



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

S. NO.	PAGE NO.	DATE	SUBJECT	SIGNATURE
1.	3-16	16.04.2022	Assignments	
2.	17-37	08.02.2022	HackerRank 20 Regex Problems	
3.	37-63	31.03.2022	HackerRank Remaining Problems	
4.	64		Experiments	
5.	65-67	10.01.2022	Implementation of Lexical Analysis	
6.	68-72	27.01.2022	Regular Expression of nfa	
7.	73-76	08.02.2022	Conversion of nfa to dfa	
8.	77-87	17.02.2022	Left Factoring and Left Recursion	
9.	86-95	22.02.2022	Computation of First and Follow	
10.	96-101	04.03.2022	Predictive Parsing	
11.	102-107	10.03.2022	Shift Reduce Parsing	
12.	108-110	17.03.2022	Leading and Trailing	
13.	111-119	24.03.2022	Computation of LR(0) Items	
14.	120-124	31.03.2022	Intermediate Code generation - Postfix, Prefix	
15.	125-128	31.03.2022	Indirect - Quadruple, Triple, Indirect Triple	
16.	129-131	07.04.2022	Implementation of DAG	
17.	132-139	07.04.2022	Implementation of One Storage SLO-2 Allocation Strategies	

# **ASSIGNMENTS**

## UNIT 1

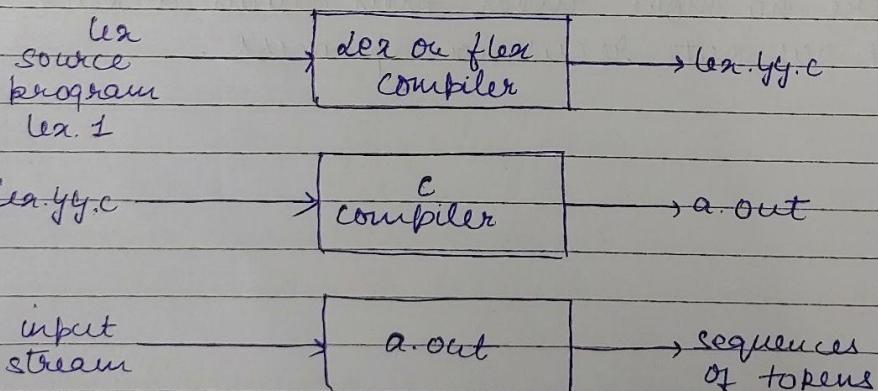
Palak Sharma  
RA1911033010112

PAGE NO.:  
DATE: / /

### UNIT - 1

#### Scanner generator tool

- The lex and flex scanner generators
- Generates lexical analyzers from the input that consists of regular expression description based on tokens of new language
- lex and its newer cousin flex are scanner generators.
- It systematically translates regular definitions into C source code for efficient scanning.
- Generated code is easy to integrate in C applications.



#### → lex specifications

- A lex specification of three parts:  
regular definitions, C declaration in ~~%%~~ % %  
%% %%

translation rules

% %

user-defined auxiliary produces...

- The translation rules are of the form

$P_1 \{ \text{action}_1 \}$

$P_2 \{ \text{action}_2 \}$

$P_n \{ \text{action}_n \}$

## UNIT 2 AND UNIT 3

Palek 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 :

$$\text{FIRST}(E) = \{ (, id \}$$

$$\text{follow}(E) = \{ \$, +, ) \}$$

$$\text{FIRST}(T) = \{ (, id \}$$

$$\text{follow}(T) = \{ *, \$, +, ) \}$$

$$\text{FIRST}(F) = \{ (, id \}$$

$$\text{follow}(F) = \{ *, \$, +, ) \}$$

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 ( $Q_0, \epsilon$ ) - closure  $\{\epsilon | E^* \rightarrow E^*\}, [E \rightarrow E^* + T]\}$   
 $E^* \rightarrow E^*$       }  $Q_1$   
 $E \rightarrow E^* + T$       }

goto ( $Q_0, T$ ) - closure  $(\{ | E \rightarrow T^*\} \cup \{T \rightarrow T^* * F\})$   
 $E \rightarrow T^*$       }  $Q_1$   
 $T \rightarrow T^* * F$       }  $Q_2$

goto ( $Q_0, E$ )

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

goto ( $Q_0, F$ )

$Q_3 \leftarrow T \rightarrow F$ .  $Q_3$

goto ( $Q_0, id$ )

$F \rightarrow id.$   $Q_5$

goto  $(\mathcal{Q}_1, \star)$

$$\begin{array}{l} E \rightarrow E + T \\ T \rightarrow . T * F \\ T \rightarrow . F \\ F \rightarrow . (E) \\ F \rightarrow . id \end{array} \quad \left. \begin{array}{c} \\ \\ \\ \end{array} \right\} \mathcal{P}_6$$

goto  $(\mathcal{Q}_2, \star)$

$$\begin{array}{l} T \rightarrow T * . F \\ F \rightarrow . (E) \\ F \rightarrow . id \end{array} \quad \left. \begin{array}{c} \\ \\ \end{array} \right\} \mathcal{P}_7$$

goto  $(\mathcal{Q}_4, E)$

$$\begin{array}{l} F \rightarrow (E.) \\ E \rightarrow E . + T \end{array} \quad \left. \begin{array}{c} \\ \end{array} \right\} \mathcal{Q}_8$$

goto  $(\mathcal{Q}_6, T)$

$$\begin{array}{l} E \rightarrow E + T . \\ T \rightarrow T . * F \end{array} \quad \left. \begin{array}{c} \\ \end{array} \right\} \mathcal{P}_9$$

goto  $(\mathcal{Q}_7, F)$

$$T \rightarrow T * F . \quad \left. \begin{array}{c} \\ \end{array} \right\} \mathcal{P}_{10}$$

goto  $(\mathcal{Q}_8, )$

$$F \rightarrow (E).$$

SLR parsing table:

State	Action						go to		
	+	*	(	)	id	\$	E	T	F
0			S4		S <del>5</del>		1	2	3
1	S6					acc			
2	r2	S7		r2		r2			
3	r4	r4		b4		r4			
4			S4		S <del>5</del>		8	2	3
5	r6	r6		r6		r6			
6			S4		S5			9	3
7			S4		S5				10
8	S6			S11					
9	r4	S7		a1		r1			
10	r3	r3		r3		r3			
11		<del>r5</del>	<del>r5</del>	r5	<del>r5</del>	r5			

Q2] CRG :

$$L \rightarrow A Q | A$$

$$Q \rightarrow uq | pq | dip$$

$$A \rightarrow LZ | ZX$$

$$Y \rightarrow 2|3|4|5$$

$$Z \rightarrow 8|9$$

$$X \rightarrow 8|1|Y|6|7|2$$

Eliminating left factoring of production 'L'.

$$L \rightarrow AL'$$

$$L' \rightarrow q | E$$

$$Q \rightarrow uq$$

$$Q \rightarrow pq$$

$$Q \rightarrow dip$$

$$A \rightarrow LZ$$

$$A \rightarrow ZX$$

$$Y \rightarrow 2$$

$$Y \rightarrow 3$$

$$Y \rightarrow 4$$

$$Y \rightarrow 5$$

$$Z \rightarrow 8$$

$$Z \rightarrow 9$$

$$X \rightarrow 0$$

$$X \rightarrow 1$$

$$X \rightarrow Y$$

$$X \rightarrow 6$$

$$X \rightarrow 7$$

$$X \rightarrow Z$$

$$\text{FIRST}(L) = \{1, 2, 3, 4, 5\}$$

$$\text{FIRST}(L') = \{E, uq, pq, dip\}$$

$$\text{FIRST}(Q) = \{uq, pq, dip\}$$

$$\text{FIRST}(A) = \{1, 2, 3, 4, 5\}$$

$$\text{FIRST}(Y) = \{2, 3, 4, 5\}$$

$$\text{FIRST}(Z) = \{8, 9\}$$

$$\text{FIRST}(X) = \{0, 1, 2, \dots, 9\}$$

$$\text{Follow}(L) = \text{Follow}(L') = \text{Follow}(Q)$$

$$\text{Follow}(A) = \text{Follow}(u) = \text{Follow}(Y) = \{\$\}$$

$$\text{Follow}(A) = \{\$, 0, 1, 2, \dots, 9\}$$



### Q3] Grammar

$$A \rightarrow T \mid B$$

$$T \rightarrow pcccs \mid pccb$$

$$B \rightarrow ce \mid ct \mid ee \mid ec$$

i/p : pccsb

Stack	Top buffer	Action
\$	pccsb\$	Shift p
\$P	ccsb\$	shift c
\$pc	csb\$	shift cs
\$pcc	b\$	Shift b
\$pccs	\$	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.

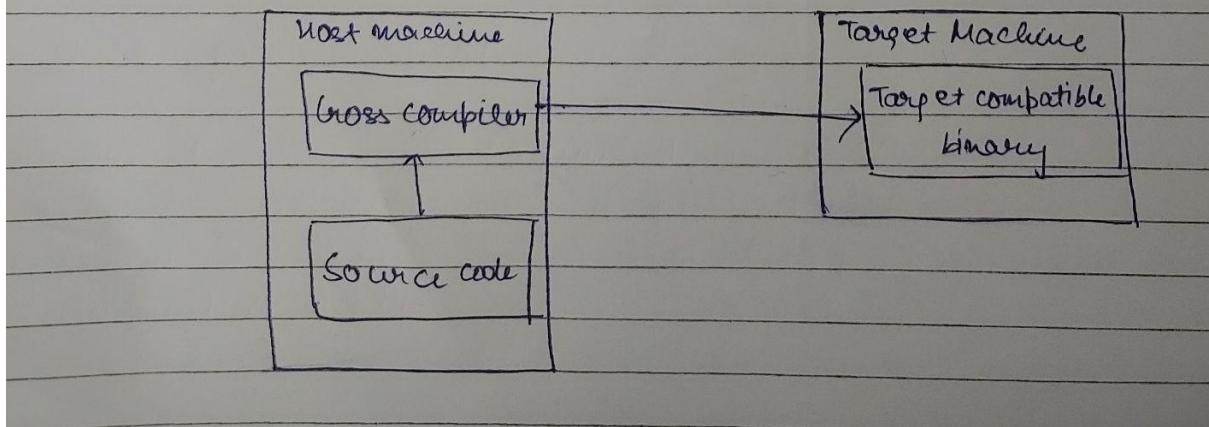
For example, A compiler that runs on windows platform also generates a code that runs on linux platform is called a cross compiler.

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

The cross compiler is also known as "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.
- Bootstrapping 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

Palek Sharma  
RA1911033010112

PAGE NO.:  
DATE: / /

### UNIT - 5

#### Storage allocation techniques

The different ways to allocate memory are:-

1. Static 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 ~~base~~ 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

Profile > Pages > Introduction > Matching Specific String

Matching Specific String ★

Points: 685 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**

Write a regex to find out if a given sentence starts or ends with hackerank

Solve challenge

Submitted Code

Language: Python 3

```
1 Regex_Pattern = r'hackerrank' # Do not delete 'r'.
2
3
```

NEED HELP?

