

**FACULTY OF ENGINEERING AND TECHNOLOGY
SCHOOL OF COMPUTING**

COMPUTER SCIENCE & ENGINEERING



LAB CODE: 18CSC304J

LAB NAME: Compiler Design

**Faculty Incharge:
Mrs.J.Jeyasudha(101566)**

Academic Year: 2021-2022

REGISTER NO: RA1911033010047

NAME: Kolla Seshu Kiran



FACULTY OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(UNDER SECTION 3 OF THE UGC ACT, 1956)
S.R.M. NAGAR, KATTANKULATHUR – 603203
Kancheepuram District

BONAFIDE CERTIFICATE

REGISTER NO RA1911033010047

Certified to Be the Bonafide Record of the work done by
Kolla Seshu Kiran of CINTEL B.Tech degree
course in the practical COMPILER DESIGN in SRM INSTITUTE OF
SCIENCE & TECHNOLOGY, Kattankulathur during the academic
year 2021-2022

Lab Incharge

Date:

HEAD OF DEPARTMENT

Submitted for University Examination held in

SRM INSTITUTE OF SCIENCE &
TECHNOLOGY, at Kattankulathur.

Date

Examiner 1

Examiner2

INDEX

Table of Contents

Serial No.	Title	Date	Marks	Sign
1	Experiment 1 - Implementation of Lexical Analyser	10/1/22		
2	Experiment 2 - Conversion from Regular Expression to NFA	27/1/22		
3	Assignment 1 – Compiler Tool IDEONE	27/1/22		
4	Experiment 3 - Conversion from NFA to DFA	8/2/22		
5	Hackerrank Regex Module (1-20)	8/2/22		
6	Experiment 4 - Elimination of Left Recursion and Left Factoring	17/2/22		
7	Experiment 5 - Computation of FIRST and FOLLOW & Predictive Parsing Table	22/2/22		
8	Assignment 2 – Predictive Parsing	22/2/22		
9	Assignment 3 – SLR Parsing	22/2/22		
10	Experiment 6 - Implementation of Shift Reduce Parsing	4/3/22		
11	Experiment 7 - Computation of LEADING and TRAILING	17/3/22		
12	Experiment 8 - Computation of LR (0) items	24/3/22		
13	Experiment 9 - Intermediate code generation – Postfix, Prefix	31/3/22		
14	Experiment 10 - Intermediate code generation – Quadruple, Triple, Indirect triple	31/3/22		
15	Assignment 4 – Cross Compiler	31/3/22		
16	Hackerrank Regex Module (21-31)	31/3/22		
17	Experiment 11 - Implementation of DAG	7/4/22		
18	Experiment 12 - Three Address Code Generation	7/4/22		
19	Experiment 13 - Implementation of storage allocation strategies	18/4/22		
20	Assignment 5 – Global Dataflow Analysis	18/4/22		
21	Hackerrank Regex Module (32-47)	18/4/22		

EXPERIMENT - 1

Title: Implementation of Lexical Analyser

Aim: To develop lexical analysers to identify identifiers, keywords and operators in c++ programs.

Algorithm:

1. Declare the header files.
2. Open the file, and place a pointer to the first character of it.
3. Read the file until we reach ';' character and store it in a buffer.
4. Using pointers retrieve the tokens from the buffer and store it in a variable.
5. Check whether the token is a delimiter, keyword, operator or a valid identifier using the helper function (is Delimiter, is Operator... etc.).
6. If the token is an identifier and is in a declaration statement add it to the symbol table. If it is not in a statement declaration check whether the identifier is in the symbol table or not. If not, throw an error.
7. If the identifier is not valid, throw an error.
8. In case of no error print the output (type of token).
9. Repeat step 3 until you have reached the end of the file.

RA1911033010047

KOLLA SESHU KIRAN

CODE:

```
1 // Lexical Analyser
2 // RA1911033010047
3 #include<iostream>
4 #include<fstream>
5
6 using namespace std;
7
8 // Declaring language constructs
9 const char delimiters[] = {' ', ';' , '(', ')', ',',
10     '.', '[', ']', '{', '}', ',', '#', '=', ',', '\\',
11     ','};
12
13 const char operators[] = {'+', '-', '*', '^',
14     '/', '>', '<', '=', '.', '%', '&', '|', '~', ','};
15
16 const string binaryOperators[] = {"<<", "++", "--", "+=",
17     "-=", "*=", "/=", ">>", "<<",
18     "||", "!="};
19
20 const string keywords[] = {"if", "else", "while", "do", "break", "continue",
21     "int", "double", "float", "return", "char", "string", "case", "sizeof",
22     "long", "short", "typedef", "switch", "unsigned", "void", "struct", "goto", "static",
23     "class", "for", "continue", "private", "include", "iostream", "using", "namespace", "std", "cout"
24     , "endl", "public"};
25
26 string symbolTable[1024];
27 int symbolIndx = 0;
28
29 bool isDelimiter(char ch) {
30     for(char c: delimiters){
31         if(ch == c){
32             return true;
33         }
34     }
35     return false;
36 }
37
38 bool isOperator(char ch){
39     for(char c: operators){
40         if(ch == c){
41             return true;
42         }
43     }
44     return false;
45 }
46
47 bool isKeyWord(string str){
48     for(string key: keywords){
49         if(key == str){
50             return true;
51         }
52     }
53     return false;
54 }
55
56 bool isNum(char c){
57     int code = (int)c;
58     if(!(48 <= code && code <=57)){
59         return false;
60     }
61     return true;
62 }
63
64 bool isInteger(string str){
65     for(char c: str){
66         if(!isNum(c)){
67             return false;
68         }
69     }
70     return true;
71 }
```

Activate Windows
Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

```
72
73
74- bool isReal(string str){
75    bool hasDecimal = false;
76    for(char c: str){
77        if(c == '.'){
78            hasDecimal = true;
79            continue;
80        }
81        if(!isNum(c)){
82            return false;
83        }
84    }
85    return hasDecimal;
86 }
87
88- bool isAlphabet(char c){
89    if((97 <= c && c <= 122) || (65 <= c && c <= 90)){
90        return true;
91    }
92    return false;
93 }
94
95- bool isBinaryOperator(char op1, char op2){
96    for(string s: binaryOperators){
97        if(s[0] == op1 && s[1] == op2){
98            return true;
99        }
100   }
101   return false;
102 }
103
104- void addSymbol(string s){
105    symbolTable[symbolIndx++] = s;
106 }
107
108 /**
109  * Identifier can contain letter, digits and underscores.
110  * Names must begin only with a letter or an underscore
111  * Names cannot contain whitespace or special characters
112 */
113- bool isIdentifier(string str){
114    if(!isAlphabet(str[0]) && str[0] != '_'){
115        return false;
116    }
117    for(char c: str){
118        if(!isAlphabet(c) && !isNum(c) && (c != '_')){
119            return false;
120        }
121    }
122    return true;
123 }
124
125- bool checkTable(string str){
126    for(int i=0; i<symbolIndx; i++){
127        if(str == symbolTable[i]){
128            return true;
129        }
130    }
131    return false;
132 }
133
134- void parse(string uri){
135    string buff, str;
136    ifstream stream(uri);
137    bool identifier = false;
138    bool comments = false, multilineComment = false;
139    bool rvalue = false;
140
141    if(!stream.is_open()){
142        throw runtime_error("Could not open the file");
143        exit(1);
144    }
```

Activate Windows

Go to Settings to activate Windows.

Activate Windows

Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

```
148-     while(getline(stream, str, ';')){  
149-         identifier = false;  
150-         for(int i=0; i<str.length(); i++){  
151-             // We dont want to show comments, spaces and newline in output hence we're skipping it  
152-             if(i > 0 && str[i] == '/' && str[i-1] == '*'){  
153-                 multilineComment = false;  
154-                 continue;  
155-             }  
156-             if(str[i] == ' ' || str[i] == '\n' || str[i] == '\t' || comments == true  
157-                || multilineComment == true){  
158-                 if(str[i] == '\n')  
159-                     comments = false;  
160-                 buff = "";  
161-                 continue;  
162-             }  
163-             buff = buff + str[i];  
164-             if((i == (str.length() - 1))  
165-                || isDelimiter(str[i+1]) || isOperator(str[i+1])  
166-                || isDelimiter(str[i]) || isOperator(str[i])){  
167-                 if(isDelimiter(buff[0])){  
168-                     if(buff[0] == '\"'){  
169-                         i++;  
170-                         while(str[i] != '\"'){  
171-                             if(str[i] != '\n')  
172-                                 cout << str[i];  
173-                             i++;  
174-                             if(i >= str.length()) throw runtime_error("missing '\"\"");  
175-                         cout << " is string" << endl;  
176-                         continue;  
177-                     }  
178-                     if(buff[0] == '\''){  
179-                         cout << str[i+1] << " is character" << endl;  
180-                         i+=2;  
181-                         cout << (bool)(str[i] != '\\');  
182-                         if(i < str.length() && str[i] != '\\') throw runtime_error("invalid character declaration");  
183-                         continue;  
184-                     }  
185-                 }  
186-                 else if(isOperator(buff[0])){  
187-                     /*  
188-                      In C++ identifier is also valid after ','. Eg: int a, b;  
189-                     */  
190-                     if(identifier == true && buff[0] == ',') {  
191-                         identifier = true;  
192-                         rvalue = false;  
193-                     }  
194-                     if(isBinaryOperator(buff[0], str[i+1])){  
195-                         cout << buff[0] << str[i+1];  
196-                         i+=1;  
197-                     }  
198-                     // Ignoring comments  
199-                     else if(buff[0] == '/' && str[i+1] == '/'){  
200-                         comments = true;  
201-                         i++;  
202-                         continue;  
203-                     }  
204-                 }  
205-             }  
206-         }  
207-     }  
208-     cout << endl;  
209- }
```

Activate Windows
Go to Settings to activate Windows.

```
210- }
```

Activate Windows
Go to Settings to activate Windows.

```
211- }
```

Activate Windows
Go to Settings to activate Windows.

```
212- }
```

Activate Windows
Go to Settings to activate Windows.

```
213- }
```

Activate Windows
Go to Settings to activate Windows.

```
214- }
```

Activate Windows
Go to Settings to activate Windows.

```
215- //
```

Activate Windows
Go to Settings to activate Windows.

```
216- //
```

Activate Windows
Go to Settings to activate Windows.

```
217- //
```

Activate Windows
Go to Settings to activate Windows.

```
218- //
```

Activate Windows
Go to Settings to activate Windows.

```
219- }
```

RA1911033010047

KOLLA SESHU KIRAN

```
220+         } else if(buff[0] == '/' && str[i+1] == '*'){
221             multilineComment = true;
222             i++;
223             continue;
224         }
225     else{
226         cout << buff[0];
227     }
228     cout << " is operator" << endl;
229     if(buff[0] == '=')
230         rvalue = true;
231 }
232 else if(isKeyWord(buff)){
233     cout << buff << " is keyword" << endl;
234     identifier = true;
235 }
236 else if(isInteger(buff)){
237     cout << buff << " is integer" << endl;
238 }
239 else if(isReal(buff)){
240     cout << buff << " is real number" << endl;
241 }
242 else if(isIdentifier(buff)){
243     if(identifier && !rvalue && !checkTable(buff)){
244         addSymbol(buff);
245         identifier = false;
246     } else {
247         if(checkTable(buff)){
248             identifier = true;
249         } else{
250             throw runtime_error(buff + " identifier is not declared");
251             buff = "";
252             continue;
253         }
254     }
255     cout << buff << " is identifier" << endl;
256 }
257 }else {
258     throw runtime_error(buff + " is not valid identifier");
259 }
260 buff = "";
261 }
262 }
263 }
264 }
265 }
266 }
267
268 int main() {
269     // Set the file location
270     parse("./code.txt");
271
272     // cout << "Symbol Table " << endl;
273     // for(int i=0; i<symbolIdx; i++){
274     //     cout << symbolTable[i] << endl;
275     // }
276     return 0;
277 }
```

Activate Windows
Go to Settings to activate Windows.

