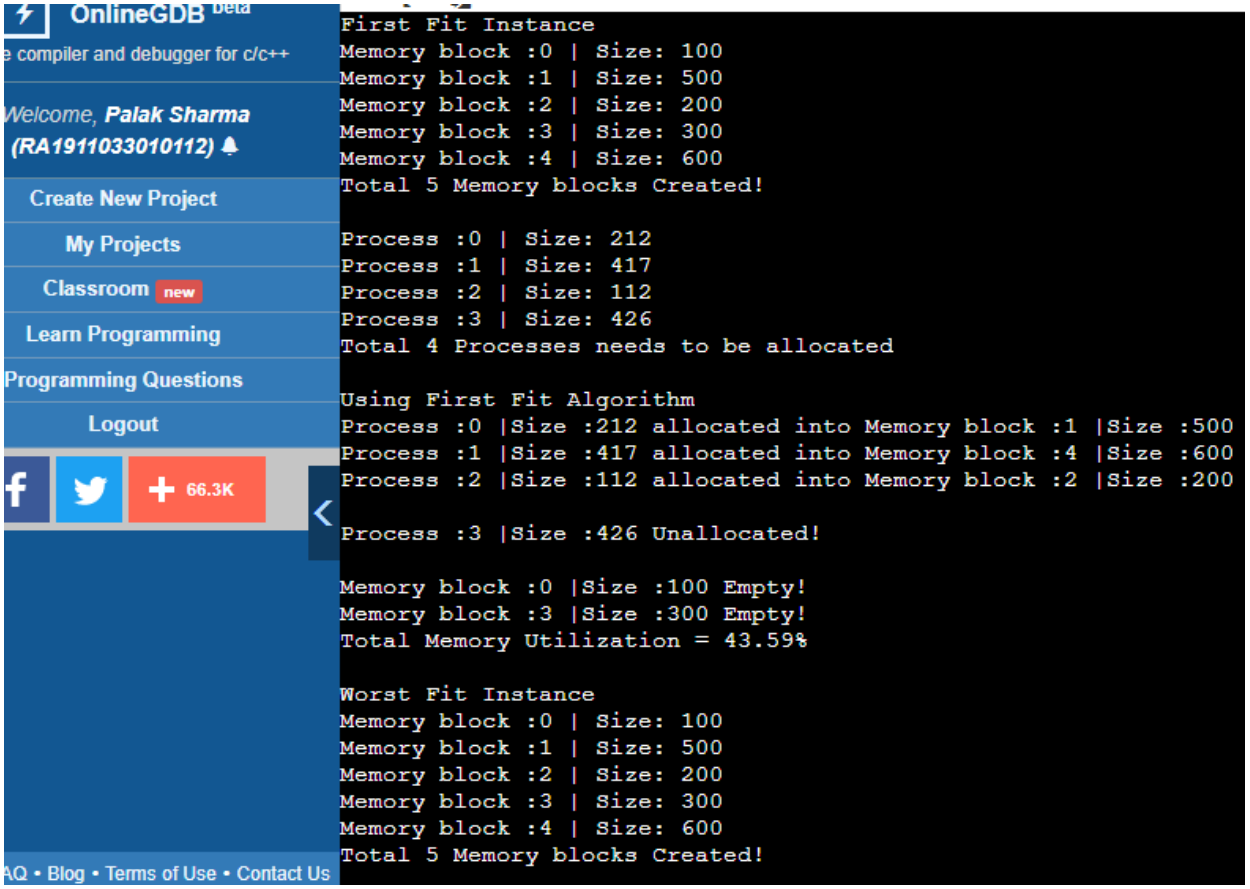
```
            self.j+=1
        self.i+=1
    def best_fit(self):
        print()
        print("Using best Fit Algorithm")
        self.i=0
        for self.x in self.total_processlist:
            self.j=0
            for self.y in self.block:
                if (self.x <= self.y) and (self.pdict[self.y]!=1):
                    if(self.issmallest(self.y)):
                        self.pdict[self.y]=1
                        self.bdict[self.x]=1
                        print("Process :{0} |Size :{1} allocated into Memory block :{2} |Size
:{3}".format(self.i,self.x,self.j,self.y))
                        break
                self.j+=1
            self.i+=1

    def issmallest(self,x1):
        self.flag =True
        self.x1=x1
        for self.y1 in self.block:
            if self.y1<self.x1 and self.pdict[self.y1]!=1 and self.y1>=self.x:
                self.flag=False
        return self.flag
```