View discussions

View editorial

View top submissions

**Test case 0** Compiler Message

**Test case 1** Success

**Test case 2** Input (stdin) Download

```
1 the hackerrank team is on a mission to flatten the world by
restructuring the ms 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

```
1 Number of matches : 2
```

2)

HackerRank PREPARE CERTIFY COMPETE

Search Points: 665 Rank: 2574

Matching Anything But a Newline ★

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**

Write a regex to find out if a string starts, contains or ends with hackerank  
[Solve challenge](#)

**Submitted Code**

Language: Python 3 [\[+\] Open in editor](#)

```
1 regex_pattern = r"^\....\....\....\....$" # Do not delete 'r'.
```

**Test case 0**

Compiler Message: SUCCESS

Input (edn): [Download](#)

```
1 222-444-abc.def
```

Expected Output: [Download](#)

```
1 True
```

**Test case 1**

**Test case 2**

**Test case 3**

**Test case 4**

**Test case 5**

**Test case 6**

**Test case 7**

3)

HackerRank PREPARE NEW CERTIFY COMPETE

Profile > Beginner > Introduction > Matching Digits & Non-Digit Characters Points: 685 Rank: 2574

Matching Digits & Non-Digit Characters ★

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]

```
1 Regex_Pattern = r"\d\d\DD\d\d\DD\d\d\dd\dd" # Do not delete 'r'.
```

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input (stdin) Download 00-11-2013

Test case 3

Test case 4 Expected Output Download true

Test case 5

4)

HackerRank PREPARE CERTIFY COMPETE

Search pt1744

Points: 665 Rank: 2574

Problems > Regex > Introduction > Matching Whitespace & Non-Whitespace Characters

Matching Whitespace & Non-Whitespace Character ★

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**

Write a regex to find out if a given string starts/ends with whitespace or not with this challenge.

**Submitted Code**

Language: Python 3 [x] Open in editor

```
1 Regex_Pattern = r"\S\S|\s\S\S|\s\S\S" # Do not delete 'r'.
```

**Test case 0** Compiler Message

**Test case 1** Success

**Test case 2** Input (stdin) Download

```
1 12 11 13
```

**Test case 3** Expected Output Download

```
1 true
```

**Test case 4**

**Test case 5**

5)

6)

HackerRank PREPARE CERTIFY COMPETE

Search Points: 685 Rank: 2574

Prep 7 RegEx 7 Introduction Matching Start & End

## Matching Start & End ★

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: Python3

```
1 Regex_Pattern = r"^\d\w\w\w\w\$" # Do not delete 'r'.
2
3
```

[Open in editor]

Test case 0 Compiler Message

Test case 1 Success

Test case 2 Input (edit) Download

Test case 3 Update

Test case 4 Expected Output Download

Test case 5

7)

HackerRank PREPARE CERTIFY COMPETE

Search

Profile Points: 685 Rank: 2574

Progress 7 / 800 Characters Class 1 Matching Specific Characters

## Matching Specific Characters ★

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 challenge:

### Find HackerRank

Write a regex to find out if a string starts and ends with hackerank

Solve Challenge

#### Submitted Code

Language: Python 3

```
1 Regex_Pattern = r"^(123|0123)(xx0|08Ma|xxu)(\., )\$" # Do not delete
2
3
```

Open in editor

#### Test cases

Test case 0	Compiler Message
Success	
Test case 1	Success
Test case 2	input (stdin) 123xx.
	Download
Test case 3	Success
Test case 4	Expected Output 1 trial
	Download
Test case 5	Success
Test case 6	Success
Test case 7	Success
Test case 8	Success
Test case 9	Success

Need Help? View discussions View editorial View top submissions

8)

HackerRank PREPARE CERTIFY COMPETE

Search Points: 695 Rank: 2574

Excluding Specific Characters ★

You made this submission 5 months ago.  
Score: 10.00 Status: Accepted  
People who solved Excluding Specific Characters attempted this next:

**Find HackerRank**

Write a regex to find out if conversations start with both start and end with hackerank

**Submitted Code**

Language: Python 3 [Open in editor]

```
1 Regex_Pattern = r'^\D[aeiou][^bcDF]\S|^AIIOU|^.{-}\$' # Do not
2 delete 'r'.
3
```

**Test case 0** Compiler Message: SUCCESS

**Test case 1** Input (stdin): think? Download: true

**Test case 2** Input (stdin): think Download: false

**Test case 3** Input (stdin): think Download: false

**Test case 4** Input (stdin): think Download: true

**Test case 5**

**Test case 6**

9)

HackerRank PREPARE CERTIFY COMPETE

Profile > Profile > Character Class > Matching Character Ranges Points: 665 Rank: 2574

Matching Character Ranges ★

Problem Submissions Leaderboard | Discussions | Editorial

You made this submission 5 months ago.

Score: 10.00 Status: Accepted

People who solved Matching Character Ranges stamped this next:

**Find HackerRank**

Write a regex to find out if a given string starts with both start and end with hackerank

**Save challenge**

**Submitted Code**

Language: Python 3 [Open in editor]

```
1 Regex_Pattern = r"^[a-z][1-9][a-z][^A-Z][A-Z].*$" # Do not delete 'r'.
```

**Test case 0** Compiler Message Success

Input (stdin) Download

```
1 H4CKR
```

Expected Output Download

```
1 true
```

**Test case 1**

**Test case 2**

**Test case 3**

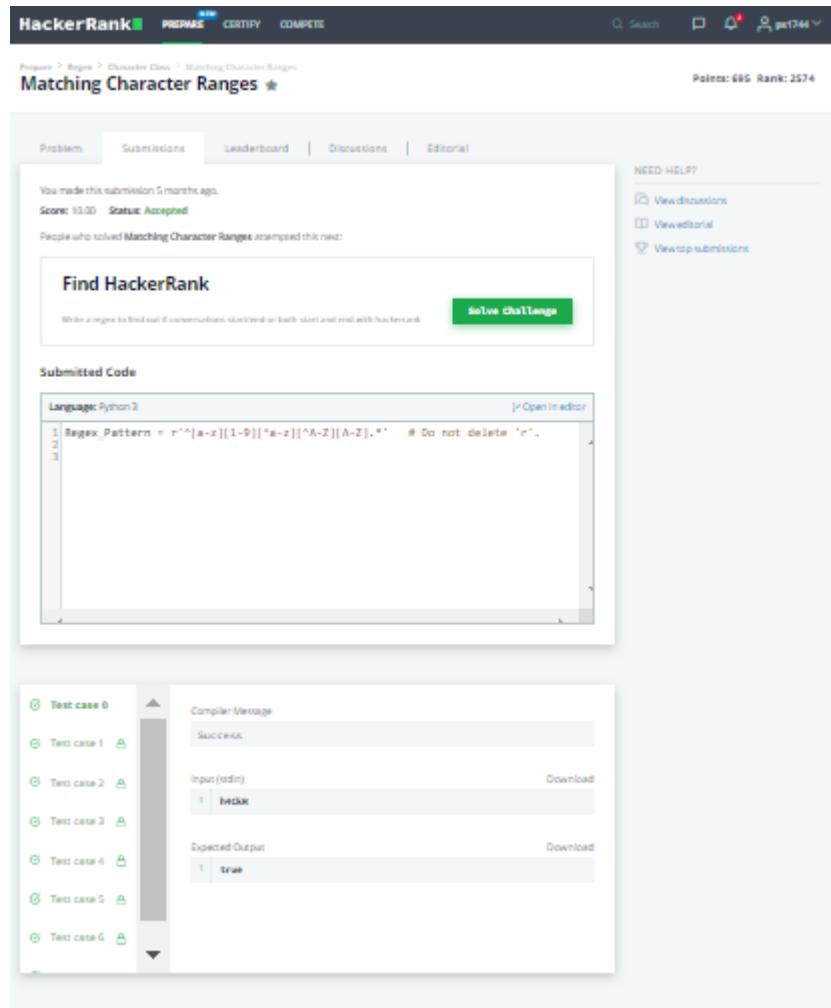
**Test case 4**

**Test case 5**

**Test case 6**

NEED HELP?

View discussions View editorial View top submissions



10)

# 11)

HackerRank PREPARE CERTIFY COMPETE Search px744

Projects > RegEx > Repetition > Matching {x,y} Repetitions Points: 605 Rank: 2574

Matching {x,y} Repetitions ★

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**  
Write a regex to find out if conversations start with both start and end with hackerank  
[Solve challenge](#)

Submitted Code

Language: Python 3 [\[+\] Open in editor](#)

```
1 Regex_Pattern = r'^\d{1,2}[a-zA-Z]{3,}(0,1)$' # Do not delete 'r'  
2  
3
```

NEED HELP?

[View discussions](#)  
[View editorial](#)  
[View top submissions](#)

**Test case 0** Compiler Message: SUCCESS  
Input (stdin) Download  
1 abcdeffedcba  
Expected Output Download  
1 True

**Test case 1**  
**Test case 2**  
**Test case 3**  
**Test case 4**  
**Test case 5**  
**Test case 6**  
**Test case 7**

12)

HackerRank PREPARE CERTIFY COMPETE

Search notifications profile pr0744

Problems 2 Regressions Matching Zero Or More Repetitions

Matching Zero Or More Repetitions ★ Points: 60G 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 attempted this next:

**Find HackerRank**

With a regex to find out if a number is a prime, start and end with hackerank

Solve challenge

Submitted Code

Language: Python 3 [Open in editor]

```
1 Regex_Pattern = r'^(\d{2,})([a-zA-Z]*$)' # Do not delete 'r'.
```

Test case 0 Compiler Message Success

Test case 1 Input (stdin) 14 Download

Test case 2 Input (stdin) 14

Test case 3 Input (stdin) 14

Test case 4 Expected Output 1 TRUE Download

Test case 5

Test case 6

13)

HackerRank PREPARE CERTIFY COMPETE

Search notifications profile

Progress 2 Regress 2 Repertoire Matching One Or More Repetitions

Matching One Or More Repetitions ★ Points: 600 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**

With a regex, find out if your solution starts or ends with hackerank

Solve challenge

Submitted Code

Language: Python 3 [Open in editor]

```
1 Regex_Pattern = r'^([A-Z]+[a-z]+)*$' # Do not delete 'r'.
```

Test case 0 Compiler Message Success

Test case 1 Input (stdin) Download

Test case 2 Input (stdin) 1 2qa

Test case 3 Input (stdin) 1 true

Test case 4 Input (stdin) 1 false

Test case 5 Input (stdin) 1

Test case 6 Input (stdin) 1

14)

HackerRank PREPARE CERTIFY COMPETE Search px1744

Problems RegEx Reports Matching Ending Items Points: 600 Rank: 2574

Matching Ending Items ★

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.  
Score: 20.00 Status: Accepted  
People who solved Matching Ending Items attempted this next:

**Find HackerRank**  
Write a regex to find out if a given string starts or ends with hackerank  
[Solve challenge](#)

Submitted Code

Language: Python 3 [Open in editor](#)