Activate Windows
Go to Settings to activate Windows.

INPUT:

```
1 //RA1911033010047
2 #include <iostream>
3 using namespace std;
4 class Student {
5 public:
6     int id;
7     string name;
8 };
9 int main () {
10     Student s1; //creating an object of Student
11     s1.id = 201;
12     s1.name = "Sonoo Jaiswal";
13     cout << s1.id << endl;
14     cout << s1.name << endl;
15     return 0;
16 }
```

RA1911033010047

KOLLA SESHU KIRAN

OUTPUT:

```
include is keyword
< is operator
iostream is keyword
> is operator
using is keyword
namespace is keyword
std is keyword
class is keyword
Student is identifier
public is keyword
int is keyword
id is identifier
string is keyword
name is identifier
int is keyword
main is identifier
Student is identifier
s1 is identifier
s1 is identifier
. is operator
id is identifier
= is operator
201 is integer
s1 is identifier
. is operator
name is identifier
= is operator
Soneoo Jaiswal is string
cout is keyword
<< is operator
s1 is identifier
```

```
. is operator           I
id is identifier
<< is operator
endl is keyword
cout is keyword
<< is operator
s1 is identifier
. is operator
name is identifier
<< is operator
endl is keyword
return is keyword
0 is integer

...Program finished with exit code 0
Press ENTER to exit console.[]
```

Activate Windows
Go to Settings to activate Windows.

Activate Windows
Go to Settings to activate Windows.

Result: The program for lexical analysis is implemented successfully and obtained successfully.

EXPERIMENT- 2

Title: Conversion of Regular Expression to Non deterministic finite automata.

Aim: To design a program that accepts a regular expression and returns the transition table for an NFA.

Algorithm:

1. Start the program
2. The user is prompted to enter a regular expression.
3. Initialize separate variables and functions for converting the regex to postfix and to display the NFA.
4. Create separate methods for different operators like +, *, ..
5. By using Switch case Initialize different cases for the input
6. For the ' .' operator Initialize a separate method by using various stack functions and do the same for the other operators like ' * ' and ' + '.
7. The regular expression is in the form of a.b (or) a+b
8. Display the output
9. Stop.

RA1911033010047

KOLLA SESHU KIRAN

CODE:

```
1 //RA1911033010047
2 #include <iostream>
3 #include <stack>
4 #include <iomanip>
5
6 int init[20],fin[20],a=0,b=0;
7
8 std::string simplify_input(std::string s);
9 std::string postfix(std::string s);
10 int reg_nfa(std::string s,int nfa_table[100][5]);
11 void print_initial_final();
12 void print_nfa_table(int nfa_table[100][5],int states);
13 void initialise(int nfa_table[100][5]);
14 void reduce_fin(int x);
15
16
17 int main() {
18
19     while(true){
20         int nfa_table[100][5];
21         initialise(nfa_table);
22         int states{};
23
24         std::cout << "Enter your Regular Expression:" << std::endl;
25         std::cout << "-- ex: a+b, ab, a* --" << std::endl;
26         std::cout << "-- Enter 'exit' to exit the program --" << std::endl;
27         std::cout << ">>> ";
28
29
30         std::string s{};
31         std::cin >> s;
32
33         if(s=="exit"){
34             break;
35         }
36
37         s = simplify_input(s);
38         // std::cout << s << std::endl;
39
40         s = postfix(s);
41         // std::cout << s << std::endl;
42
43         states = reg_nfa(s, nfa_table);
44         print_nfa_table(nfa_table, states);
45     }
46
47
48     return 0;
49 }
50
51 void print_nfa_table(int nfa_table[][5],int states){
52     std::cout << std::endl;
53     for(int i{}; i<60; i++){
54         std::cout << "-";
55     }
56     std::cout << std::endl<< std::endl;
57     std::cout << std::setw(43) << "TRANSITION TABLE FOR NFA" << std::endl << std::endl;
58     std::cout << std::setw(10) << "States" << std::setw(10) << "a" << std::setw(10) << "b" << std::setw(10) << "e" << std::setw(10)
59     for(int i{}; i<60; i++){
60         std::cout << "-";
61     }
62     std::cout << std::endl;
63     for(int i{1}; i<states; i++){
64         for(int j{0}; j<5; j++){
65             if(nfa_table[i][j] == -1)
66                 std::cout << std::setw(10) << "~";
67             else
68                 std::cout << std::setw(10) << nfa_table[i][j];
69         }
70         std::cout << std::endl;
71     }
72 }
```

Activate Windows

Activate Windows

Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

```
73     std::cout << std::endl;
74     for(int i{}; i<60; i++){
75         std::cout << "-";
76     }
77     std::cout << std::endl;
78
79     print_initial_final();
80 }
81
82
83 void print_initial_final(){
84     std::cout << std::endl;
85     std::cout << "Initial state is: ";
86     for(int i{}; i<a; i++){
87         std::cout << init[i] << " ";
88     }
89     std::cout << std::endl;
90     std::cout << "Final state(s) are: ";
91     for(int i{}; i<b; i++){
92         std::cout << fin[i] << " ";
93     }
94     std::cout << std::endl << std::endl << std::endl;
95 }
96
97
98 void reduce_fin(int x){
99     for(int i=x; i<b-1; i++){
100         fin[i] = fin[i+1];
101     }
102     b -= 1;
103 }
104
105
106 int reg_nfa(std::string s,int nfa_table[100][5]){
107     int l = s.length();
108     int states = 1;
109     int m,n,j,counter;
```

Activate Windows
Go to Settings to activate Windows.

Activate Windows
Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

```
147     m = init[j-counter];
148     nfa_table[states][3+counter]=m;
149   }
150   a=a-2;
151   init[a]=states;
152   a += 1;
153   nfa_table[states][0] = states;
154   states += 1;
155   for(j=b-1,counter=0;counter<2;counter++){
156     m = fin[j-counter];
157     nfa_table[m][3]=states;
158   }
159   b=b-2;
160   fin[b]=states;
161   b += 1;
162   nfa_table[states][0] = states;
163   states += 1;
164   break;
165 }
166 case '*': {
167   m = init[a-1];
168   nfa_table[states][3] = m;
169   nfa_table[states][0] = states;
170   init[a-1] = states;
171
172   states += 1;
173   n = fin[b-1];
174   nfa_table[n][3]=m;
175   nfa_table[n][4]=states;
176   nfa_table[states-1][4]=states;
177   fin[b-1]=states;
178   nfa_table[states][0]=states;
179   states += 1;
180   break;
181 }
182 }
183 }
```

Activate Windows
Go to Settings to activate Windows.

OUTPUT:

```
Enter your Regular Expression:
-- ex: a+b, ab, a* --
-- Enter 'exit' to exit the program --
>>> a+b
```

TRANSITION TABLE FOR NFA

States	a	b	e	e	I
1	2	~	~	~	
2	~	~	6	~	
3	~	4	~	~	
4	~	~	6	~	
5	~	~	3	1	
6	~	~	~	~	

```
Initial state is: 5
Final state(s) are: 6
```

Activate Windows
Go to Settings to activate Windows.

RA1911033010047

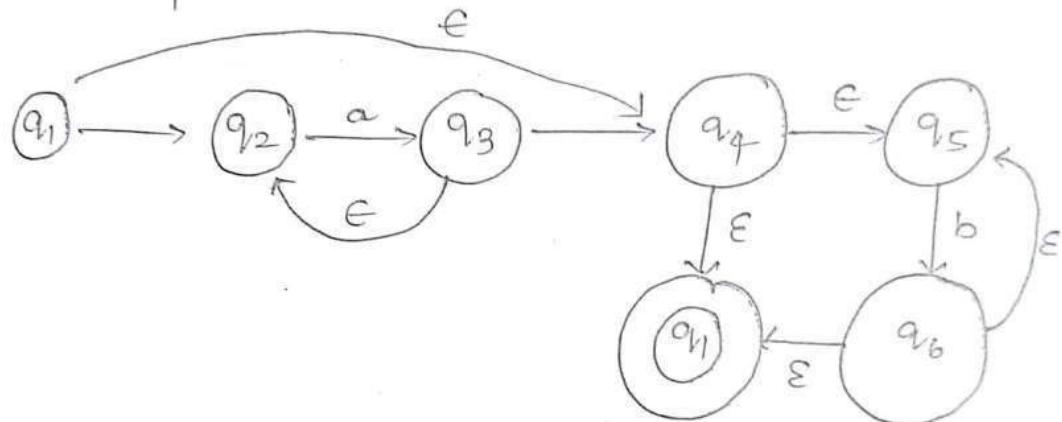
KOLLA SESHU KIRAN

Manual Calculation:

* Experiment 2 *

RE to NFA

Regular Expression taken : $a+b$



Result: The program to convert a regular expression to non deterministic finite automata.

RA1911033010047

kolla · Seshu Kiran

* UNIT - I *

* Assignment - I *

* COMPILER DESIGN *

27/1/22

ONLINE COMPILER TOOL

IDEONE:

it is a free online compiler and debugger, specifically designed for programmers and developers, where you can compile and run code with your custom input in just a matter of seconds. You can compile any programming language with ideone. It also has some awesome features like you can make your code secret to share with specific coders, and is the best platform to share your code with your project members.

- Very easy to load and a fast platform to quick checking of your code.
- Compile in more than 60 programming languages.
- Saves your codes in an organized manner and share it via the provided link.
- A fully-featured debugging tool that makes debugging easy and fun.
- Provides the functionality to make code public, private or secret.

- Embed source code with the auto-generated JavaScript code.
- You can enter some notes with your code for an easy understanding of the viewers.
- Useful shortcuts for easy and fast operation
- The Ideone platform is cost-free and can be used by anyone.
- It supports cloud-based platform.
- Ideone is an online compiler and a debugger. It allows us to compile source code and execute it online. and supports more than 60 programming languages.
- It is an online compiler.
- Free compiler and debugger.
- Supports 60 different programming languages.
- We can choose the programming language and enter the source code and execute the program.
- Options to read input data from standard input are present.
- It is an online IDE, when a compiler is integrated with IDE, we get the entire package at one place so that we can complete the code, compile, debug and execute the program in the same software.

RA1911033010047

KOLLA SESHU KIRAN

HACKERRANK REGEX SOLUTIONS(1 - 20)

Date: 8-02-2022

The screenshot shows a challenge titled "Matching Specific String" on the HackerRank platform. The challenge has a difficulty rating of ★ and a status of Accepted with a score of 5.00. The code submitted is:

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

The challenge interface includes tabs for Problem, Submissions, Leaderboard, Discussions, and Editorial. A sidebar on the right provides help links: View discussions, View editorial, and View top submissions. The challenge details also mention Points: 710 and Rank: 1. Below the challenge area, there's a section for Test cases, Compiler Message, Input (stdin), and Expected Output.