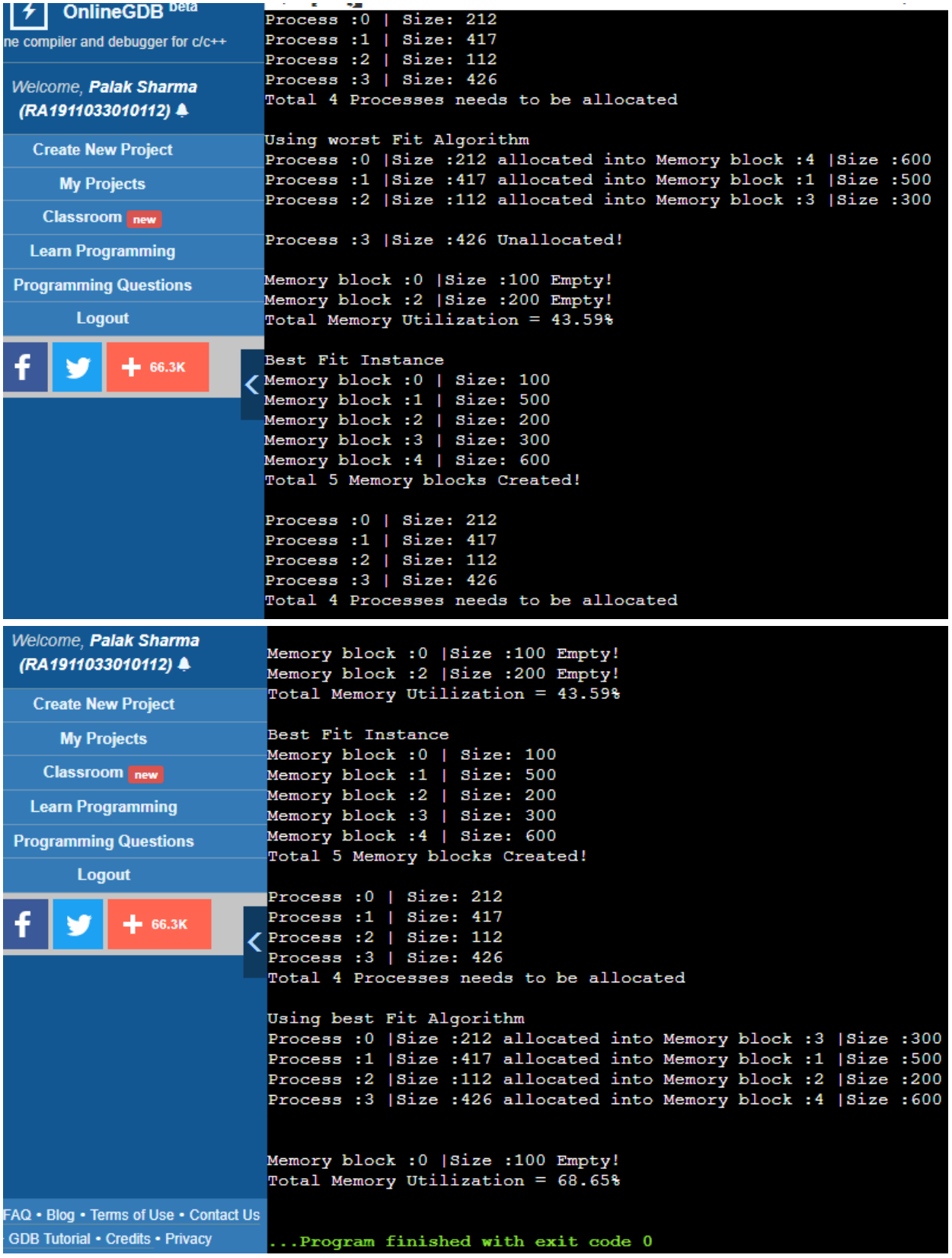
```
def isbiggest(self,x1):
    self.flag =True
    self.x1=x1
    for self.y1 in self.block:
        if self.y1>self.x1 and self.pdict[self.y1]!=1:
            self.flag=False
    return self.flag


def unallocated_processes(self):
        print()
        self.i=0
        for self.x in self.total_processlist:
            if (self.bdict[self.x]!=1):
                print("Process :{0} |Size :{1} Unallocated! ".format(self.i,self.x))
            self.i+=1
def unallocated_blocks(self):
        print()
        self.j=0
        for self.y in self.block:
            if (self.pdict[self.y]!=1):
                print("Memory block :{0} |Size :{1} Empty! ".format(self.j,self.y))
            self.j+=1
def memory_utilization(self):
    self.total = 0
    self.sum=0
    for self.z in self.total_processlist:
        if self.bdict[self.z]!=0:
```

```
            self.sum = self.sum + self.z
        for self.z in self.block:
            self.total = self.total + self.z
        print("Total Memory Utilization = {0:.2f}%".format(self.sum*100/self.total))



if __name__ == '__main__':
    bl = [100,500,200,300,600]
    pl = [212,417,112,426]
    print("First Fit Instance")
    obj = memory_allocation()
    obj.create_instance_memory_blocks(bl)
    obj.process_to_be_allocated(pl)
    obj.first_fit()
    obj.unallocated_processes()
    obj.unallocated_blocks()
    obj.memory_utilization()
    print()
    print("Worst Fit Instance")
    obj1 = memory_allocation()
    obj1.create_instance_memory_blocks(bl)
    obj1.process_to_be_allocated(pl)
    obj1.worst_fit()
    obj1.unallocated_processes()
    obj1.unallocated_blocks()
    obj1.memory_utilization()
    print()
```

```
print("Best Fit Instance")

obj2 = memory_allocation()

obj2.create_instance_memory_blocks(bl)

obj2.process_to_be_allocated(pl)

obj2.best_fit()

obj2.unallocated_processes()

obj2.unallocated_blocks()

obj2.memory_utilization()
```

OUTPUT :





RESULT : The program was successfully compiled and run.