```
1 Regex_Pattern = r"^[a-zA-Z]*$" # Do not delete 'r'.
2
3
```

Test case 0 Compiler Message Success

Test case 1 [A](#)

Test case 2 [A](#) Input (stdin) Download  
1 Kites

Test case 3 [A](#)

Test case 4 [A](#) Expected Output Download  
1 true

Test case 5 [A](#)

Test case 6 [A](#)

15)

Prepared by: Regex | Dropping and Capturing | Matching Word Boundaries

Points: 685 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**

Write a regex to find out if conversations start/end on both start and end with hackerank

**Solve challenge**

**Submitted Code**

Language: Python 3 [\[+\] Open in editor](#)

```
1 Regex_Pattern = r"\b[aeiouAEIOU][a-zA-Z]*\b" # Do not delete 'r'.
```

**Test case 0** Compiler Message

**Test case 1** Success

**Test case 2** Input (stdin) [Download](#)

```
1 Found any match?
```

**Test case 3** Expected Output [Download](#)

```
1 true
```

**Test case 4**

**Test case 5**

**Test case 6**

NEED HELP?  
[View discussions](#)  
[View editorial](#)  
[View top submissions](#)

16)

17)

18)

HackerRank PREPARE CERTIFY COMPETE Search pt1744

Profile > Rego > Hackerrank - Matching Same Text Again & Again  
Matching Same Text Again & Again ★ Points: 605 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 stamped this now:

**Find HackerRank**  
Write a regex to find out if conversations start with both short and real with hackerank  
[Solve challenge](#)

Submitted Code

Language: Python 2 [Open in editor]

```
1 Regex_Pattern = r"^(a-z)\w\w\w\w(d\b|\D|\A-Z)(a-z,A-Z)(aisou|EIOU)\5)\$" #  
2  
3
```

Test case 0 Compiler Message Success

Test case 1 Input (stdin) Download  
ab is available is avail

Test case 2 Input (stdin) Download

Test case 3 Input (stdin) Download

Test case 4 Input (stdin) Download

Test case 5 Input (stdin) Download

Test case 6 Input (stdin) Download

19)

HackerRank PREPARE CERTIFY COMPETE

Projects 2 RegEx 2 Backreferences To Failed Groups

Backreferences To Failed Groups ★ Points: 60G 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 attempted this next:

**Find HackerRank**

With a regex, find out if your submission started on both start and end with hackerank

Solve challenge

Submitted Code

Language: Python 3 [Open in editor]

```
2 Regex_Pattern = r"^(d{2}(?!)d{2})d{2}1d{2}1d{2}$" # Do not delete 'r'.
```

Test case 0 Compiler Message

Test case 1 Success

Test case 2 Input (stdin) Download

```
1 12-24-56-78
```

Test case 3

Test case 4 Expected Output Download

```
1 True
```

Test case 5

20)

HackerRank PREPARE CERTIFY COMPETE

Search user: pef1744

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**

With a regex to find all 8 consecutive, identical or both start and end with backslash.

Solve Challenge

**Submitted Code**

Language: R/R

```
1
2
3 $Regex_Pattern = '^(\d{2}(-\d{2})?|.|.)\d{2}\1\d{2}\1\d{2}$'; //Do not
4 delete '/'. Replace _____ with your regex.
5
6
```

**Test case 0** Compiler Message SUCCESS

**Test case 1** Input (stdin) Download

12-34-56-78

**Test case 2** Expected Output Download

12345678

**Test case 3**

**Test case 4**

**Test case 5**

**Test case 6**

21)

The screenshot shows a HackerRank submission page for the 'Forward References' challenge. At the top, there's a navigation bar with 'HackerRank', 'PREPARE', 'CERTIFY', 'COMPETE', a search bar, and user information ('Points: 665 Rank: 2574'). Below the navigation is the problem title 'Forward References ★'. The main area has tabs for 'Problem', 'Submissions' (which is selected), 'Leaderboard', 'Discussions', and 'Editorial'. Under 'Submissions', it says 'You made this submission 5 months ago.' and 'Score: 20.00 Status: Accepted'. A note below states 'People who solved Forward References attempted this next:'. A 'Find HackerRank' section contains a challenge button. The 'Submitted Code' section shows the following code in R/R:

```
1
2
3
4 $Regex_Pattern = '/^(\?tic|{tac})+$/'; //Do not delete '/'. Replace
5 ----- with your regex.
6
```

The 'Test cases' section lists five test cases:

- Test case 0: Compiler Message (Success)
- Test case 1: Input (stdin) `tactic` (Success)
- Test case 2: Input (stdin) `tacttic` (Success)
- Test case 3: Expected Output `true` (Success)
- Test case 4: Expected Output `false` (Success)

22)

HackerRank PREPARE CERTIFY COMPETE

Search user: pvt744

Positive Lookahead ★ Points: 685 Rank: 2574

Problem Submissions Leaderboard | Discussions | Editorial

You made this submission 5 months ago.

Score: 20.00 Status: Accepted

People who solved Positive Lookahead attempted this next:

**Find HackerRank** Solve Challenge

Submitted Code

Language: PHP Open in editor

```
2
3
4 $regex Pattern = '/a(?=oo)/'; //Do not delete '/'. Replace _____ with
your regex.
5
6
```

Test case 0 Compiler Message

Test case 1 Success

Test case 2 Input (stdin) Download

```
1 gaaaaa
```

Test case 3 Expected Output Download

```
1 number of matches : 3
```

23)

HackerRank PREPARE CERTIFY COMPETE

Search Points: 685 Rank: 2574

Prepare > Begin > Assertions > Negative Lookahead

### Negative Lookahead ★

Score: 20.00 Status: Accepted

People who solved Negative Lookahead: 1096

**Find HackerRank**

Write a regex to find all occurrences. Allowed at least one digit and one word character.

**Solve Challenge**

**Submitted Code**

Language: PHP

```
2
3
4 $Regex_Pattern = '/(\d)(?!\d)/'; //Do not delete '/'. Replace _____
5
6
```

**Test case 0** Compiler Message: Success

**Test case 1** Input (stdin): `goooo` Download

**Test case 2** Input (stdin): `1234567890` Download

**Test case 3** Expected Output: `Number of matches : 2` Download

24)

HackerRank PREPARE CERTIFY COMPETE

Search Points: 695 Rank: 2574

Positive Lookbehind ★

You made this submission 5 months ago.  
Score: 20.00 Status: Accepted

People who solved Positive Lookbehind accepted this next:

**Find HackerRank**

Write a regex to find out if a given string is defined on both start and end with hackerank  
[Solve challenge](#)

**Submitted Code**

Language: PHP [Open in editor]

```
1
2
3
4 $Regex_Pattern = '/(?(?<=[13579])[0-9])/'; //Do not delete '/'. Replace
5 ----- with your regex.
6
```

**Test case 0** Compiler Message

**Test case 1** Success

**Test case 2** Input (stdin) Download

```
1 123456
```

**Test case 3** Expected Output Download

```
1 number of matches : 1
```

25)

HackerRank PREPARE CERTIFY COMPETE

Programs > Regex > Assertions > Negative Lookbehind ★

Points: 685 Rank: 2574

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.  
Score: 20.00 Status: Accepted

People who solved Negative Lookbehind also solved this next:

**Find HackerRank**

Write a regex to find out if conversations start with a double slash and end with backslash.

**Solve challenge**

**Submitted Code**

Language: PHP [\[+\] Open in editor](#)

```
2
3
4 $Regex_Pattern = "/(?!([aeiouAEIOU])|\n\$)/"; //Do not delete '/'. Replace
5     _____ with your regex.
6
```

**Test case 0** Compiler Message

**Test case 1** Success

**Test case 2** Input (stdin) Download

```
1 1111
```

**Test case 3** Expected Output Download

```
1 number of matches : 2
```

26)

27)

HackerRank PREPARE CERTIFY COMPETE

Search pt1744

Problems > Beginner > Applications > Detect HTML links

Detect HTML links ★

Points: 685 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

NEED HELP?

View discussions View top submissions

Submitted Code

Language: Python 3

[> Open in editor]

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re
3 n = int(input())
4 l = []
5 for i in range(n):
6     x = str(input())
7     l.append(x)
8 pattern = re.compile('ca href="(.*?)"\s+?(.*?)</a>')
9 for x in l:
10     a = re.findall(pattern, x)
11     for z in a:
12         y1 = z[0].strip()
13         y2 = z[1].strip()
14         p2 = re.compile('[<.*>]*(.*)<.*>')
15
16
17
```

Test case 0

Compiler Message

Success

Input (stdin)

Download

```
1 2
2 <p><a href="http://www.quickit.com/html/tutorial/html_links.cfm">
3 sample link</a></p>
4 <br class="more-info"><a href="http://www.quickit.com/html/example/html_links_exampl
5 et.cfm">More Link Examples...</a></div>
```

Expected Output

Download

```
1 http://www.quickit.com/html/tutorial/html_links.cfm,example
```

28)

HackerRank PREPARE CERTIFY COMPETE

Search Notifications User: pr1244 Points: 685 Rank: 2574

Problems 200 200 Applications 1 Find A Sub-Word ★

Find A Sub-Word ★

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**

Write a regex to find out if a word or sentence starts or ends with hackerank

**Solve challenge**

**Submitted Code**

Language: Python 3 [Open in editor]

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2
3 import re
4 n = int(input())
5 x = []
6 for i in range(n):
7     y = str(input())
8     x.append(y)
9 t = int(input())
10 for i in range(t):
11     y = str(input())
12     pattern = re.compile('(^|\\w)' + y + '(\\w|$)')
13     count = 0
14     for x in x:
15         if pattern.search(x):
16             count += 1
17
18 print(count)
```

**Test case 0** Compiler Message Success

**Test case 1** Input (stdin) Download

```
1
2 awaiting pessimist optimist this is
```

**Test case 2** Input (stdin) Download

```
1
2
3
4
```

**Test case 3** Input (stdin) Download

```
1
2
3
4
```

**Test case 4** Input (stdin) Download

```
1
2
3
4
```

**Test case 5** Expected Output Download

```
1
```

**Test case 6** Input (stdin) Download

```
1
```

29)

30)

Prep | Regress | Applications | IP Address Validation

Points: 685 Rank: 2574

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.

Score: 10.00 Status: Accepted

People who solved IP Address Validation attempted this next:

## Find HackerRank

Write a regex to find out if a given string is a valid IPv4 address and print with hasRank.

Solve Challenge

### Submitted Code

Language: Python 3

```
# 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])|(0[0-9]))((1[0-9][0-9])|(0[0-9]))((1[0-9][0-9])|(0[0-9]))((\\d))\\."
((2[0-5][0-5])|(1[0-9][0-9])|(0[0-9]))((1[0-9][0-9])|(0[0-9]))((1[0-9][0-9])|(0[0-9]))((\\d))\\."
p2 = re.compile("((a-[0-9])|(a-[0-9]))((1,4)|(7))((a-[0-2])|(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:
        print("Neither")
```

### Test case 0

Compiler Message

### Test case 1

Success

### Test case 2

input (stdin)

```
T
```

Download

### Test case 3

input (stdin)

```
18800.18800.18800.18800.2286
18800.18800.18800.18800.2286
18800.18800.18800.18800.2286
1881.18800.18800.18800.2286
22.233.111.64
22.233.111.164
222.233.111.64
```

31)

32)

HackerRank PREPARE CERTIFY COMPETE

Search Notifications User: pmt244

Prepare Begin Applications Detect the Email Addresses

Points: 695 Rank: 2574

Problem Submissions Leaderboard | Discussions

You made this submission 5 months ago.

Score: 100 Status: Accepted

People who solved Detect the Email Addresses attempted this next:

**Find HackerRank**

With a regular expression, find all email addresses mentioned in both start and end with hackerank.

**Solve Challenge**

**Submitted Code**

Language: Python 3

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re
3 p = re.compile('([0-9a-zA-Z]+[.][0-9a-zA-Z]*@[0-9a-zA-Z]+[.][0-9a-zA-Z]+[0-9a-zA-Z.]+)*')
4 n = int(input())
5 l = []
6 for i in range(n):
7     x = str(input())
8     r = re.findall(p, x)
9     for z in r:
10         if(z not in l):
11             l.append(z)
12 l.sort()
13 for i in range(len(l)):
14     print(l[i])
```

**Test case 0** Compiler Message Success

**Test case 1**

**Test case 2** Input (stdin) Download

```
1 20
```

**Test case 3** Finally this phase is testimony to our quest and ever open ears for hearing from our customers since 1992. We look forward to hearing from you today.