Test case 0: Compiler Message: Success

Test case 1: Input (stdin):

```
1 The hackerrank team is on a mission to flatten the world by
    restructuring the DNA of every company on the planet. We rank
    programmers based on their coding skills, helping companies source
    great programmers and reduce the time to hire. As a result, we are
    revolutionizing the way companies discover and evaluate talented
    engineers. The hackerrank platform is the destination for the best
    engineers to hone their skills and companies to find top engineers.
```

Test case 2: Input (stdin):

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

Download buttons are available for both the compiler message and expected output sections.

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search RA1911033010047

Prepare > Regex > Introduction > Matching Digits & Non-Digit Characters

Matching Digits & Non-Digit Characters ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.

Score: 5.00 Status: Accepted

Submitted Code

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

```
1 Regex_Pattern = r"\d\d(.)\d\d(.)\d\d\d\d" # Do not delete 'r'.
```

NEED HELP?

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

Activate Window Go to Settings to activate

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input (stdin) Download
1 06-11-2015

Test case 3

Test case 4 Expected Output Download
1 true

Test case 5

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search RA1911033010047

Prepare > Regex > Introduction > Matching Anything But a Newline

Matching Anything But a Newline ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 3 months ago.

Score: 5.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

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

NEED HELP?

View discussions View editorial View top submissions

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input (stdin) Download

```
1 123.456.abc.def
```

Test case 3

Test case 4 Expected Output Download

```
1 true
```

Test case 5

Test case 6

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search RA1911033010047

Prepare > Regex > Introduction > Matching Whitespace & Non-Whitespace Character

Matching Whitespace & Non-Whitespace Character ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.

Score: 5.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

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

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 12 11 15
```

Test case 3 Output

Test case 4 Expected Output Download

```
1 true
```

Test case 5 Output



RA1911033010047

KOLLA SESHU KIRAN

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search Profile RA1911033010047

Prepare > Regex > Introduction > Matching Start & End

Matching Start & End ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.

Score: 5.00 Status: Accepted

Submitted Code

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

```
1 Regex_Pattern = r"^\d\S{4}\.$" # Do not delete 'r'.
```

NEED HELP?

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

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input(stdin) Download
1 0qwer.

Test case 3

Test case 4 Expected Output Download
1 true

Test case 5

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Character Class > Matching Specific Characters ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.
Score: 10.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 Regex_Pattern = r'^[123][120][xs0][30Aa][xsu][.,]$' # Do not delete
'f'
2
3
```

NEED HELP?

View discussions
 View editorial
 View top submissions

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input (stdin) Download

```
1 1203x.
```

Test case 3

Test case 4 Expected Output Download

```
1 true
```

Test case 5

Test case 6

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search Profile RA1911033010047

Prepare > Regex > Character Class > Excluding Specific Characters

Excluding Specific Characters ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.

Score: 10.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 Regex_Pattern = r'^[^d][^aeiou][^bcDF][^s][^AEIOU][^.,,]$' # Do not
2 delete 'r'.
3
```

NEED HELP?

View discussions View editorial View top submissions

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input(stdin) Download
1 think?

Test case 3

Test case 4 Expected Output Download
1 true

Test case 5

Test case 6

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search Profile RA1911033010047

Prepare > Regex > Character Class > Matching Character Ranges

Matching Character Ranges ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.
Score: 10.00 Status: Accepted

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
2
3
```

NEED HELP?

View discussions View editorial 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

Input(stdin) Download

```
1 h4ckR
```

Expected Output Download

```
1 true
```

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE

Search Profile RA1911033010047

Prepare > Regex > Repetitions > Matching {x} Repetitions

Matching {x} Repetitions ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.

Score: 20.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 Regex_Pattern = r'^[a-zA-Z02468]{40}[\s13579]{5}$' # Do not delete
2
3
```

NEED HELP?

View discussions View editorial View top submissions

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input(stdin) Download

```
1 22222222aaaaaaa22222222aaaaaaa13 57
```

Test case 3

Test case 4 Expected Output Download

```
1 true
```

Test case 5

Test case 6

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Repetitions > Matching {x,y} Repetitions.

Matching {x, y} Repetitions ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

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

Submitted Code

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

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

NEED HELP?

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

Test case 0 Compiler Message: Success

Test case 1 Input (stdin) Download

```
1 3threeormorealphabets.
```

Test case 2 Expected Output Download

```
1 true
```

Test case 3

Test case 4

Test case 5

Test case 6

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE RA1911033010047

Prepare > Regex > Repetitions > Matching Zero Or More Repetitions

Matching Zero Or More Repetitions ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.

Score: 20.00 Status: Accepted

Submitted Code

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

```
1 Regex_Pattern = r'^\d{2,}[a-z]*[A-Z]*$' # Do not delete 'r'.
2
3
```

NEED HELP?

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

Test case 0	Compiler Message
Success	
Test case 1	
Test case 2	Input(stdin) Download
1 14	
Test case 3	
Test case 4	Expected Output Download
1 true	
Test case 5	
Test case 6	

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Repetitions > Matching One Or More Repetitions

Matching One Or More Repetitions ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

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

Submitted Code

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

```
1 Regex_Pattern = r'^\d+[A-Z]+[a-z]+$' # 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 1Qa
```

Test case 3

Test case 4 Expected Output Download

```
1 true
```

Test case 5

Test case 6

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Repetitions > Matching Ending Items.

Matching Ending Items ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

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

Submitted Code

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

```
1 Regex_Pattern = r'^[a-zA-Z]*$' # 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 Kites

Test case 3 [Input\(stdin\)](#) [Download](#)

Test case 4 [Input\(stdin\)](#) [Download](#)
1 true

Test case 5 [Input\(stdin\)](#) [Download](#)

Test case 6 [Input\(stdin\)](#) [Download](#)

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search Profile RA1911033010047

Prepare > Regex > Grouping and Capturing > Matching Word Boundaries ★ Points: 710 Rank: 1

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

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 Success

Test case 1

Test case 2 Input(stdin) Download
1 Found any match?

Test case 3

Test case 4 Expected Output Download
1 true

Test case 5

Test case 6

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

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Grouping and Capturing > Capturing & Non-Capturing Groups
Capturing & Non-Capturing Groups ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

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

Submitted Code

Language: Python 3 Open in editor

```
1 Regex_Pattern = r'(ok){3,}' # 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 okokok! cya
```

Test case 3

Test case 4 Expected Output Download

```
1 true
```

Test case 5

Test case 6

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE

Search Profile RA1911033010047

Prepare > Regex > Grouping and Capturing > Alternative Matching

Alternative Matching ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.

Score: 20.00 Status: Accepted

Submitted Code

Language: Python 3 [1 Open in editor](#)

```
1 Regex_Pattern = r'^^(Mr\.|Mrs\.|Ms\.|Dr\.|Er\.) ([a-zA-Z]+)$' # Do not
  delete 'r'.
2
3
```

NEED HELP?

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

Test case 0 Compiler Message Success

Test case 1 Input(stdin) Download

```
1 Mr.DOSHI
```

Test case 2 Expected Output Download

```
1 true
```

Test case 3

Test case 4

Test case 5

Test case 6

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search Profile RA1911033010047

Prepare > Regex > Backreferences > Matching Same Text Again & Again

Matching Same Text Again & Again ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission a month ago.

Score: 20.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 Regex_Pattern = r'^([a-z])([a-zA-Z_])(\s)([^a-zA-Z])(\d)(\D)([A-Z])
2 ([a-zA-Z])([aeiouAEIOU])(\$)\1\2\3\4\5\6\7\8\9\$'
3
```

NEED HELP?

View discussions View editorial View top submissions

Test case 0 Compiler Message Success

Test case 1 Input (stdin) Download

```
1 ab #i?AZa$ab #17AZa$
```

Test case 2 Expected Output Download

```
1 true
```

Test case 3

Test case 4

Test case 5

Test case 6

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search Profile RA1911033010047

Prepare > Regex > Backreferences > Backreferences To Failed Groups ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

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

Submitted Code

Language: Python 3 Open in editor

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

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 12-34-56-78

Test case 3

Test case 4 Expected Output Download
1 true

Test case 5

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search RA1911033010047

Prepare > Reges > Backreferences > Branch Reset Groups

Branch Reset Groups ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

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

Submitted Code

Language: PHP

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

NEED HELP?

View discussions
 View editorial
 View top submissions

Test case 0 Compiler Message Success

Test case 1 Input(stdin) Download

```
1 12-34-56-78
```

Test case 2 Expected Output Download

```
1 true
```

Test case 3

Test case 4

Test case 5

Test case 6

EXPERIMENT- 3

Title: Conversion of Non deterministic finite automata to deterministic finite automata.

Aim: To implement a program to convert a Non-Deterministic Finite Automata to a Deterministic Finite Automata.

Algorithm:

1. Take the number of states, transition as well as the NFA table as input (state name, path, end state).
2. Store all the input in a data Frame, which is a 2-dimensional array to represent the NFA table. Display the NFA table to the user.
3. Iterate through each state along with its transitions for all input paths.
 - a. If there is a transition with multiple endpoints for a particular state for a particular input, then create a new state by combining the end points and find out the transition for the new state of a particular input by union (joining) of the end points transitions, for that particular input. Repeat this for all inputs of the new state to find out the transitions. Store the new state along with its transitions in the final DFA table as well as add it to the current NFA table. Also change the transition for the current state for the current input to the new state.
 - b. If transition has a single endpoint for a particular input then make no changes and add it to the final DFA table.
4. Once we have completed iterating through all the states, print out the final DFA table.

RA1911033010047

KOLLA SESHU KIRAN

CODE:

```
1 #RA1911033010047
2 import pandas as pd
3
4 nfa = {}
5 n = int(input("Enter no. of states : "))
6 t = int(input("Enter no. of transitions : "))
7 for i in range(n):
8     state = input("Enter state name : ")
9     nfa[state] = {}
10    for j in range(t):
11        path = input("Enter path : ")
12        print("Enter end state from state {} travelling through path {} : ".format(
13            state, path))
14        reaching_state = [x for x in input().split()]
15        nfa[state][path] = reaching_state
16
17
18 print("\nNFA table :- ")
19 nfa_table = pd.DataFrame(nfa)
20 print(nfa_table.transpose())
21
22 print("\nEnter final state of NFA : ")
23 nfa_final_state = [x for x in input().split()]
24
25 new_states_list = []
26
27 dfa = {}
28 keys_list = list(
29     list(nfa.keys())[0])
30 path_list = list(nfa[keys_list[0]].keys())
31
32 dfa[keys_list[0]] = {}
33 for y in range(t):
34     var = "".join(nfa[keys_list[0]][
35         path_list[y]])
36     dfa[keys_list[0]][path_list[y]] = var
37
38     if var not in keys_list:
39         new_states_list.append(var)
40         keys_list.append(var)
41
42 while len(new_states_list) != 0:
43     dfa[new_states_list[0]] = {}
44     for _ in range(len(new_states_list[0])):
45         for i in range(len(path_list)):
46             temp = []
47             for j in range(len(new_states_list[0])):
48                 temp += nfa[new_states_list[0][j]][path_list[i]]
49             s = ""
50             s = s.join(temp)
51             if s not in keys_list:
52                 new_states_list.append(s)
53                 keys_list.append(s)
54             dfa[new_states_list[0][i]][path_list[i]] = s
55
56     new_states_list.remove(new_states_list[0])
57
58 print("\nDFA table :- ")
59 dfa_table = pd.DataFrame(dfa)
60 print(dfa_table.transpose())
61
62 dfa_states_list = list(dfa.keys())
63 dfa_final_states = []
64 for x in dfa_states_list:
65     for i in x:
66         if i in nfa_final_state:
67             dfa_final_states.append(x)
68             break
69
70 print("\nFinal states of the DFA are : ", dfa_final_states)
```

Activate Windows

Go to Settings to activate Windows.

Activate Windows

Go to Settings to activate Windows.

Output:

```
No. of states : 3
No. of transitions : 2
state name : A
path : 0
Enter end state from state A travelling through path 0 :
A
path : 1
Enter end state from state A travelling through path 1 :
AB
state name : B
path : 0
Enter end state from state B travelling through path 0 :
C
path : 1
Enter end state from state B travelling through path 1 :
C
state name : C
path : 0
Enter end state from state C travelling through path 0 :

path : 1
Enter end state from state C travelling through path 1 :
NFA :-  

('A': {'0': ['A'], '1': ['AB']}, 'B': {'0': ['C'], '1': ['C']}, 'C': {'0': [], '1': []})  

Printing NFA table :-  

      0   1  

A [A] [AB]  

B [C] [C]  

C [] []
Enter final state of NFA :
C

DFA :-  

('A': {'0': 'A', '1': 'AB'}, 'AB': {'0': 'AC', '1': 'ABC'}, 'AC': {'0': 'A', '1': 'AB'}, 'ABC': {'0': 'AC', '1': 'ABC'})  

Printing DFA table :-  

      0   1  

A    A   AB  

AB   AC  ABC  

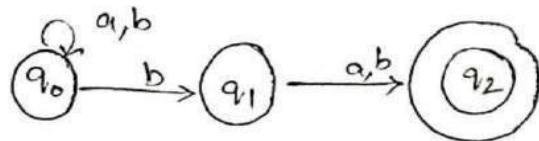
AC   A   AB  

ABC  AC  ABC  

Final states of the DFA are : ['AC', 'ABC']

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

NFA?



8 transition states possible.

		a	b
A	\emptyset	\emptyset	\emptyset
B	q_0	q_0	$q_0 q_1$
C	q_1	q_2	q_2
D	q_2^*	\emptyset	\emptyset
E	$q_0 q_1$	$q_0 q_2$	$q_0 q_1 q_2$
F	$q_1 q_2^*$	q_2	q_2
G	$q_0 q_2^*$	q_0	$q_0 q_1$
H	$q_0 q_1 q_2^*$	$q_0 q_2$	$q_0 q_1 q_2$

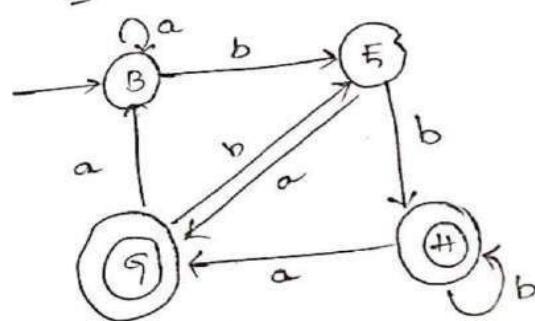
Renaming

	a	b
A	A	A
B	B	E
C	D	D
D*	A	A
E	G	H
F	D	D
G	B	F
H	G	H

To draw the DFA

	a	b
B	B	E
E	G	H
G	B	F
H	G	H

DFA :=

**Result:** The program was successfully implemented.

EXPERIMENT- 4

Title: Elimination of Left Recursion and Left Factoring

Aim: To write a program to eliminate the left recursion and left factoring for a given grammar.

Algorithm:

Left Recursion:

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 lexeme is stored in A variable.
 - b. We again split the right side lexemes on basis of ‘|’ character and store it in a list (lexemes).
 - c. For each lexeme on the lexemes list we check if the A string is in the left substring of the lexeme, if so it’s a left recursion case and we store the remaining portion of the lexeme in the alpha list.
 - d. If there is no left recursion we store it in the beta list. e. Repeat this for all the lexemes on the right side.
4. If the alpha list is empty it means there is no left recursion in the production rule, hence we print the production rule as it is.
5. If the alpha list is not empty there is left recursion and we print the production rule in the form of: $A \rightarrow \beta_1 A' | \beta_2 A' | \dots | \beta_n A'$ $A' \rightarrow \alpha_1 A' | \alpha_2 A' | \dots | \alpha_m A'$
Where $\alpha_1, \alpha_2, \dots, \alpha_m$ are the elements stored in the alpha list and $\beta_1, \beta_2, \dots, \beta_n$ are the elements in the beta list.
6. Repeat step 2 for all the production rules.
7. Stop.

RA1911033010047

KOLLA SESHU KIRAN

CODE FOR LEFT RECURSION:

```
1 #RA1911033010047
2 gram = {
3     "E":["E+T","T"],
4     "T":["T+F","F"],
5     "F":["(E)","i"]
6 }
7
8 def removeDirectLR(gramA, A):
9     temp = gramA[A]
10    tempCr = []
11    tempInCr = []
12    for i in temp:
13        if i[0] == A:
14            tempInCr.append(i[1:]+[A+"'"])
15        else:
16            tempCr.append(i+[A+"'"])
17    tempInCr.append("e")
18    gramA[A] = tempCr
19    gramA[A+"'"] = tempInCr
20    return gramA
21
22
23 def checkForIndirect(gramA, a, ai):
24     if ai not in gramA:
25         return False
26     if a == ai:
27         return True
28     for i in gramA[ai]:
29         if i[0] == ai:
30             return False
31         if i[0] in gramA:
32             return checkForIndirect(gramA, a, i[0])
33     return False
34
35 def rep(gramA, A):
36     temp = gramA[A]
37     newTemp = []
38     for i in temp:
39
40         if checkForIndirect(gramA, A, i[0]):
41             t = []
42             for k in gramA[i[0]]:
43                 t=[]
44                 t+=k
45                 t+=i[1:]
46                 newTemp.append(t)
47         else:
48             newTemp.append(i)
49     gramA[A] = newTemp
50     return gramA
51
52 def rem(gram):
53     c = 1
54     conv = {}
55     gramA = {}
56     revconv = {}
57     for j in gram:
58         conv[j] = "A"+str(c)
59         gramA["A"+str(c)] = []
60         c+=1
61
62     for i in gram:
63         for j in gram[i]:
64             temp = []
65             for k in j:
66                 if k in conv:
67                     temp.append(conv[k])
68                 else:
69                     temp.append(k)
70                 gramA[conv[i]].append(temp)
71
72     for i in range(c-1,0,-1):
73         ai = "A"+str(i)
74         for j in range(0,i):
75             aj = gramA[ai][0][0]
76             if aj==aj :
```

Activate Windows
Go to Settings to activate Windows.

Activate Windows
Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

```
76+         if ail==aj :
77+             if aj in gramA and checkForIndirect(gramA,ai,aj):
78+                 gramA = rep(gramA, ai)
79+
80+     for i in range(1,c):
81+         ai = "A"+str(i)
82+         for j in gramA[ai]:
83+             if ai==j[0]:
84+                 gramA = removeDirectLR(gramA, ai)
85+                 break
86+
87+     op = {}
88+     for i in gramA:
89+         a = str(i)
90+         for j in conv:
91+             a = a.replace(conv[j],j)
92+         revconv[i] = a
93+
94+     for i in gramA:
95+         l = []
96+         for j in gramA[i]:
97+             k = []
98+             for m in j:
99+                 if m in revconv:
100+                     k.append(m.replace(m,revconv[m]))
101+                 else:
102+                     k.append(m)
103+             l.append(k)
104+         op[revconv[i]] = l
105+
106+     return op
107+
108 result = rem(gram)
109
110 for i in result:
111     print(f'{i} -> {result[i]}')
```

Activate Windows
Go to Settings to activate Windows.

Output:

```
E->[['T', "E'"]]
T->[['F', "T'"]]
F->[['(', 'E', ')'], ['i']]
E'->[['+', 'T', "E'"], ['e']]
T'->[['*', 'F', "T'"], ['e']]

...Program finished with exit code 0
Press ENTER to exit console.[]
```

Activate Windows
Go to Settings to activate Windows.

Algorithm for Left Factoring:

1. Start.
2. First we take the production rule as an input from the user. We also sanitize the input from spaces.
3. We split the lexemes on the left and the right side of the “->” symbol.
4. We again split the right side lexemes on basis of ‘|’ character and store it in a list (lexemes).
5. For all the lexemes on the lexemes list we find out the common prefix.
6. We print the first production in the form of if there exists a common prefix:

$A \rightarrow \alpha A'$

where alpha(α) represents a common prefix.

7. We remove the common prefix from all the lexemes having it (stores in beta list). We make this as our new lexeme list.
 8. We repeat step 5 until there is no common prefix.
 9. If the beta list is not empty, we print the final production rule in the form of:
 $A' \rightarrow \beta_1 | \beta_2 | \dots | \beta_n | \epsilon$
- where $\beta_1, \beta_2, \dots, \beta_n$ are the elements in the beta list containing the remaining portion of the string after removing all the prefixes.

RA1911033010047

KOLLA SESHU KIRAN

CODE FOR LEFT FACTORING:

Output:

```
S->aZ'  
Z'->e  
S->iEiSY'  
Y'->e |eS  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

RA1911033010047

KOLLA SESHU KIRAN

Result: The programs for both the elimination left recursion and left factoring are successfully implemented.

EXPERIMENT - 5

Title: First and follow

Aim: To write a program to find the first and follow of a given grammar.

Algorithm:

For computing the first:

- 1) If X is a terminal then

$$\text{FIRST}(X) = \{X\}$$

Example: $F \rightarrow I \mid id$

We can write it as $\text{FIRST}(F) \rightarrow \{ (, id)$

- 2) If X is a non-terminal like $E \rightarrow T$ then to get $\text{FIRST}(E)$ substitute T with other productions until you get a terminal as the first symbol
- 3) If $X \rightarrow \epsilon$ then add ϵ to $\text{FIRST}(X)$.

For computing the follow:

1. Always check the right side of the productions for a non-terminal, whose FOLLOW set is being found. (never see the left side).
2. (a) If that non-terminal (S,A,B...) is followed by any terminal (a,b...,*,+,(),...) , then add that terminal into the FOLLOW set.
(b) If that non-terminal is followed by any other non-terminal then add FIRST of other nonterminals into the FOLLOW set.

RA1911033010047

KOLLA SESHU KIRAN

Code:

```
1 #RA1911033010047
2 gram = {
3     "E":["E+T","T"],
4     "T":["T+F","F"],
5     "F":["(E)","id"]
6 }
7
8 def removeDirectLR(gramA, A):
9     temp = gramA[A]
10    tempCr = []
11    tempInCr = []
12    for i in temp:
13        if i[0] == A:
14            #tempInCr.append(i[1:])
15            tempInCr.append(i[1:]+[A+"'"])
16        else:
17            #tempCr.append(i)
18            tempCr.append(i+[A+"'"])
19    tempInCr.append(["e"])
20    gramA[A] = tempCr
21    gramA[A+"'"] = tempInCr
22    return gramA
23
24
25 def checkForIndirect(gramA, a, ai):
26    if ai not in gramA:
27        return False
28    if a == ai:
29        return True
30    for i in gramA[ai]:
31        if i[0] == ai:
32            return False
33        if i[0] in gramA:
34            return checkForIndirect(gramA, a, i[0])
35    return False
36
37 def rep(gramA, A):
38     temp = gramA[A]
39
40     newTemp = []
41     for i in temp:
42         if checkForIndirect(gramA, A, i[0]):
43             t = []
44             for k in gramA[i[0]]:
45                 t=[]
46                 t+=k
47                 t+=i[1:]
48                 newTemp.append(t)
49
50             else:
51                 newTemp.append(i)
52     gramA[A] = newTemp
53     return gramA
54
55 def rem(gram):
56     c = 1
57     conv = {}
58     gramA = {}
59     revconv = {}
60     for j in gram:
61         conv[j] = "A"+str(c)
62         gramA["A"+str(c)] = []
63         c+=1
64
65     for i in gram:
66         for j in gram[i]:
67             temp = []
68             for k in j:
69                 if k in conv:
70                     temp.append(conv[k])
71                 else:
72                     temp.append(k)
73             gramA[conv[i]].append(temp)
74
75     for i in range(c-1,0,-1):
76         ai = "A"+str(i)
```

Activate Windows

Go to Settings to activate Windows.

Activate Windows

Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

```
79+         if ai!=aj :
80+             if aj in gramA and checkForIndirect(gramA,ai,aj):
81+                 gramA = rep(gramA, ai)
82+
83+     for i in range(1,c):
84+         ai = "A"+str(i)
85+         for j in gramA[ai]:
86+             if ai==j[0]:
87+                 gramA = removeDirectLR(gramA, ai)
88+                 break
89+
90+     op = {}
91+     for i in gramA:
92+         a = str(i)
93+         for j in conv:
94+             a = a.replace(conv[j],j)
95+         revconv[i] = a
96+
97+     for i in gramA:
98+         l = []
99+         for j in gramA[i]:
100+             k = []
101+             for m in j:
102+                 if m in revconv:
103+                     k.append(m.replace(m,revconv[m]))
104+                 else:
105+                     k.append(m)
106+             l.append(k)
107+         op[revconv[i]] = l
108+
109+     return op
110+
111 result = rem(gram)
112
113
114 def first(gram, term):
115     a = []
116     if term not in gram:
117
118         return [term]
119         for i in gram[term]:
120             if i[0] not in gram:
121                 a.append(i[0])
122             elif i[0] in gram:
123                 a += first(gram, i[0])
124
125     firsts = {}
126     for i in result:
127         firsts[i] = first(gram,i)
128     print(f'First({term}):',firsts[term])
129
130
131 def follow(gram, term):
132     a = []
133     for rule in gram:
134         for i in gram[rule]:
135             if term in i:
136                 temp = i
137                 indx = i.index(term)
138                 if indx+1==len(i):
139                     if i[-1] in firsts:
140                         a+=firsts[i[-1]]
141                     else:
142                         a+=[i[-1]]
143                     else:
144                         a+=[["e"]]
145                     if rule != term and "e" in a:
146                         a+= follow(gram,rule)
147
148     return a
149
150 follows = {}
151 for i in result:
152     follows[i] = list(set(follow(gram,i)))
153     if "e" in follows[i]:
154         follows[i].pop(follows[i].index("e"))
155         follows[i]+=["$"]
156
157 print(f'Follow({term}):',follows[term])
```

Activate Windows

Go to Settings to activate Windows.

Activate Windows

Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

Output:

```
First(E): ['(', 'i']
First(T): ['(', 'i']
First(F): ['(', 'i']
First(E'): ['+', 'e']
First(T'): ['*', 'e']
Follow(E): [')', '$']
Follow(T): [')', '+', '$']
Follow(F): [')', '+', '*', '$']
Follow(E'): [')', '$']
Follow(T'): [')', '+', '$']

...Program finished with exit code 0
Press ENTER to exit console.[]
```

Activate Windows
Go to Settings to activate Windows.

Predictive Parsing

Title: Predictive Parsing Table

Aim: To implement the predictive parsing table.

Algorithm:

LL(1) Parsing: Here the 1st L represents that the scanning of the Input will be done from Left to Right manner and second L shows that in this Parsing technique we are going to use Leftmost Derivation Tree, and finally the 1 represents the number of look ahead, means how many symbols are you going to see when you want to make a decision.

Construction of LL(1) Parsing Table: To construct the Parsing table, we have two functions:

- 1: **First():** If there is a variable, and from that variable if we try to derive all the strings then the beginning Terminal Symbol is called the first.
- 2: **Follow():** What is the Terminal Symbol which follows a variable in the process of derivation.

Now, after computing the First and Follow set for each Non-Terminal symbol we have to construct the Parsing table. In the table Rows will contain the Non-Terminals and the column will contain the Terminal Symbols. All The Null Productions of the Grammars will go under the Follow elements and the remaining productions will lie under the elements of the First set.

RA1911033010047

KOLLA SESHU KIRAN

CODE:

```
1 #RA1911033010047
2 gram = {
3     "E": ["E+T", "T"],
4     "T": ["T+F", "F"],
5     "F": ["(E)", "i"]
6 }
7
8 def removeDirectLR(gramA, A):
9     """gramA is dictionary"""
10    temp = gramA[A]
11    tempCr = []
12    tempInCr = []
13    for i in temp:
14        if i[0] == A:
15            #tempInCr.append(i[1:])
16            tempInCr.append(i[1:]+[A+"'"])
17        else:
18            #tempCr.append(i)
19            tempCr.append(i+[A+"'"])
20    tempInCr.append(["e"])
21    gramA[A] = tempCr
22    gramA[A+"'"] = tempInCr
23    return gramA
24
25
26 def checkForIndirect(gramA, a, ai):
27     if ai not in gramA:
28         return False
29     if a == ai:
30         return True
31     for i in gramA[ai]:
32         if i[0] == ai:
33             return False
34         if i[0] in gramA:
35             return checkForIndirect(gramA, a, i[0])
36     return False
37
38 def rep(gramA, A):
39     temp = gramA[A]
40     newTemp = []
41     for i in temp:
42         if checkForIndirect(gramA, A, i[0]):
43             t = []
44             for k in gramA[i[0]]:
45                 t+=k
46                 t+=i[1:]
47                 newTemp.append(t)
48
49         else:
50             newTemp.append(i)
51     gramA[A] = newTemp
52     return gramA
53
54
55 def rem(gram):
56     c = 1
57     conv = {}
58     gramA = {}
59     revconv = {}
60     for j in gram:
61         conv[j] = "A"+str(c)
62         gramA["A"+str(c)] = []
63         c+=1
64
65     for i in gram:
66         for j in gram[i]:
67             temp = []
68             for k in j:
69                 if k in conv:
70                     temp.append(conv[k])
71                 else:
72                     temp.append(k)
73             gramA[conv[i]].append(temp)
74
75
76 #print(gramA)
```

Activate Windows
Go to Settings to activate Windows.

Activate Windows
Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

```
77-     for i in range(c-1,0,-1):
78-         ai = "A"+str(i)
79-         for j in range(0,i):
80-             aj = gramA[ai][0][0]
81-             if ai==aj :
82-                 if aj in gramA and checkForIndirect(gramA,ai,aj):
83-                     gramA = rep(gramA, ai)
84-
85-     for i in range(1,c):
86-         ai = "A"+str(i)
87-         for j in gramA[ai]:
88-             if ai==j[0]:
89-                 gramA = removeDirectLR(gramA, ai)
90-                 break
91-
92-     op = {}
93-     for i in gramA:
94-         a = str(i)
95-         for j in conv:
96-             a = a.replace(conv[j],j)
97-         revconv[i] = a
98-
99-     for i in gramA:
100-         l = []
101-         for j in gramA[i]:
102-             k = []
103-             for m in j:
104-                 if m in revconv:
105-                     k.append(m.replace(m,revconv[m]))
106-                 else:
107-                     k.append(m)
108-             l.append(k)
109-         op[revconv[i]] = l
110-
111-     return op
112-
113 result = rem(gram)
114 terminals = []
115-
116-     for i in result:
117-         for j in result[i]:
118-             for k in j:
119-                 if k not in result:
120-                     terminals+=[k]
121 #print(terminals)
122-
123 def first(gram, term):
124     a = []
125     if term not in gram:
126         return [term]
127     for i in gram[term]:
128         if i[0] not in gram:
129             a.append(i[0])
130         elif i[0] in gram:
131             a += first(gram, i[0])
132     return a
133-
134 firsts = {}
135 for i in result:
136     firsts[i] = first(result,i)
137 # print(f'First({i}):',firsts[i])
138-
139 def follow(gram, term):
140     a = []
141     for rule in gram:
142         for i in gram[rule]:
143             if term in i:
144                 temp = i
145                 indx = i.index(term)
146                 if indx+1==len(i):
147                     if i[-1] in firsts:
148                         a+=firsts[i[-1]]
149                     else:
150                         a+=[i[-1]]
151                 else:
152                     a+=[“e”]
```

Activate Windows

Go to Settings to activate Windows.

Activate Windows

Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