**Test case 4** All India National Toll Free Number: 1800 8425 8425

Working Hours: 20:00 am to 09:00 pm (Monday - Friday).

18:00 am to 21:00 pm (Saturday). To report ATM Card Lost, kindly contact: +91 (44) 2822 3288 / 2822 3189.

The Customer Care: +91 8842 481 981

For all your queries, an any of our services in any branch in

33)

HackerRank PREPARE CERTIFY COMPETE

Search Points: 695 Rank: 2574

Detect the Domain Name ★

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**

Write a regex to find out if a given URL starts with http:// or https://

**Solve challenge**

Submitted Code

Language: Python 3 [Open in editor]

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
1 pattern = re.compile('^(http|https)://((www.|[a-zA-Z-]+)|([a-zA-Z-]+\.[a-zA-Z-]+\.[a-zA-Z-]+))(/[^$]*)$'
2
3 for i in range(int(input())):
4     string = input()
5     iterator = regex.finditer(string)
6     if iterator:
7         for match in iterator:
8             s.add(match.group(1)+match.group(4))
9
10 print(';'.join(t for t in sorted(s)))
```

Test case 0 Compiler Message Success

Test case 1

Test case 2

Test case 3

Test case 4

Input(stdin)

Download

```
1 1027
2 <?xml version="1.0" encoding="UTF-8"?>
3 <html>
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6     <meta name="format-detection" content="telephone-nl" />
7   </head>
8   <body>
9     <div> cricketandhindi.com - India, Business, Stock, Sports, Cricket, Entertainment, Bollywood, Music, Video and Breaking news, </div>
10    <div> cricketandhindi.com - India, Business, Stock, Sports, Cricket, Entertainment, Bollywood, Music, Video and Breaking news, </div>
11  </body>
12</html>
```

34)

HackerRank PREPARE CERTIFY COMPETE Search Notifications Points: 685 Rank: 2574

Profile > Beginner > Applications > Building a Smart IDE: Identifying comments ★

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  
People who solved Building a Smart IDE: Identifying comments accepted this next:

**Find HackerRank**

Write a regex to find out if conversations start/lead or both start and end with hackerank

Solve challenge

Submitted Code

Language: Python 3

[> Open in editor]

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re, sys
3 l = []
4 x = sys.stdin.read()
5 x = str(x)
6 for i in range(len(x)):
7     if(x[i] == '\n'):
8         x = x[:i] + '$' + x[i+1:]
9         while(i < len(x) and x[i+1] == '\n'):
10            i += 1
11            x = x[:i] + x[i+1:]
12 x += '$'
13 p1 = re.compile("//(*$|${})|(/\\/*|\\d\\D)*?/*/")
14 l = []
15 ...
```

NEED HELP?

View discussions View top submissions

Test case 0 Compiler Message Success

Test case 1 Success

Test case 2 Input (stdin)

```
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
        }
    }
}
```

Download

35)

HackerRank PREPARE CERTIFY COMPETE

Search

Profile: pr1744 Points: 685 Rank: 2574

Project: Beginner Applications - Detecting Valid Latitude and Longitude Pairs

Detecting Valid Latitude and Longitude Pairs ★

Score: 20.00 Status: Accepted

People who solved Detecting Valid Latitude and Longitude Pairs stamped this next:

**Find HackerRank**

Write a regular expression that matches latitude and longitude pairs in the following format: (+|-)°d°(.,.)°d° (+|-)°d°(.,.)°d°

**Solve challenge**

**Submitted Code**

Language: Python 3

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re
3 p = re.compile('([+-]?\d*(\.\d*)?\d*), ([+-]?\d*(\.\d*)?\d*)')
4 t = int(input())
5 for i in range(t):
6     try:
7         x = str(input())
8         z = x[1:len(x)-1]
9         zx = z.split(',')
10        xx = zx[0]
11        x = xx[0]
12        y = xx[1]
13        if(x[0] == '+' or x[0] == '-'):
14            x = x[1:]
15    except:
16        print("Error")
```

**Test case 0** Compiler Message Success

**Test case 1**

**Test case 2** Input (stdin)

```
1 12
```

**Test case 3**

```
1 (75, 180)
2 (+90.0, -147.45)
```

**Test case 4** Error

```
1 (77.1111111111111, 149.99999999999999)
2 (+90, +180)
```

**Test case 5**

```
1 (90, 180)
2 (-90.000000, -180.000000)
3 (75, 280)
4 (+180.0, -147.45)
```

36)

The screenshot shows the HackerRank interface for the 'HackerRank Tweets' challenge. At the top, there are tabs for 'Problem', 'Submissions', 'Leaderboard', and 'Discussions'. The 'Problem' tab is selected. Below the tabs, it says 'You made this submission 5 months ago.' and 'Score: 15.00 Status: Accepted'. A note indicates that people who solved this challenge have stamped their profile with the 'HackerRank' badge.

**Find HackerRank**

Write a regex to find out if conversations start with both start and end with hackerank

**Solve challenge**

**Submitted Code**

Language: Python 3 [Open in editor]

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re
3 p = re.compile('((h|H)(a|A)(c|C)(k|K)(e|E)(r|R)(s|S)(n|N)(k|K))')
4 t = int(input())
5 c = 0
6 for i in range(t):
7     x = str(input())
8     c += len(re.findall(p, x))
9 print(c)
10
```

**Test case 0**

Compiler Message: SUCCESS

Input (stdin) Download

```
1 4
2 1 love_hackerrank
3 1 just scored 27 points in the Picking Cards challenge on
4 1 just signed up for summer cup @hackerrank
5 1 interesting talk by harri, co-founder of hackerrank
```

**Test case 1**

**Test case 2**

**Test case 3**

**Test case 4**

**Test case 5**

**Test case 6**

**Test case 7**

**Test case 8**

**Test case 9**

**Test case 10**

**Test case 11**

**Test case 12**

**Test case 13**

**Test case 14**

**Test case 15**

**Test case 16**

**Test case 17**

**Test case 18**

**Test case 19**

**Test case 20**

**Test case 21**

**Test case 22**

**Test case 23**

**Test case 24**

**Test case 25**

**Test case 26**

**Test case 27**

**Test case 28**

**Test case 29**

**Test case 30**

**Test case 31**

**Test case 32**

**Test case 33**

**Test case 34**

**Test case 35**

**Test case 36**

**Test case 37**

**Test case 38**

**Test case 39**

**Test case 40**

**Test case 41**

**Test case 42**

**Test case 43**

**Test case 44**

**Test case 45**

**Test case 46**

**Test case 47**

**Test case 48**

**Test case 49**

**Test case 50**

**Test case 51**

**Test case 52**

**Test case 53**

**Test case 54**

**Test case 55**

**Test case 56**

**Test case 57**

**Test case 58**

**Test case 59**

**Test case 60**

**Test case 61**

**Test case 62**

**Test case 63**

**Test case 64**

**Test case 65**

**Test case 66**

**Test case 67**

**Test case 68**

**Test case 69**

**Test case 70**

**Test case 71**

**Test case 72**

**Test case 73**

**Test case 74**

**Test case 75**

**Test case 76**

**Test case 77**

**Test case 78**

**Test case 79**

**Test case 80**

**Test case 81**

**Test case 82**

**Test case 83**

**Test case 84**

**Test case 85**

**Test case 86**

**Test case 87**

**Test case 88**

**Test case 89**

**Test case 90**

**Test case 91**

**Test case 92**

**Test case 93**

**Test case 94**

**Test case 95**

**Test case 96**

**Test case 97**

**Test case 98**

**Test case 99**

**Test case 100**

**Test case 101**

**Test case 102**

**Test case 103**

**Test case 104**

**Test case 105**

**Test case 106**

**Test case 107**

**Test case 108**

**Test case 109**

**Test case 110**

**Test case 111**

**Test case 112**

**Test case 113**

**Test case 114**

**Test case 115**

**Test case 116**

**Test case 117**

**Test case 118**

**Test case 119**

**Test case 120**

**Test case 121**

**Test case 122**

**Test case 123**

**Test case 124**

**Test case 125**

**Test case 126**

**Test case 127**

**Test case 128**

**Test case 129**

**Test case 130**

**Test case 131**

**Test case 132**

**Test case 133**

**Test case 134**

**Test case 135**

**Test case 136**

**Test case 137**

**Test case 138**

**Test case 139**

**Test case 140**

**Test case 141**

**Test case 142**

**Test case 143**

**Test case 144**

**Test case 145**

**Test case 146**

**Test case 147**

**Test case 148**

**Test case 149**

**Test case 150**

**Test case 151**

**Test case 152**

**Test case 153**

**Test case 154**

**Test case 155**

**Test case 156**

**Test case 157**

**Test case 158**

**Test case 159**

**Test case 160**

**Test case 161**

**Test case 162**

**Test case 163**

**Test case 164**

**Test case 165**

**Test case 166**

**Test case 167**

**Test case 168**

**Test case 169**

**Test case 170**

**Test case 171**

**Test case 172**

**Test case 173**

**Test case 174**

**Test case 175**

**Test case 176**

**Test case 177**

**Test case 178**

**Test case 179**

**Test case 180**

**Test case 181**

**Test case 182**

**Test case 183**

**Test case 184**

**Test case 185**

**Test case 186**

**Test case 187**

**Test case 188**

**Test case 189**

**Test case 190**

**Test case 191**

**Test case 192**

**Test case 193**

**Test case 194**

**Test case 195**

**Test case 196**

**Test case 197**

**Test case 198**

**Test case 199**

**Test case 200**

**Test case 201**

**Test case 202**

**Test case 203**

**Test case 204**

**Test case 205**

**Test case 206**

**Test case 207**

**Test case 208**

**Test case 209**

**Test case 210**

**Test case 211**

**Test case 212**

**Test case 213**

**Test case 214**

**Test case 215**

**Test case 216**

**Test case 217**

**Test case 218**

**Test case 219**

**Test case 220**

**Test case 221**

**Test case 222**

**Test case 223**

**Test case 224**

**Test case 225**

**Test case 226**

**Test case 227**

**Test case 228**

**Test case 229**

**Test case 230**

**Test case 231**

**Test case 232**

**Test case 233**

**Test case 234**

**Test case 235**

**Test case 236**

**Test case 237**

**Test case 238**

**Test case 239**

**Test case 240**

**Test case 241**

**Test case 242**

**Test case 243**

**Test case 244**

**Test case 245**

**Test case 246**

**Test case 247**

**Test case 248**

**Test case 249**

**Test case 250**

**Test case 251**

**Test case 252**

**Test case 253**

**Test case 254**

**Test case 255**

**Test case 256**

**Test case 257**

**Test case 258**

**Test case 259**

**Test case 260**

**Test case 261**

**Test case 262**

**Test case 263**

**Test case 264**

**Test case 265**

**Test case 266**

**Test case 267**

**Test case 268**

**Test case 269**

**Test case 270**

**Test case 271**

**Test case 272**

**Test case 273**

**Test case 274**

**Test case 275**

**Test case 276**

**Test case 277**

**Test case 278**

**Test case 279**

**Test case 280**

**Test case 281**

**Test case 282**

**Test case 283**

**Test case 284**

**Test case 285**

**Test case 286**

**Test case 287**

**Test case 288**

**Test case 289**

**Test case 290**

**Test case 291**

**Test case 292**

**Test case 293**

**Test case 294**

**Test case 295**

**Test case 296**

**Test case 297**

**Test case 298**

**Test case 299**

**Test case 300**

**Test case 301**

**Test case 302**

**Test case 303**

**Test case 304**

**Test case 305**

**Test case 306**

**Test case 307**

**Test case 308**

**Test case 309**

**Test case 310**

**Test case 311**

**Test case 312**

**Test case 313**

**Test case 314**

**Test case 315**

**Test case 316**

**Test case 317**

**Test case 318**

**Test case 319**

**Test case 320**

**Test case 321**

**Test case 322**

**Test case 323**

**Test case 324**

**Test case 325**

**Test case 326**

**Test case 327**

**Test case 328**

**Test case 329**

**Test case 330**

**Test case 331**

**Test case 332**

**Test case 333**

**Test case 334**

**Test case 335**

**Test case 336**

**Test case 337**

**Test case 338**

**Test case 339**

**Test case 340**

**Test case 341**

**Test case 342**

**Test case 343**

**Test case 344**

**Test case 345**

**Test case 346**

**Test case 347**

**Test case 348**

**Test case 349**

**Test case 350**

**Test case 351**

**Test case 352**

**Test case 353**

**Test case 354**

**Test case 355**

**Test case 356**

**Test case 357**

**Test case 358**

**Test case 359**

**Test case 360**

**Test case 361**

**Test case 362**

**Test case 363**

**Test case 364**

**Test case 365**

**Test case 366**

**Test case 367**

**Test case 368**

**Test case 369**

**Test case 370**

**Test case 371**

**Test case 372**

**Test case 373**

**Test case 374**

**Test case 375**

**Test case 376**

**Test case 377**

**Test case 378**

**Test case 379**

**Test case 380**

**Test case 381**

**Test case 382**

**Test case 383**

**Test case 384**

**Test case 385**

**Test case 386**

**Test case 387**

**Test case 388**

**Test case 389**

**Test case 390**

**Test case 391**

**Test case 392**

**Test case 393**

**Test case 394**

**Test case 395**

**Test case 396**

**Test case 397**

**Test case 398**

**Test case 399**

**Test case 400**

**Test case 401**

**Test case 402**

**Test case 403**

**Test case 404**

**Test case 405**

**Test case 406**

**Test case 407**

**Test case 408**

**Test case 409**

**Test case 410**

**Test case 411**

**Test case 412**

**Test case 413**

**Test case 414**

**Test case 415**

**Test case 416**

**Test case 417**

**Test case 418**

**Test case 419**

**Test case 420**

**Test case 421**

**Test case 422**

**Test case 423**

**Test case 424**

**Test case 425**

**Test case 426**

**Test case 427**

**Test case 428**

**Test case 429**

**Test case 430**

**Test case 431**

**Test case 432**

**Test case 433**

**Test case 434**

**Test case 435**

**Test case 436**

**Test case 437**

**Test case 438**

**Test case 439**

**Test case 440**

**Test case 441**

**Test case 442**

**Test case 443**

**Test case 444**

**Test case 445**

**Test case 446**

**Test case 447**

**Test case 448**

**Test case 449**

**Test case 450**

**Test case 451**

**Test case 452**

**Test case 453**

**Test case 454**

**Test case 455**

**Test case 456**

**Test case 457**

**Test case 458**

**Test case 459**

**Test case 460**

**Test case 461**

**Test case 462**

**Test case 463**

**Test case 464**

**Test case 465**

**Test case 466**

**Test case 467**

**Test case 468**

**Test case 469**

**Test case 470**

**Test case 471**

**Test case 472**

**Test case 473**

**Test case 474**

**Test case 475**

**Test case 476**

**Test case 477**

**Test case 478**

**Test case 479**

**Test case 480**

**Test case 481**

**Test case 482**

**Test case 483**

**Test case 484**

**Test case 485**

**Test case 486**

**Test case 487**

**Test case 488**

**Test case 489**

**Test case 490**

**Test case 491**

**Test case 492**

**Test case 493**

**Test case 494**

**Test case 495**

**Test case 496**

**Test case 497**

**Test case 498**

**Test case 499**

**Test case 500**

**Test case 501**

**Test case 502**

**Test case 503**

**Test case 504**

**Test case 505**

**Test case 506**

**Test case 507**

**Test case 508**

**Test case 509**

**Test case 510**

**Test case 511**

**Test case 512**

**Test case 513**

**Test case 514**

**Test case 515**

**Test case 516**

**Test case 517**

**Test case 518**

**Test case 519**

**Test case 520**

**Test case 521**

**Test case 522**

**Test case 523**

**Test case 524**

**Test case 525**

**Test case 526**

**Test case 527**

**Test case 528**

**Test case 529**

**Test case 530**

**Test case 531**

**Test case 532**

**Test case 533**

**Test case 534**

**Test case 535**

**Test case 536**

**Test case 537**

**Test case 538**

**Test case 539**

**Test case 540**

**Test case 541**

**Test case 542**

**Test case 543**

**Test case 544**

**Test case 545**

**Test case 546**

**Test case 547**

**Test case 548**

**Test case 549**

**Test case 550**

**Test case 551**

**Test case 552**

**Test case 553**

**Test case 554**

**Test case 555**

**Test case 556**

**Test case 557**

**Test case 558**

**Test case 559**

**Test case 560**

**Test case 561**

**Test case 562**

**Test case 563**

**Test case 564**

**Test case 565**

**Test case 566**

**Test case 567**

**Test case 568**

**Test case 569**

**Test case 570**

**Test case 571**

**Test case 572**

**Test case 573**

**Test case 574**

**Test case 575**

**Test case 576**

**Test case 577**

**Test case 578**

**Test case 579**

**Test case 580**

**Test case 581**

**Test case 582**

**Test case 583**

**Test case 584**

**Test case 585**

**Test case 586**

**Test case 587**

**Test case 588**

**Test case 589**

**Test case 590**

**Test case 591**

**Test case 592**

**Test case 593**

**Test case 594**

**Test case 595**

**Test case 596**

**Test case 597**

**Test case 598**

**Test case 599**

**Test case 600**

**Test case 601**

**Test case 602**

**Test case 603**

**Test case 604**

**Test case 605**

**Test case 606**

**Test case 607**

**Test case 608**

**Test case 609**

**Test case 610**

**Test case 611**

**Test case 612**

**Test case 613**

**Test case 614**

**Test case 615**

**Test case 616**

**Test case 617**

**Test case 618**

**Test case 619**

**Test case 620**

**Test case 621**

**Test case 622**

**Test case 623**

**Test case 624**

**Test case 625**

**Test case 626**

**Test case 627**

**Test case 628**

**Test case 629**

**Test case 630**

**Test case 631**

**Test case 632**

**Test case 633**

**Test case 634**

**Test case 635**

**Test case 636**

**Test case 637**

**Test case 638**

**Test case 639**

**Test case 640**

**Test case 641**

**Test case 642**

**Test case 643**

**Test case 644**

**Test case 645**

**Test case 646**

**Test case 647**

**Test case 648**

**Test case 649**

**Test case 650**

**Test case 651**

**Test case 652**

**Test case 653**

**Test case 654**

**Test case 655**

**Test case 656**

**Test case 657**

**Test case 658**

**Test case 659**

**Test case 660**

**Test case 661**

**Test case 662**

**Test case 663**

**Test case 664**

**Test case 665**

**Test case 666**

**Test case 667**

**Test case 668**

**Test case 669**

**Test case 670**

**Test case 671**

**Test case 672**

**Test case 673**

**Test case 674**

**Test case 675**

**Test case 676**

**Test case 677**

**Test case 678**

**Test case 679**

**Test case 680**

**Test case 681**

**Test case 682**

**Test case 683**

**Test case 684**

**Test case 685**

**Test case 686**

**Test case 687**

**Test case 688**

**Test case 689**

**Test case 690**

**Test case 691**

**Test case 692**

**Test case 693**

**Test case 694**

**Test case 695**

**Test case 696**

**Test case 697**

**Test case 698**

**Test case 699**

**Test case 700**

**Test case 701**

**Test case 702**

**Test case 703**

**Test case 704**

**Test case 705**

**Test case 706**

**Test case 707**

**Test case 708**

**Test case 709**

**Test case 710**

**Test case 711**

**Test case 712**

**Test case 713**

**Test case 714**

**Test case 715**

**Test case 716**

**Test case 717**

**Test case 718**

**Test case 719**

**Test case 720**

**Test case 721**

**Test**

37)

HackerRank PREPARE CERTIFY COMPETE

Profile > Beginner > Applications > Build a Stack Exchange Scraper ★ Points: 685 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**

Write a regex to find out if a given sentence, starting or both start and end with hackerank  
[Solve challenge](#)

Submitted Code

Language: Python 3 [Open in editor]

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re, sys
3 p1 = re.compile('<a href="/questions/[0-9]+>.*</a>')
4 p2 = re.compile('<a href="/questions/[0-9]+.*<(.*)></a>'')
5 p3 = re.compile('.<class>"relativetime">(.*)</span>')
6 a = sys.stdin.read()
7 x = re.findall(p1, a)
8 y = re.findall(p2, a)
9 z = re.findall(p3, a)
10 for i in range(len(x)):
11     print(x[i].strip()+' '+y[i].strip()+' '+z[i].strip())
12
```

Test case 0 Compiler Message Success

Test case 1

Test case 2

Test case 3

Test case 4

**Hidden Test Case**  
Unlock this test case for 10 points.

unlock

38)

HackerRank PREPARE CERTIFY COMPETE

Search Points: 695 Rank: 2574

Utopian Identification Number ★

You made this submission 5 months ago.

Score: 15.00 Status: Accepted

People who solved Utopian Identification Number attempted this next:

**Find HackerRank**

Write a regex to find out if a given string starts with both start and ends with both end with hackerank

**Solve challenge**

**Submitted Code**

Language: Python 3 [Open in editor]

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re
3 p = re.compile("[a-z]{0,3}[0-9]{2,8}{A-Z]{3,}")
4 n = int(input())
5 for i in range(n):
6     x = str(input())
7     if(len(re.findall(p, x)) > 0):
8         print("VALID")
9     else:
10        print("INVALID")
11
```

**Test case 0** Compiler Message SUCCESS

Input (stdin) Download

```
1 2
2 abc123321ABCDEF
3 0123AB
```

Expected Output Download

```
1 VALID
2 INVALID
```

39)

HackerRank  PREPARE CERTIFY COMPETE

Search    pt744

Problems > RegEx > Applications > Valid PAN format

## Valid PAN format ★

Points: 605 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

We'll let you know if your solution is submitted or if it fails to pass a test case.