```
153-         if rule != term and "e" in a:
154-             a+= follow(gram,rule)
155-     return a
156-
157- follows = {}
158- for i in result:
159-     follows[i] = list(set(follow(result,i)))
160-     if "e" in follows[i]:
161-         follows[i].pop(follows[i].index("e"))
162-     follows[i]+=["$"]
163- # print(f'Follow({i}):',follows[i])
164-
165- resMod = {}
166- for i in result:
167-     l = []
168-     for j in result[i]:
169-         temp = ""
170-         for k in j:
171-             temp+=k
172-         l.append(temp)
173-     resMod[i] = l
174-
175- # create predictive parsing table
176- tterm = list(terminals)
177- tterm.pop(tterm.index("e"))
178- tterm=[ "$"]
179- pptable = {}
180- for i in result:
181-     for j in tterm:
182-         if j in firsts[i]:
183-             pptable[(i,j)]=resMod[i[0]][0]
184-         else:
185-             pptable[(i,j)]=""
186-     if "e" in firsts[i]:
187-         for j in tterm:
188-             if j in follows[i]:
189-                 pptable[(i,j)]="e"
190- pptable[("F","i")]= "i"
191- toprint = f'"": <10'
192- for i in tterm:
193-     toprint+= f'|{i}: <10'
194- print(tprint)
195- for i in result:
196-     toprint = f'{i}: <10'
197-     for j in tterm:
198-         if pptable[(i,j)]!="":
199-             toprint+=f'|{i}->"'+pptable[(i,j)]: <10'
200-         else:
201-             toprint+=f'|{pptable[(i,j)]: <10}'
202-     print(f'-:{<76}')
203-     print(tprint)
```

Activate Windows
Go to Settings to activate Windows.

Activate Windows
Go to Settings to activate Windows.

Output:

	i	+	*)	(\$
E	E->TE'				E->TE'	
T	T->FT'				T->FT'	
F	F->i				F->(E)	
E'		E'->TE'		E'->e		E'->e
T'		T'->e	T'->FT'	T'->e		T'->e

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

Result : The program to find the first and follow function of a given grammar and the predictive parsing table is successfully implemented and output is obtained.

Compiler Design

* UNIT - II *

* 22/2/22 *

Assignment - 2

* PREDICTIVE PARSING TABLE *

"cse - se - M₂"

- (Q) Seetha is applying for driving license. She checks for the eligibility in RTO office. It says the first mandatory eligibility is age. The age (A) must be from 18 to 59. The optional eligibility is educational qualification (Q). It can be undergraduate (ug), post graduate (pg), diploma (dip). Define context free grammar (CFG) for the given scenario. Check whether the person who holds a post graduate degree but 61 years is eligible to get the driving license using predictive parsing.

Solution

The context free grammar for the above given scenario is

$$L \rightarrow A Q | A$$

$$Q \rightarrow ug | pg | dip$$

$$A \rightarrow 1z | 4x$$

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

$$z \rightarrow 8|9$$

$$x \rightarrow 0|1|4|6|7|z$$

Step 1:-

Elimination of left factoring of production,
since there is no left recursion, we will proceed
with the left factoring.

$$L \rightarrow AL' \quad z \rightarrow 8$$

$$L' \rightarrow Q | \epsilon \quad z \rightarrow 9$$

$$Q \rightarrow uq \quad x \rightarrow 0$$

$$Q \rightarrow pq \quad x \rightarrow 1$$

$$Q \rightarrow d\epsilon p \quad x \rightarrow 4$$

$$Q \rightarrow d\epsilon p \quad x \rightarrow 6$$

$$A \rightarrow 1z \quad x \rightarrow 7$$

$$A \rightarrow Yx \quad x \rightarrow 2$$

$$Y \rightarrow z$$

$$Y \rightarrow 3$$

$$Y \rightarrow 4$$

$$Y \rightarrow 5$$

Step 2:-

We have to compute the first and follow
for the grammar.

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

$$\text{First}(L') = \{\epsilon, uq, pq, d\epsilon p\}$$

$$\text{First}(Q) = \{uq, pq, d\epsilon p\}$$

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

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

$$\text{First}(z) = \{8, 9\}$$

$$\text{First}(x) = \{0, 1, 2, \dots, 9\}$$

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

$\Rightarrow \text{Follow}(z) \supset \text{Follow}(x) \supset \text{Follow}(y)$

$$\Rightarrow \{ \$, 0, 1, 2, 3, 4, \dots, 9 \}$$

	0	1	2	3	4	5	6	7	8	9	uq	pq	dsp	\$
L	$L \rightarrow AL'$													
L'												$L' \rightarrow Q$	$L' \rightarrow Q$	$L' \rightarrow Q$
Q												$Q \rightarrow uq$	$Q \rightarrow pq$	$Q \rightarrow dsp$
A	$A \rightarrow 12 4x$													
y		$y \rightarrow 2$	$y \rightarrow 3$	$y \rightarrow 4$	$y \rightarrow 5$									
z								$z \rightarrow 8$	$z \rightarrow 9$					
x		$x \rightarrow 0$	$x \rightarrow 1$	$x \rightarrow 4$	$x \rightarrow 4$	$x \rightarrow 6$	$x \rightarrow 7$	$x \rightarrow 2$	$x \rightarrow 2$					

* Construction of predictive parsing *

Now, we will check whether the input string will parse or not therefore.

stack	input	Action
\$L	61pq\$	error

*empty cell represents error.

→ Error because we can't (1, b) from the above drawn predictive parsing table.
Therefore, the given input string will not parse.

* Assignment - 03 *

RA1911033010047

Date:- 22/2/22

Kolla. Sehukiran

* SLR PARSER *

a)

Engineering - E

knowledge - ed.

Technology - T

Fundamental - F

The grammar is

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid ed.$$

Step 1:- Find First and Follow for all the non-terminals.

First.

$$\text{First}(E) = \{G, ed\}$$

$$\text{First}(T) = \{G, ed\}$$

$$\text{First}(F) = \{G, ed\}$$

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

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

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

Step 2:- Number all the productions.

$$1. E \rightarrow E + T$$

$$2. E \rightarrow T$$

$$3. T \rightarrow T * F$$

$$4. T \rightarrow F$$

5. $F \rightarrow (E)$

6. $F \rightarrow id$

Step 3:- Write the Augmented Grammar:

$E' \rightarrow E$

$E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow :F$

$F \rightarrow (E)$

$F \rightarrow id$

Step 4:- construct the canonical collection of sets of LR(0) items.

closure of $\{[E' \rightarrow \cdot E]\}$

$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$ $\xrightarrow{f_0}$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow \cdot id$

goto (I_0, E) \leftarrow closure $(\{[E \rightarrow E \cdot], [E \rightarrow E \cdot + T]\})$

$E \rightarrow E \cdot$

$E \rightarrow E \cdot + T$

goto (I_0, T) \leftarrow closure $(\{T \rightarrow T \cdot\}, [T \rightarrow T \cdot * F])$

$E \rightarrow T \cdot$

$T \rightarrow T \cdot * F$

goto ($I_0, +$) \leftarrow closure ($\{[T \rightarrow F]?\}$)

$$T \rightarrow F \} I_3$$

goto (I_0, c) \leftarrow closure ($\{[F \rightarrow (\cdot E)]?\}$)

$$\begin{aligned} & F \rightarrow (\cdot E) \\ & E \rightarrow \cdot E + T \\ & E \rightarrow \cdot T \\ & T \rightarrow \cdot T * F \\ & T \rightarrow \cdot F \\ & F \rightarrow \cdot (E) \\ & F \rightarrow \cdot \text{id} \end{aligned} \quad \left\} I_4$$

goto (I_0, id) \leftarrow closure ($\{[F \rightarrow \text{id}]?\}$)

$$F \rightarrow \text{id} \} I_5$$

goto ($I_1, +$) \leftarrow closure ($\{[E \rightarrow E + \cdot T]\}$)

$$\begin{aligned} & E \rightarrow E + \cdot T \\ & T \rightarrow \cdot T * F \\ & T \rightarrow \cdot F \\ & F \rightarrow \cdot (E) \\ & F \rightarrow \cdot \text{id} \end{aligned} \quad \left\} I_6$$

goto ($I_2, *$) \leftarrow closure ($\{[T \rightarrow T * \cdot F]\}$)

$$\begin{aligned} & T \rightarrow T * \cdot F \\ & F \rightarrow \cdot (E) \\ & F \rightarrow \cdot \text{id} \end{aligned} \quad \left\} I_7$$

goto (I_4, E) \leftarrow closure ($\{[F \rightarrow (E \cdot)], [E \rightarrow E \cdot + T]\}$)

$$\begin{aligned} & F \rightarrow (E \cdot) \\ & E \rightarrow E \cdot + T \end{aligned} \quad \left\} I_8$$

goto(I_4, T) - closure($\{[E \rightarrow T] [T \rightarrow T \cdot * F]\}$)

$$\left. \begin{array}{l} E \rightarrow T \\ T \rightarrow T \cdot * F \end{array} \right\} I_2$$

goto(I_4, F) - closure($\{[T \rightarrow F]\}$)

$$T \rightarrow F \} I_3$$

goto(I_4, C) - closure($\{[F \rightarrow (\cdot E)]\}$)

$$\left. \begin{array}{l} F \rightarrow (\cdot E) \\ F \rightarrow \cdot E + T \\ F \rightarrow \cdot T \\ T \rightarrow \cdot T * F \\ T \rightarrow \cdot F \\ F \rightarrow \cdot (E) \\ F \rightarrow \cdot \text{id} \end{array} \right\} I_4$$

goto(I_4, id) - closure($\{[F \rightarrow \text{id} \cdot]\}$)

$$F \rightarrow \text{id} \cdot \} I_5$$

goto(I_6, T) - closure($\{[E \rightarrow E + T] [T \rightarrow T \cdot * F]\}$)

$$\left. \begin{array}{l} E \rightarrow E + T \\ T \rightarrow T \cdot * F \end{array} \right\} I_6$$

goto(I_6, F) - closure($\{[T \rightarrow F]\}$)

$$T \rightarrow F \} I_3$$

goto(I_6, C) - closure($\{[F \rightarrow (\cdot E)]\}$)

$$\left. \begin{array}{l} F \rightarrow (\cdot E) \\ F \rightarrow \cdot E + T \\ F \rightarrow \cdot T \\ T \rightarrow \cdot T * F \end{array} \right\} I_4$$

$$\left. \begin{array}{l} T \rightarrow \cdot F \\ F \rightarrow \cdot (E) \\ F \rightarrow \cdot \text{id} \end{array} \right\} I_4$$

goto(I_6, id) - closure($\{[F \rightarrow \text{id}]\}$)

$$F \rightarrow \text{id} \cdot \left. \right\} I_5$$

goto(I_7, F) - closure($\{[T \rightarrow T * F]\}$)

$$T \rightarrow T * F \cdot \left. \right\} I_{10}$$

goto(I_7, C) - closure($\{[F \rightarrow (\cdot E)]\}$)

$$\left. \begin{array}{l} F \rightarrow (\cdot E) \\ E \rightarrow \cdot E + T \\ E \rightarrow \cdot T \\ T \rightarrow \cdot T * F \\ T \rightarrow \cdot F \\ F \rightarrow \cdot (E) \\ F \rightarrow \cdot \text{id} \end{array} \right\} I_4$$

goto(I_7, id) - closure($\{[F \rightarrow \text{id}]\}$)

$$F \rightarrow \text{id} \cdot \left. \right\} I_5$$

goto($I_8,)$) - closure($\{[F \rightarrow (\cdot E)]\}$)

$$F \rightarrow (\cdot E) \cdot \left. \right\} I_{11}$$

goto($I_8, +$) - closure($\{[F \rightarrow E + \cdot T]\}$)

$$E \rightarrow E + \cdot T$$

$$T \rightarrow \cdot T * F$$

$$T \rightarrow \cdot F$$

$$F \rightarrow \cdot (E)$$

$$F \rightarrow \cdot \text{id} \left. \right\} I_6$$

goto ($I_q, *$) - closure $\left(\{ [T \rightarrow T^* \cdot F] \} \right)$

$$\begin{array}{c} T \rightarrow T^* \cdot F \\ F \rightarrow \cdot (F) \\ F \rightarrow \cdot \rho d \end{array} \left. \begin{array}{c} \\ \\ \end{array} \right\} I_7$$

state	Action						GOTO		
	ρd	+	*	()	\$	E	T	F
0	s_5			s_q			1	2	3
1		s_6				acc			
2		r_2	s_7		r_2	r_2			
3		r_4	r_4		r_4	r_4			
4	s_5			s_4			8	2	3
5		r_6	r_6		r_6	r_6			
6	s_5			s_4				9	3
7	s_5			s_4					10
8		s_6			s_{11}				
9		r_1	s_7		r_1	r_1			
10		r_3	r_3		r_3	r_3			
11		r_5	r_5		r_5	r_5			

Step 1:-

Stack	Input	Action
0	qd + id * id \$	shift
0 id5	+ id * id \$	reduce F → id
0F3	+ id * id \$	reduce T → F
0+2	+ id * id \$	reduce F → T
0F1	+ id * id \$	shift
0E1 + 6	id * id \$	shift
0E1 + 6 id5	* id \$	reduce F → id
0E1 + 6F3	* id \$	reduce T → F
0E1 + 6T9	* id \$	shift
0E1 + 6T9 * 7	id \$	shift
0E1 + 6T9 * 7 id5	\$	reduce F → id.
0E1 + 6T9 * 7 F10	\$	reduce T → T * F
0E1 + 6T9	\$	reduce F → F + T
0F	\$	Accept

EXPERIMENT - 6

Title: Shift Reduce Parsing

Aim: To perform the shift reduce parsing

Algorithm:

Shift Reduce parser attempts for the construction of parse in a similar manner as done in bottom up parsing i.e. the parse tree is constructed from leaves(bottom) to the root(up). A more general form of shift reduce parser is LR parser.

This parser requires some data structures i.e.

1. An input buffer for storing the input string.
2. A stack for storing and accessing the production rules.

Basic Operations –

1. Shift: This involves moving symbols from the input buffer onto the stack.

2. Reduce: If the handle appears on top of the stack then, its reduction by using appropriate production rule is done i.e. RHS of production rule is popped out of stack and LHS of production rule is pushed onto the stack.

3. Accept: If only the start symbol is present in the stack and the input buffer is empty then, the parsing action is called accept. When accepted action is obtained, it means successful parsing is done.

4. Error: This is the situation in which the parser can neither perform shift action nor reduce action and not even accept action.

CODE:

```

1 #RA1911033010047
2 gram = {
3     "E": ["2E2", "3E3", "4"]
4 }
5 starting_terminal = "E"
6 inp = "2324232$"
7 }
8 starting_terminal = "S"
9 inp = "i+i*i"
10 """
11 stack = "$"
12 print(f'{ "Stack": <15>'+ "|"+f'{ "Input Buffer": <15>'+ "|"+f'{ "Parsing Action"}')
13 print(f'{ "-":<50>}')
14
15 while True:
16     action = True
17     i = 0
18     while i<len(gram[starting_terminal]):
19         if gram[starting_terminal][i] in stack:
20             stack = stack.replace(gram[starting_terminal][i],starting_terminal)
21             print(f'{stack: <15>'+ "|"+f'{inp: <15>'+ "|"+f'{Reduce S->{gram[starting_terminal][i]}')
22             i+=1
23             action = False
24         i+=1
25     if len(inp)>1:
26         stack+=inp[0]
27         inp=inp[1:]
28         print(f'{stack: <15>'+ "|"+f'{inp: <15>'+ "|"+f'{Shift}')
29         action = False
30
31     if inp == "$" and stack == ("$"+starting_terminal):
32         print(f'{stack: <15>'+ "|"+f'{inp: <15>'+ "|"+f'{Accepted}')
33         break
34
35     if action:
36         print(f'{stack: <15>'+ "|"+f'{inp: <15>'+ "|"+f'{Rejected}')
37         break

```

Activate Windows
Go to Settings to activate Windows.

Output:

Stack	Input Buffer	Parsing Action
\$2	324232\$	Shift
\$23	24232\$	Shift
\$232	4232\$	Shift
\$2324	232\$	Shift
\$232E	232\$	Reduce S->4
\$232E2	32\$	Shift
\$23E	32\$	Reduce S->2E2
\$23E3	2\$	Shift
\$2E	2\$	Reduce S->3E3
\$2E2	\$	Shift
\$E	\$	Reduce S->2E2
\$E	\$	Accepted
...Program finished with exit code 0		

Result: The program to implement shift reduce parser is successfully implemented.

EXPERIMENT - 7

Title: COMPUTATION OF LEADING AND TRAILING SETS

Aim: To Compute Leading and Trailing Sets for a given grammar.

Algorithm:

1. Leading and Trailing are functions specific to generating an operator-precedence parser, which is only applicable if you have an operator precedence grammar. An operator precedence grammar is a special case of an operator grammar, and an operator grammar has the important property that *no production has two consecutive non-terminals*.

(An operator precedence grammar is, loosely speaking, an operator grammar which can be parsed with an operator precedence parser)

2. Given an operator grammar, the function Leading (resp. Trailing) of a non-terminal produces the set of terminals which could be (recursively) the first (resp. last) terminal in a production for that non-terminal.

3. Another way to think of that a terminal is in the Leading set for a non-terminal is if it is "visible" from the beginning of a production. We consider non-terminals to be "transparent", so a terminal could be visible through a non-terminal or by looking into a visible non-terminal.

Code:

```

1 #RA1911033010047
2 a = ["E=E-T",
3      "E=T",
4      "T=T-F",
5      "T=F",
6      "F=(E)",
7      "F=i"]
8
9 rules = {}
10 terms = []
11 for i in a:
12     temp = i.split("=")
13     terms.append(temp[0])
14     try:
15         rules[temp[0]] += [temp[1]]
16     except:
17         rules[temp[0]] = [temp[1]]
18
19 terms = list(set(terms))
20 print(rules,terms)
21
22 def leading(gram, rules, term, start):
23     s = []
24     if gram[0] not in terms:
25         return gram[0]
26     elif len(gram) == 1:
27         return [0]
28     elif gram[1] not in terms and gram[-1] is not start:
29         for i in rules[gram[-1]]:
30             s+= leading(i, rules, gram[-1], start)
31         s+= [gram[1]]
32     return s
33
34 def trailing(gram, rules, term, start):
35     s = []
36     if gram[-1] not in terms:
37         return gram[-1]
38     elif len(gram) == 1:
39         return [0]
40     elif gram[-2] not in terms and gram[-1] is not start:
41         for i in rules[gram[-1]]:
42             s+= trailing(i, rules, gram[-1], start)
43             s+= [gram[-2]]
44     return s
45
46 leads = {}
47 trails = {}
48 for i in terms:
49     s = [0]
50     for j in rules[i]:
51         s+=leading(j,rules,i,i)
52     s = set(s)
53     s.remove(0)
54     leads[i] = s
55     s = [0]
56     for j in rules[i]:
57         s+=trailing(j,rules,i,i)
58     s = set(s)
59     s.remove(0)
60     trails[i] = s
61
62 for i in terms:
63     print("LEADING("+i+":" ,leads[i])
64 for i in terms:
65     print("TRAILING("+i+":" ,trails[i])
```

Activate Windows

Output:

```
input
{'E': ['E*T', 'T'], 'T': ['T*F', 'F'], 'F': ['(E)', 'i']} ('T', 'E', 'F')
LEADING(T): {'i', '*', '('}
LEADING(E): {'(', '+', '*', 'i'}
LEADING(F): {'i', ')'}
TRAILING(T): {'i', '*', ')'}
TRAILING(E): {'*', '+', ')', 'i'}
TRAILING(F): {'i', ')'}

...Program finished with exit code 0
Press ENTER to exit console.[]
```

Result: The Program to compute the leading and trailing sets of a given grammar is successfully implemented.

EXPERIMENT - 8

Title: COMPUTATION OF LR(0) ITEMS

Aim: To compute the lr(0) items for a given grammar.

Algorithm:

- The LR Parser is a Shift-reduce Parser that makes use of a Deterministic Finite Automata, recognizing the Set Of All Viable Prefixes by reading the stack from Bottom To Top.
- If a Finite-State Machine that recognizes viable prefixes of the right sentential forms is constructed, it can be used to guide the handle selection in the Shift-reduce Parser.
- Handle: Handle is a substring that matches the body of a production
- Handle is a Right Sentential Form + position where reduction can be performed + production used for reduction.

LR(0) Items

An LR(0) Item of a Grammar G is a Production of G with a Dot (.) at some position of the right side. Production $A \rightarrow XYZ$ yields the Four items:

1. $A \rightarrow \bullet XYZ$ We hope to see a string derivable from XYZ next on the input.
2. $A \rightarrow X \bullet YZ$ We have just seen on the input a string derivable from X and that we hope to see a string derivable from YZ next on the input.
3. $A \rightarrow XY \bullet Z$
4. $A \rightarrow X \bullet YZ$

The production $A \rightarrow \epsilon$ generates only one item, $A \rightarrow \bullet$.

Each of this item is a Viable prefixes

Closure Item : An Item created by the closure operation on a state.

• Complete Item : An Item where the Item Dot is at the end of the

RHS.

CODE:

```

1 #RA1911033010047
2 gram = {
3     "S":["CC"],
4     "C":["aC","d"]
5 }
6 start = "S"
7 terms = ["a","d","$"]
8
9 non_terms = []
10 for i in gram:
11     non_terms.append(i)
12 gram["S"]=[start]
13
14
15 new_row = {}
16 for i in terms+non_terms:
17     new_row[i]=""
18
19
20 non_terms += ["S"]
21 # each row in state table will be dictionary {nonterms ,term,$}
22 stateTable = []
23 # I = [(terminal, closure)]
24 # I = [("S","A,A")]
25
26 def Closure(term, I):
27     if term in non_terms:
28         for i in gram[term]:
29             I+=[(term,"." + i)]
30     I = list(set(I))
31     for i in I:
32         # print("." != i[1][-1],i[1][i[1].index(".") + 1])
33         if "." != i[1][-1] and i[1][i[1].index(".") + 1] in non_terms and i[1][i[1].index(".") + 1] != term:
34             I += Closure(i[1][i[1].index(".") + 1], [])
35     return I

```

Activate Windows

```

36 Is = []
37 Is+=set(Closure("S", []))
38
39
40
41 countI = 0
42 omegaList = [set(Is)]
43 while countI<len(omegaList):
44     newrow = dict(new_row)
45     vars_in_I = []
46     Is = omegaList[countI]
47     countI+=1
48     for i in Is:
49         if i[1][-1]!=".":
50             idx = i[1].index(".")
51             vars_in_I+=[i[1][idx+1]]
52     vars_in_I = list(set(vars_in_I))
53     # print(vars_in_I)
54     for i in vars_in_I:
55         In = []
56         for j in Is:
57             if "." + i in j[1]:
58                 rep = j[1].replace("." + i, i + ".")
59                 In+=[(j[0],rep)]
60             if (In[0][1][-1]) == ".":
61                 temp = set(Closure(i,In))
62                 if temp not in omegaList:
63                     omegaList.append(temp)
64                 if i in non_terms:
65                     newrow[i] = str(omegaList.index(temp))
66                 else:
67                     newrow[i] = "s"+str(omegaList.index(temp))
68                 print(f'Goto({countI-1},{i}):{temp} That is {omegaList.index(temp)})')
69             else:
70                 temp = set(In)

```

RA1911033010047

KOLLA SESHU KIRAN