[Solve challenge](#)

Submitted Code

Language: Python 3 [Open in editor]

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re
3 p = re.compile('[A-Z]{5}[0-9]{4}[A-Z]{4}')
4 n = int(input())
5 for i in range(n):
6     x = str(input())
7     if(len(x) != 10):
8         print("NO")
9     elif(len(re.findall(p, x)) > 0):
10        print("YES")
11    else:
12        print("NO")
```

**Test case 0** 

Compiler Message  
Success

Input (stdin)

```
1 3
2 ABCDE12345Y
3 ABCDE12345Y
4 AwEDG12345Y
```

Download

**Test case 1** 

Expected Output

```
1 YES
2 NO
```

Download

**Test case 2** 

**Test case 3** 

**Test case 4** 

**Test case 5** 

**Test case 6** 

**Test case 7** 

40)

HackerRank PREPARE CERTIFY COMPETE Search Notifications User: pet744 Points: 685 Rank: 2574

Saying Hi ★

You made this submission 5 months ago.  
Score: 15.00 Status: Accepted  
People who solved Saying Hi attempted this race.

**Find HackerRank**

Write a regex to find out if conversations start with or both start and end with hackerank

**Solve challenge**

**Submitted Code**

Language: Python 3 [Open in editor]

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 n = int(input())
3 import re
4 p = re.compile("^(H|h|[Ii])\s*d0$")
5 for i in range(n):
6     x = str(input())
7     if(len(re.findall(p, x)) > 0):
8         print(x)
9 
```

**Compiler Message**

Success

Input (stdin) Download

1
2 hi Alex how are you doing
3 hi does how are you doing
4 used by alex
5 hidden agenda
6 Alex greeted Martha by saying xi Martha

Expected Output Download

41)

HackerRank PROPOSE CERTIFY COMPETE

Progress > Beginner > Applications > HackerRank Language

HackerRank Language ★ Points: 685 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**

Write a program to find out if a given string starts or ends with hackerRank

**Submit challenge**

**Submitted Code**

Language: Python 3 [\[+\] Open in editor](#)

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re
3 p = re.compile('(^|\\d\\w*)+'
4                 '(C|CPP|JAVA|PYTHON|PERL|PHP|RUBY|CSHARP|MISCELLY|CLJS|DASH|SCALA|ERLANG|'
5                 'CLISP|LISP|BRAINFUCK|JAVASCRIPT|GO|OCAML|R|PASCAL|SDCL|DART|GROOVY|OBJECT|'
6                 'EWIC)\\$')
7 n = int(input())
8 for i in range(n):
9     x = str(input())
10    if(len(re.findall(p, x)) > 0):
11        print("VALID")
12    else:
13        print("INVALID")
```

**Test case 0**

Compiler Message: Success

Input (stdin):

```
1 X
2 11811 C
3 11812 CPP
4 11844 X
```

Download

**Test case 1**

**Test case 2**

**Test case 3**

**Test case 4**

**Test case 5**

**Test case 6**

**Test case 7**

**Test case 8**

**Test case 9**

Expected Output:

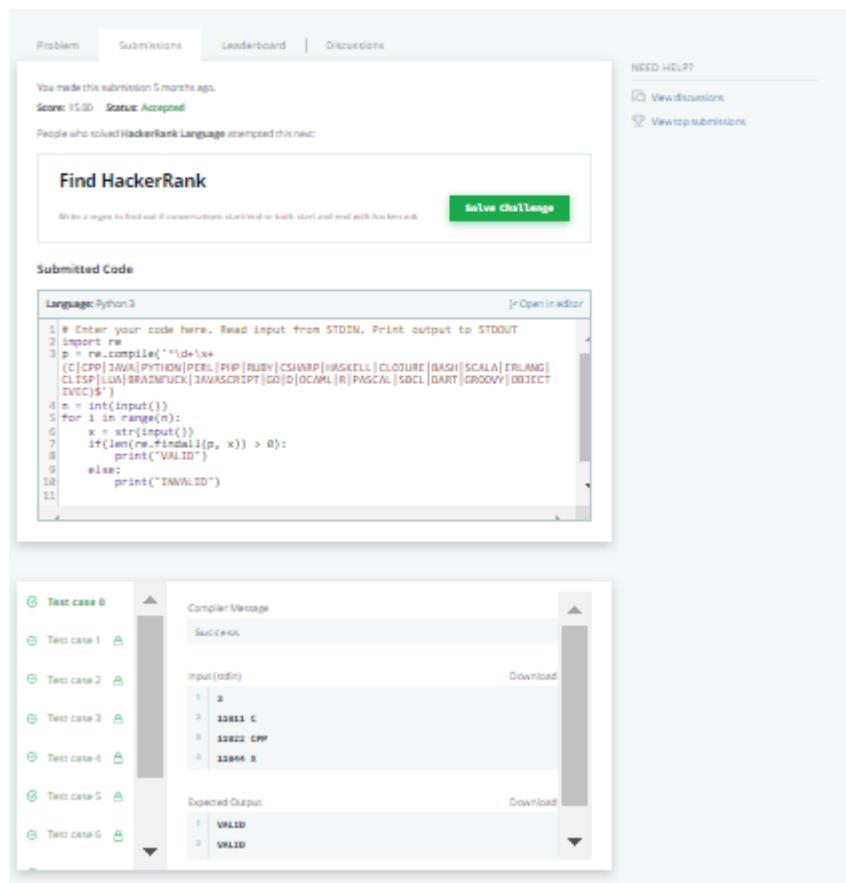
```
1 VALID
2 VALID
```

Download

NEED HELP?

[View discussions](#)

[View top submissions](#)



42)

HackerRank  PREPARE CERTIFY COMPETE

Search     pt1744

Points: 685 Rank: 2574

Building a Smart IDE: Programming Language Detection ★

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** 

Write a regex to find out if command-line arguments had been used with this challenge.

**Submitted Code**

Language: Python 3 

```
1 # Enter your code here. Read input from STDIN. Print output to
2 # STDOUT
3 import sys
4 x = sys.argv
5 x = str(x)
6 if('String args[]' in x or 'java.' in x or 'IOException' in x):
7     print("Java")
8 elif('c.includ' in x or 'return 0' in x or 'void main(' in x or 'int
9     main(' in x):
10     print("C")
11 else:
12     print("Python")
```

**Test case 0** Compiler Message: Success

Input(stdin):

```
1 # let us create a test string
2
3
4 testString1 = "Hello World!"
5 print("Original String: " + testString1)
6
7 # Print this string in lower case
8
9
10 # Converting a string to lower case
11
```

43)

Problems > Regex > Applications > Split the Phone Numbers

**Split the Phone Numbers** ★

Points: 605 Rank: 2574

Problem Submissions Leaderboard Discussions

You made this submission 5 months ago.  
Score: 15.00 Status: Accepted  
People who solved Split the Phone Numbers attempted this next:

**Find HackerRank**

Write a regex to find out if conversations, start with our hash tag and end with hackerank.

**Submitted Code**

Language: Python 3

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
p = re.compile('(\d{1,3})[-|\x](\d{1,3})[-|\x](\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])
```

NEED HELP?

[View discussions](#)

[View top submissions](#)

**Test case 0** Compiler Message: SUCCESS

**Test case 1**

**Test case 2**

**Test case 3**

**Test case 4**

**Test case 5**

**Test case 6**

**Test case 7**

**Hidden Test Case**

Unlock this test case for 8 points.

unlock

44)

HackerRank PREPARE CERTIFY COMPETE

Progress: 20% Applications: 100% HTML/PYTHON Points: 665 Rank: 2574

Detect HTML Attributes ★

You made this submission 5 months ago.

Score: 20.00 Status: Accepted

People who solved Detect HTML Attributes attempted this next:

**Find HackerRank**

Write a regex to find out if a given URL starts with both short and real with hackerank

**Submit challenge**

**Submitted Code**

Language: Python 3 [Open in editor]

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re
3 p = re.compile('^(a-zA-Z0-9)+\w*[^>]*>')
4 n = int(input())
5 tags = []
6 attr = dict()
7 for i in range(n):
8     x = str(input())
9     y = re.findall(p, x)
10    if(len(y) > 0):
11        for z in y:
12            p1 = re.compile('^(a-zA-Z0-9)+\w*[^>]*>')
13            p2 = re.compile('^(S+)=["\'][^"\']*"["\'][^"\']*"')
14            c = re.findall(p1, z)
15
16
17
```

**Test case 0**

Compiler Message

Success

Input (stdin)

```
1
2
```

Download

**Test case 1**

Compiler Message

Success

Input (stdin)

```
<a href="http://www.quickit.co/html/tutorial/html_links.cgi">
example link</a></p>
```

Download

**Test case 2**

Compiler Message

Success

Input (stdin)

```
<div class="more-info">
More Link Examples...</a></div>
```

Download

**Test case 3**

Compiler Message

Success

Input (stdin)

```
<div class="more-info">
More Link Examples...</a></div>
```

Download

**Test case 4**

Compiler Message

Success

Input (stdin)

```
<div class="more-info">
More Link Examples...</a></div>
```

Download

**Test case 5**

Compiler Message

Success

Input (stdin)

```
<div class="more-info">
More Link Examples...</a></div>
```

Download

**Test case 6**

Compiler Message

Success

Input (stdin)

```
asheef
```

Download

45)

HackerRank PREPARE CERTIFY COMPETE

Search Notifications User: prf744

Projects > Beginner > Applications > The British and American Style of Spelling ★ Points: 666 Rank: 2574

Problem Submissions Leaderboard Discussions

You made this submission 5 months ago.  
Score: 15.00 Status: Accepted  
People who solved The British and American Style of Spelling attempted this next:

**Find HackerRank**  
Write a regex to find out if a word contains, start/end or both start and end with hackerank  
[Solve challenge](#)

**Submitted Code**

Language: Python 3 [Open in editor]

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 sen = []
3 import re
4 n = int(input())
5 for i in range(n):
6     x = str(input())
7     sen.append(x)
8 t = int(input())
9 for i in range(t):
10    x = str(input())
11    x = x[:len(x)-2]
12    c = 0
13    p = re.compile(x+'(se|ze)')
14    for j in sen:
15        if p.match(j):
16            c += 1
17 print(c)
```

**Test case 0** Compiler Message: SUCCESS  
Input(stdin): 2  
hackerank is easy to familiarize with  
to familiarize oneself with ui of hackerank is easy  
Expected Output: 1  
familiarize

**Test case 1**

**Test case 2**

**Test case 3**

**Test case 4**

**Test case 5**

**Test case 6**

**Test case 7**

46)

HackerRank PREPARE CERTIFY COMPETE

Search user: ge1744

UK and US: Part 2 ★

Points: 685 Rank: 2574

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**

With a regex, find out if concatenated, shuffled or both start and end with hackerank

**Solve challenge**

**Submitted Code**

Language: Python 3 [Open in editor]

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re
3 sent = []
4 n = int(input())
5 for i in range(n):
6     x = str(input())
7     sent.append(x)
8 t = int(input())
9 for i in range(t):
10    x = str(input())
11    y = x
12    c = 0
13    x = x.replace('our', 'or')
14    p = re.compile('(?:(s|\\A)|' + '(+x+|+y+)' + '(?:(s|\\Z))')
```

**Test case 0**

Compiler Message: SUCCESS

Input (stdin):

```
2
the odour coming out of the leftover food was intolerable
ammonia has a very pungent odor
```

Expected Output:

```
1
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 :

The screenshot shows a web-based C/C++ compiler interface. On the left, there's a sidebar with various links: 'Welcome, Palak Sharma (RA1911033010112)', 'Create New Project', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', and 'Logout'. Below these are social sharing icons for Facebook, Twitter, and LinkedIn, followed by a '150K' link. The main area is a dark blue box containing the output of a lexical analyzer. The output text is as follows:

```
#include is an identifier
<stdio.h> is an identifier
    is an identifier
void is a keyword
main is a keyword
( is a punctuation
) is a punctuation
    is an identifier
{ is a punctuation
int is a keyword
x is an identifier
= is an operator
6 is a number
; is a punctuation
int is a keyword
y is an identifier
= is an operator
4 is a number
; is a punctuation
x is an identifier
= is an operator
x is an identifier
+ is an operator
y is an identifier
; is a punctuation
printf("%d", is an identifier
x); is an identifier
} is a punctuation
```

**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"
        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)

```

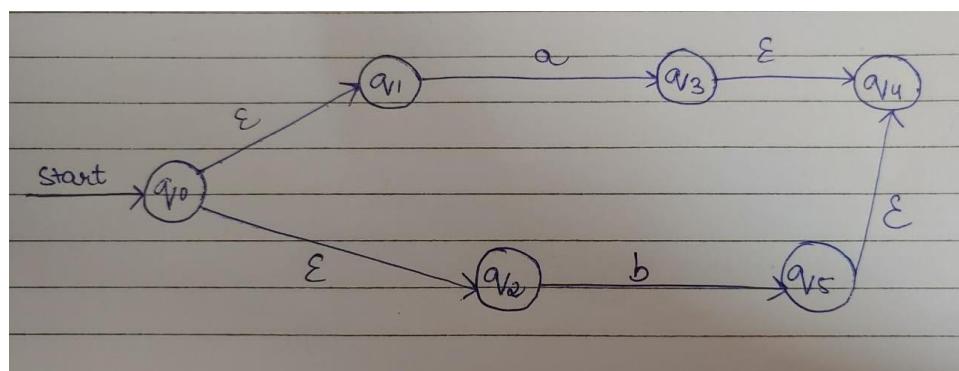
## OUTPUT :

```

Welcome, Palak Sharma
(RA1911033010112) ✨
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
main.py
Enter regex: a*b
State      Symbol      Next state
q0          epsilon    q1 q2
q1          a           q3
q3          epsilon    q1 q2
q2          epsilon    q4
q4          b           q5
...Program finished with exit code 0
Press ENTER to exit console.

```

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 :

The screenshot shows the OnlineGDB interface with a Python script running. The terminal window displays the following output:

```
No. of states : 3
No. of transitions : 2
state name : A
path : ()
Enter end state from state A travelling through path () :
A
path : 1
Enter end state from state A travelling through path 1 :
A B
state name : B
path : ()
Enter end state from state B travelling through path () :
C
path : 1
Enter end state from state B travelling through path 1 :
C
state name : C
path : ()
Enter end state from state C travelling through path () :
path : 1
Enter end state from state C travelling through path 1 :
```

Below the terminal, the script continues to run:

```
Enter end state from state C travelling through path 1 :
NFA :-
{'A': {'()': ['A'], '1': ['A', 'B']}, 'B': {'()': ['C'], '1': ['C']}, 'C': {'()': [], '1': []}}
Printing NFA table :-
() 1
A [A, B]
B [C]
C []
Enter final state of NFA :
C
DFA :-
({'A': {'()': 'A', '1': 'AB'}, 'AB': {'()': 'AC', '1': 'ABC'}, 'AC': {'()': 'A', '1': 'AB'}, 'ABC': {'()': 'AC', '1': 'ABC'})}
Printing DFA table :-
() 1
A   A   AB
AB  AC  ABC
AC  A   AB
ABC AC  ABC
Final states of the DFA are : ['AC', 'ABC']
...Program finished with exit code 0
```

At the bottom right, there is a watermark: "Activate Windows Go to Settings to activate Windows."

## 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:-

$A \rightarrow A\alpha_1 | A\alpha_2 | \dots | A\alpha_m$

$A \rightarrow \beta_1 | \beta_2 | \dots | \beta_n$

Then replace it by

$A \rightarrow \beta_i A' \quad i=1,2,3,\dots,m$

$A' \rightarrow \alpha_j \quad j=1,2,3,\dots,n$

$A' \rightarrow \epsilon$

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<<"\nEnter 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].length())))==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 :

The screenshot shows a web-based IDE interface for OnlineGDB beta. On the left, there's a sidebar with various links like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', and 'Logout'. Below the sidebar are social sharing icons for Facebook, Twitter, and a '150K' follower count.

The main area is a terminal window with the following interaction:

```

Enter number of non terminals: 3
Enter non terminals one by one:
Non terminal 1 : E
Non terminal 2 : T
Non terminal 3 : F
Enter 'esp' for null
Number of E productions: 2
One by one enter all E productions
RHS of production 1: E+T
RHS of production 2: T
Number of T productions: 2

```

Welcome, **Palak Sharma**  
(RA1911033010112) 

Create New Project

My Projects

Classroom new

Learn Programming

Programming Questions

Logout



+ 150K

```
Number of T productions: 2
One by one enter all T productions
RHS of production 1: T*T
RHS of production 2: F
Number of F productions: 2
One by one enter all F productions
RHS of production 1: (E)
RHS of production 2: i
110
<
New set of non-terminals: E T F E' T'
New set of productions:
E -> TE'
T -> FT'
F -> (E)
F -> i
E' -> +TE'
E' -> esp
T' -> *FT'
T' -> esp
...Program finished with exit code 0
```

[FAQ](#) • [Blog](#) • [Terms of Use](#) • [Contact Us](#)

**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:  $A \rightarrow aA_1|aA_2$
4. If found, replace the particular productions with:  
$$\begin{aligned} A &\rightarrow aA' \\ A' &\rightarrow A_1 | A_2 | \epsilon \end{aligned}$$
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 :

The screenshot shows a web-based compiler interface for C/C++ development. On the left, there's a sidebar with various links: 'Welcome, Palak Sharma (RA1911033010112)', 'Create New Project', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', and 'Logout'. Below these are social sharing icons for Facebook, Twitter, and a red button with '+ 150K'.

The main area is a terminal window where a user has run a program to implement Left Factoring. The terminal output is as follows:

```
Enter the no. of nonterminals : 2
Nonterminal 1
Enter the no. of productions : 3
Enter LHS : S
S->iCtSes
S->iCtS
S->a
Nonterminal 2
Enter the no. of productions : 1
Enter LHS : C
C->b

The resulting productions are :
S' -> ?| eS | |
C -> b
S -> iCtSS' | a
```

**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  $\text{FIRST}(x) = \{ 'x' \}$
- > If  $x \rightarrow \epsilon$ , is a production rule, then add  $\epsilon$  to  $\text{FIRST}(x)$ . > If  $X \rightarrow Y_1 Y_2 Y_3 \dots Y_n$  is a production,
  - $\text{FIRST}(X) = \text{FIRST}(Y_1)$
  - If  $\text{FIRST}(Y_1)$  contains  $\epsilon$  then  $\text{FIRST}(X) = \{ \text{FIRST}(Y_1) - \epsilon \} \cup \{ \text{FIRST}(Y_2) \}$
  - If  $\text{FIRST}(Y_i)$  contains  $\epsilon$  for all  $i = 1$  to  $n$ , then add  $\epsilon$  to  $\text{FIRST}(X)$ .
- > 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 ( $S, A, B, \dots$ ) is followed by any terminal ( $a, b, \dots, *, +, (, ) \dots$ ), then add that “terminal” into FOLLOW set.
  - If that non-terminal is followed by any other non-terminal then add “ $\text{FIRST}$  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 already 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 :**

The screenshot shows a web-based compiler interface with a sidebar on the left containing user information and navigation links. The main area displays the results of a grammar analysis.

**Sidebar (Left):**

- Line compiler and debugger for c/c++
- Welcome, **Palak Sharma** (RA1911033010112)
- Create New Project
- My Projects
- Classroom new
- Learn Programming
- Programming Questions
- Logout

**Social sharing icons:** Facebook, Twitter, + 150K

**Analysis Results (Right):**

```
First(E) = { (, i, }

First(R) = { +, #, }

First(T) = { (, i, }

First(Y) = { *, #, }

First(F) = { (, i, }

-----
Follow(E) = { $, ), }

Follow(R) = { $, ), }

Follow(T) = { +, $, ), }

Follow(Y) = { +, $, ), }

Follow(F) = { *, +, $, ), }
```

...Program finished with exit code

## 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  $A \rightarrow \alpha$  in G) {  
    for (each terminal a in FIRST( $\alpha$ ))  
        add  $A \rightarrow \alpha$  to M[A, a];  
    if ( $\epsilon$  is in FIRST( $\alpha$ ))  
        for (each symbol b in FOLLOW(A))  
            add  $A \rightarrow \alpha$  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 :

```

enter the no. of nonterminals
2
enter the productions in a grammar
S->CC
C->eC|d
first
FIRS[S]=ed
FIRS[C]=ed
follow
FOLLOW[S]=$
FOLLOW[C]=ed$
M[S,e]=S->CC
M[S,d]=S->CC
M[C,e]=C->eC
M[C,d]=C->d
...Program finished with exit code 0
Press ENTER to exit console.

```

**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 from 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("-":<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 :

OnlineGDB beta	Stack	Input Buffer	Parsing Action
line compiler and debugger for c/c++			
Welcome, Palak Sharma (RA1911033010112) 🔔	\$i	+i*i	shift
	\$E	+i*i	Reduce S->i
	\$E+	i*i	Shift
	\$E+i	*i	Shift
Create New Project	\$E+E	*i	Reduce S->i
	\$E	*i	Reduce S->E+E
My Projects	\$E*	i	Shift
	\$E*	i	Rejected
Classroom new			
Learn Programming			
Programming Questions			...Program finished with exit code 0 Press ENTER to exit console.

## 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 its 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 :

The screenshot shows two terminal windows on the OnlineGDB platform. The left window displays the user interface with navigation links like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', and 'Logout'. It also shows social sharing icons for Facebook, Twitter, and LinkedIn, and a '150K' follower count.

The right window contains the output of a C++ program. The first part of the output is:

```
LEADING AND TRAILING
Enter the no. of variables : 3
Enter the variables :
E
T
F
```

The second part of the output is:

```
Enter the no. of terminals : 5
Enter the terminals : (
)
*
+
i
```

The third part of the output is:

```
PRODUCTION DETAILS
Enter the no. of production of E:2
E->E+T
E->T
```

The fourth part of the output is:

```
Enter the no. of terminals : 5
Enter the terminals : (
)
*
+
i
```

The fifth part of the output is:

```
PRODUCTION DETAILS
Enter the no. of production of E:2
E->E+T
E->T
```

The sixth part of the output is:

```
Enter the no. of production of T:2
T->T*T
T->F
```

The seventh part of the output is:

```
Enter the no. of production of F:2
F->(E)
F->i
```

The eighth part of the output is:

```
LEADING(E) = (, *, +, i,
LEADING(T) = (, *, i,
LEADING(F) = (, i,
```

The ninth part of the output is:

```
TRAILING(E) = ), *, +, i,
TRAILING(T) = ), *, i,
TRAILING(F) = ), i,
```

The final part of the output is:

```
...Program finished with exit code 0
```

## 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  $S' \rightarrow S \$$  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]={"ABCDEFGHIJKLMNPQR"};
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,'|'};
strcat(g[0].rhs,temp);
cout<<"\n\n augmented 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";

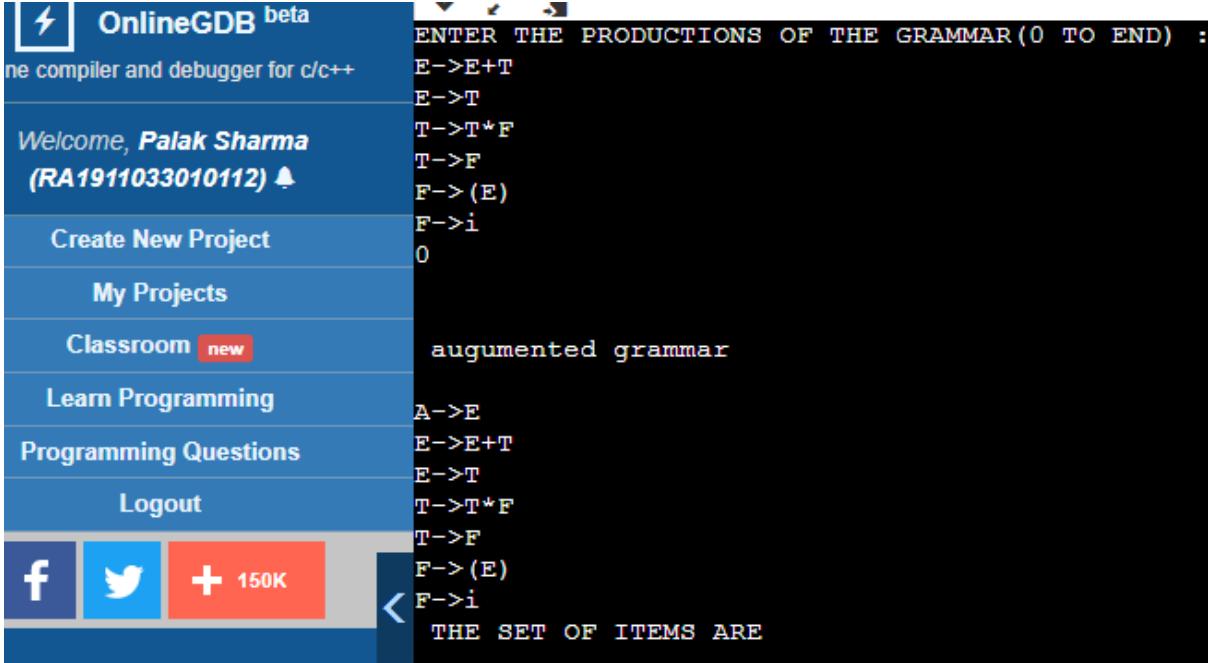
```

```

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:



The screenshot shows the OnlineGDB beta interface. On the left, there's a sidebar with links like 'Create New Project', 'My Projects', 'Classroom new', 'Learn Programming', and 'Programming Questions'. Below the sidebar are social media sharing icons for Facebook, Twitter, and a '150K' follower count. The main area has a terminal-like window with the following text:

```

ENTER THE PRODUCTIONS OF THE GRAMMAR (0 TO END) :
E->E+T
E->T
T->T*T
T->F
F-> ( E )
F->i
0
augumented grammar
A->E
E->E+T
E->T
T->T*T
T->F
F-> ( E )
F->i
THE SET OF ITEMS ARE

```

 GDB  
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\*T  
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\*T  
T->.F  
F->.(E)  
F->.i

 OnlineGDB Beta  
The compiler and debugger for c/c++

Welcome, Palak Sharma  
(RA1911033010112) 

Create New Project  
My Projects  
Classroom new  
Learn Programming  
Programming Questions  
Logout

   150K 

I5  
F->i.  
  
I6  
E->E+.T  
T->.T\*F  
T->.F  
F->. (E)  
F->.i  
  
I7  
T->T\*.F  
F->. (E)  
F->.i  
  
I8  
F->(E.)  
E->E.+T  
  
I9  
E->E+T.  
T->T.\*F  
  
I10  
T->T\*F.  
  
I11  
F->(E).

**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(fPREFIX: {exp_stack[-1]})

return exp_stack[-1]

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

```
pre = infix_to_prefix(expres)
```

```
pos = infix_to_postfix(expres)
```

## OUTPUT :

The screenshot shows a Java application window. On the left, there's a sidebar with navigation links: 'Welcome, Palak Sharma (RA1911033010112)', 'Create New Project', 'My Projects', 'Classroom new', and 'Learn Programming'. The main area has a terminal-like interface. It displays the following text:

```
INPUT THE EXPRESSION: A+B*C/R
PREFIX: +A/*BCR
POSTFIX: ABC*R/+
...Program finished with exit code 0
Press ENTER to exit console.
```

## 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  $a := b \text{ op } c$  perform the various actions. These are as follows:

1. Invoke a function getreg to find out the location L where the result of computation  $b \text{ op } c$  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  $\text{MOV } y', L$  to place a copy of y in L.
3. Generate the instruction  $\text{OP } z', L$  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  $x := y \text{ op } z$  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=%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 :

```

Enter the expression:a=b+c-d
The Intermediate code is:
t1=b+c
t2=t1-d
a=t2
...Program finished with exit code 0
Press ENTER to exit console.

```

**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 variable names 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 for the 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 :

Label	ptr	leftPtr	rightPtr
A	+	x	y
B	*	A	z
C	/	B	x

...Program finished with exit code 0  
Press ENTER to exit console.

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 in static area 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 data objects 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+=1
    self.utilization=(self.sum*100)/self.total
```

```
    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 :

The screenshot shows the OnlineGDB interface with a dark theme. On the left, there's a sidebar with links like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', and 'Logout'. Below the sidebar are social sharing icons for Facebook, Twitter, and LinkedIn.

The main content area displays the output of a C++ program. It starts with the 'First Fit Instance' algorithm:

```
First Fit Instance
Memory block :0 | Size: 100
Memory block :1 | Size: 500
Memory block :2 | Size: 200
Memory block :3 | Size: 300
Memory block :4 | Size: 600
Total 5 Memory blocks Created!
```

It then lists processes and their sizes:

```
Process :0 | Size: 212
Process :1 | Size: 417
Process :2 | Size: 112
Process :3 | Size: 426
Total 4 Processes needs to be allocated
```

Using the First Fit Algorithm, it shows the allocation of processes into memory blocks:

```
Using First Fit Algorithm
Process :0 |Size :212 allocated into Memory block :1 |Size :500
Process :1 |Size :417 allocated into Memory block :4 |Size :600
Process :2 |Size :112 allocated into Memory block :2 |Size :200
Process :3 |Size :426 Unallocated!
```

Finally, it provides memory utilization and a summary for the 'Worst Fit Instance' algorithm:

```
Memory block :0 |Size :100 Empty!
Memory block :3 |Size :300 Empty!
Total Memory Utilization = 43.59%
```

```
Worst Fit Instance
Memory block :0 | Size: 100
Memory block :1 | Size: 500
Memory block :2 | Size: 200
Memory block :3 | Size: 300
Memory block :4 | Size: 600
Total 5 Memory blocks Created!
```

At the bottom, there are links for 'FAQ', 'Blog', 'Terms of Use', and 'Contact Us'.

**OnlineGDB** Beta  
an compiler and debugger for c/c++

Welcome, Palak Sharma (RA1911033010112)

- [Create New Project](#)
- [My Projects](#)
- [Classroom new](#)
- [Learn Programming](#)
- [Programming Questions](#)
- [Logout](#)

f
 t
 + 66.3K

```

Process :0 | Size: 212
Process :1 | Size: 417
Process :2 | Size: 112
Process :3 | Size: 426
Total 4 Processes needs to be allocated

Using worst Fit Algorithm
Process :0 |Size :212 allocated into Memory block :4 |Size :600
Process :1 |Size :417 allocated into Memory block :1 |Size :500
Process :2 |Size :112 allocated into Memory block :3 |Size :300
Process :3 |Size :426 Unallocated!

Memory block :0 |Size :100 Empty!
Memory block :2 |Size :200 Empty!
Total Memory Utilization = 43.59%

Best Fit Instance
Memory block :0 | Size: 100
Memory block :1 | Size: 500
Memory block :2 | Size: 200
Memory block :3 | Size: 300
Memory block :4 | Size: 600
Total 5 Memory blocks Created!

Process :0 | Size: 212
Process :1 | Size: 417
Process :2 | Size: 112
Process :3 | Size: 426
Total 4 Processes needs to be allocated

```

---

Welcome, Palak Sharma (RA1911033010112)

- [Create New Project](#)
- [My Projects](#)
- [Classroom new](#)
- [Learn Programming](#)
- [Programming Questions](#)
- [Logout](#)

f
 t
 + 66.3K

```

Memory block :0 |Size :100 Empty!
Memory block :2 |Size :200 Empty!
Total Memory Utilization = 43.59%

Best Fit Instance
Memory block :0 | Size: 100
Memory block :1 | Size: 500
Memory block :2 | Size: 200
Memory block :3 | Size: 300
Memory block :4 | Size: 600
Total 5 Memory blocks Created!

Process :0 | Size: 212
Process :1 | Size: 417
Process :2 | Size: 112
Process :3 | Size: 426
Total 4 Processes needs to be allocated

Using best Fit Algorithm
Process :0 |Size :212 allocated into Memory block :3 |Size :300
Process :1 |Size :417 allocated into Memory block :1 |Size :500
Process :2 |Size :112 allocated into Memory block :2 |Size :200
Process :3 |Size :426 allocated into Memory block :4 |Size :600

Memory block :0 |Size :100 Empty!
Total Memory Utilization = 68.65%

```

FAQ • Blog • Terms of Use • Contact Us  
GDB Tutorial • Credits • Privacy

...Program finished with exit code 0

**RESULT :** The program was successfully compiled and run.