```
71-     if temp not in omegaList:
72-         omegaList.append(temp)
73-     if i in non_terms:
74-         newrow[i] = str(omegaList.index(temp))
75-     else:
76-         newrow[i] = "s"+str(omegaList.index(temp))
77-     print(f'Goto(I{countI-1},{i}):{temp} That is I{omegaList.index(temp)}')
78-
79     stateTable.append(newrow)
80 print("\n\nList of I's\n")
81- for i in omegaList:
82-     print(f'I{omegaList.index(i)}: {i}')
83-
84-
85 #populate replace elements in state Table
86 I0 = []
87- for i in list(omegaList[0]):
88-     I0 += [i[1].replace(".", "")]
89 print(I0)
90-
91- for i in omegaList:
92-     for j in i:
93-         if "." in j[1][-1]:
94-             if j[1][-2]=="S":
95-                 stateTable[omegaList.index(i)]["$"] = "Accept"
96-                 break
97-             for k in terms:
98-                 stateTable[omegaList.index(i)][k] = "r"+str(I0.index(j[1].replace(".", "")))
99 print("\nStateTable")
100-
101 print(f" ":{<9},end="")
102- for i in new_row:
103-     print(f'|{i}:{<11}',end="")
104-
105 print(f'\n{"-":<66}')
```

Activate Windows

```
106- for i in stateTable:
107-     print(f'I{"+str(stateTable.index(i))}': <9>,end="")
108-     for j in i:
109-         print(f'|{i[j]}:<10>',end=" ")
110-     print()
```

Activate Windows

Output:

```
Goto(I0,d):('C', 'd.') That is I1
Goto(I0,c):('C', '.ac'), ('S', 'c.C'), ('C', '.d') That is I2
Goto(I0,S):('S', 'S.') That is I3
Goto(I0,a):('C', '.ac'), ('C', 'a.C'), ('C', '.d') That is I4
Goto(I2,d):('C', 'd.') That is I1
Goto(I2,c):('S', 'cc.') That is I5
Goto(I2,a):('C', '.aC'), ('C', 'a.C'), ('C', '.d') That is I4
Goto(I4,d):('C', 'd.') That is I1
Goto(I4,c):('C', 'aC.') That is I6
Goto(I4,a):('C', '.aC'), ('C', 'a.C'), ('C', '.d') That is I4
```

List of I's

```
I0: (('C', '.ac'), ('S', 'S.'), ('S', 'cc'), ('C', '.d'))
I1: (('C', 'd.'))
I2: (('C', '.ac'), ('S', 'c.C'), ('C', '.d'))
I3: (('S', 'S.'))
I4: (('C', '.ac'), ('C', 'a.C'), ('C', '.d'))
I5: (('S', 'cc.'))
I6: (('C', 'aC.'))
['ac', 'S', 'cc', 'd']
```

State	a	d	S	s	C
I(0)	s4	s1		3	2
I(1)	r3	r3	r3		
I(2)	s4	s1			5
I(3)			Accept		
I(4)	s4	s1			6
I(5)	r2	r2	r2		
I(6)	r0	r0	r0		

Result: The program to compute the Ir(0) items for a given grammar is successfully implemented.

EXPERIMENT - 9

Title: Intermediate code generation – Postfix, Prefix.

Aim: To generate the intermediate code and postfix, prefix.

Algorithm:

In the analysis-synthesis model of a compiler, the front end of a compiler translates a source program into an independent intermediate code, then the back end of the compiler uses this intermediate code to generate the target code (which can be understood by the machine).

The benefits of using machine independent intermediate code are:

- Because of the machine independent intermediate code, portability will be enhanced. For example, if a compiler translates the source language to its target machine language without having the option for generating intermediate code, then for each new machine, a full native compiler is required. Because, obviously, there were some modifications in the compiler itself according to the machine specifications.
- Retargeting is facilitated
- It is easier to apply source code modification to improve the performance of source code by optimizing the intermediate code.

Postfix Notation –

The ordinary (infix) way of writing the sum of a and b is with operator in the middle : a + b

The postfix notation for the same expression places the operator at the right end as ab +. In general, if e1 and e2 are any postfix expressions, and

+ is any binary operator, the result of applying + to the values denoted by e1 and e2 is postfix notation by e1e2 +. No parentheses are needed in postfix notation because the position and arity (number of arguments) of the operators permit only one way to decode a postfix expression. In postfix notation the operator follows the operand.

If we generate machine code directly from source code then for target machine we will have n optimizers and n code generators but if we will have a machine independent intermediate code, we will have only one optimizer. Intermediate code can be either language specific (e.g., Bytecode for Java) or language independent (three-address code).

Example – The postfix representation of the expression $(a - b)^* (c + d) + (a - b)$ is : ab – cd + *ab -+.

2. Three-Address Code –

A statement involving no more than three references(two for operands and one for result) is known as three address statements. A sequence of three address statements is known as three address code. Three address statements are of the form $x = y \text{ op } z$, here x, y, z will have address (memory location). Sometimes a statement might contain less than three references but it is still called three address statements.

Example – The three address code for the expression $a + b^* c + d$:

$T1 = b^* c$

$T2 = a + T1$

$T3 = T2 + d$

T 1 , T 2 , T 3 are temporary variables.

3. Syntax Tree –

Syntax tree is nothing more than a condensed form of a parse tree. The operator and keyword nodes of the parse tree are moved to their parents and a chain of single productions is replaced by a single link in the syntax tree; the internal nodes are operators and child nodes are operands. To form a syntax tree put parentheses in the expression, this way it's easy to recognize which operand should come first.

Code:

```

1 #RA1911033010047
2 OPERATORS = set(['+', '-', '*', '/', '(', ')'])
3 PRI = {'+':1, '-':1, '*':2, '/':2}
4
5 ### INFIX ==> POSTFIX ###
6 def infix_to_postfix(formula):
7     stack = [] # only pop when the coming op has priority
8     output = ''
9     for ch in formula:
10        if ch not in OPERATORS:
11            output += ch
12        elif ch == '(':
13            stack.append('(')
14        elif ch == ')':
15            while stack and stack[-1] != '(':
16                output += stack.pop()
17            stack.pop() # pop '('
18        else:
19            while stack and stack[-1] != '(' and PRI[ch] <= PRI[stack[-1]]:
20                output += stack.pop()
21            stack.append(ch)
22    # leftover
23    while stack:
24        output += stack.pop()
25    print(f'POSTFIX: {output}')
26    return output
27
28 ### INFIX ==> PREFIX ###
29 def infix_to_prefix(formula):
30     op_stack = []
31     exp_stack = []
32     for ch in formula:
33        if not ch in OPERATORS:
34            exp_stack.append(ch)
35        elif ch == '(':
36            op_stack.append(ch)
37        elif ch == ')':
38            while op_stack[-1] != '(':
39                op = op_stack.pop()
40                a = exp_stack.pop()
41                b = exp_stack.pop()
42                exp_stack.append( op+b+a )
43            op_stack.pop() # pop '('
44        else:
45            while op_stack and op_stack[-1] != '(' and PRI[ch] <= PRI[op_stack[-1]]:
46                op = op_stack.pop()
47                a = exp_stack.pop()
48                b = exp_stack.pop()
49                exp_stack.append( op+b+a )
50            op_stack.append(ch)
51
52    # leftover
53    while op_stack:
54        op = op_stack.pop()
55        a = exp_stack.pop()
56        b = exp_stack.pop()
57        exp_stack.append( op+b+a )
58    print(f'PREFIX: {exp_stack[-1]}')
59    return exp_stack[-1]
60
61 ### THREE ADDRESS CODE GENERATION ###
62 def generate3AC(pos):
63     print("### THREE ADDRESS CODE GENERATION ###")
64     exp_stack = []
65     t = 1
66
67     for i in pos:
68        if i not in OPERATORS:
69            exp_stack.append(i)
70        else:
71            print(f't{t} := {exp_stack[-2]} {i} {exp_stack[-1]}')
72            exp_stack=exp_stack[:-2]
73            exp_stack.append(f't{t}')
74            t+=1
75
76 expres = input("INPUT THE EXPRESSION: ")

```

Activate Windows
Go to Settings to activate Windows.

Activate Windows
Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

```
77 pre = infix_to_prefix(expr)
78 pos = infix_to_postfix(expr)
79 generate3AC(pos)
```

Go to Settings to activate Windows.

Output:

```
INPUT THE EXPRESSION: (a+b)*(a-b)
PREFIX: *+ab-ab
POSTFIX: ab+ab-
### THREE ADDRESS CODE GENERATION ####
t1 := a + b
t2 := a - b
t3 := t1 * t2

...Program finished with exit code 0
Press ENTER to exit console.[]
```

Activate Windows
Go to Settings to activate Windows.

Result: The program for intermediate code generation and postfix, prefix is successfully implemented.

EXPERIMENT - 10

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

Aim: To implement 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 is 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 has no next uses or does 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 registers will no longer contain y or z.

Code:

```
1 #RA1911033010047
2 OPERATORS = set(['+', '-', '*', '/', '(', ')'])
3 PRI = {'+':1, '-':1, '*':2, '/':2}
4
5 ### INFIX ==> POSTFIX ###
6 def infix_to_postfix(formula):
7     stack = [] # only pop when the coming op has priority
8     output = ''
9     for ch in formula:
10         if ch not in OPERATORS:
11             output += ch
12         elif ch == '(':
13             stack.append('(')
14         elif ch == ')':
15             while stack and stack[-1] != '(':
16                 output += stack.pop()
17             stack.pop() # pop '('
18         else:
19             while stack and stack[-1] != '(' and PRI[ch] <= PRI[stack[-1]]:
20                 output += stack.pop()
21             stack.append(ch)
22     # leftover
23     while stack:
24         output += stack.pop()
25     print(f'POSTFIX: {output}')
26     return output
27
28 ### INFIX ==> PREFIX ###
29 def infix_to_prefix(formula):
30     op_stack = []
31     exp_stack = []
32     for ch in formula:
33         if not ch in OPERATORS:
34             exp_stack.append(ch)
35         elif ch == '(':
36             op_stack.append(ch)
37         elif ch == ')':
38             while op_stack[-1] != '(':
```

Activate Windows
Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

```
39         op = op_stack.pop()
40         a = exp_stack.pop()
41         b = exp_stack.pop()
42         exp_stack.append( op+b+a )
43         op_stack.pop() # pop '('
44     else:
45         while op_stack and op_stack[-1] != '(' and PRI[ch] <= PRI[op_stack[-1]]:
46             op = op_stack.pop()
47             a = exp_stack.pop()
48             b = exp_stack.pop()
49             exp_stack.append( op+b+a )
50         op_stack.append(ch)
51
52     # leftover
53     while op_stack:
54         op = op_stack.pop()
55         a = exp_stack.pop()
56         b = exp_stack.pop()
57         exp_stack.append( op+b+a )
58     print(f'PREFIX: {exp_stack[-1]}')
59     return exp_stack[-1]
60
61 ### THREE ADDRESS CODE GENERATION ###
62 def generate3AC(pos):
63     print("### THREE ADDRESS CODE GENERATION ###")
64     exp_stack = []
65     t = 1
66
67     for i in pos:
68         if i not in OPERATORS:
69             exp_stack.append(i)
70         else:
71             print(f't{t} := {exp_stack[-2]} {i} {exp_stack[-1]}')
72             exp_stack=exp_stack[:-2]
73             exp_stack.append(f't{t}')
74             t+=1
75
76 expres = input("INPUT THE EXPRESSION: ")
```

```
77 pre = infix_to_prefix(expres)
78 pos = infix_to_postfix(expres)
79 generate3AC(pos)
80 def Quadruple(pos):
81     stack = []
82     op = []
83     x = 1
84     for i in pos:
85         if i not in OPERATORS:
86             stack.append(i)
87         elif i == '+':
88             op1 = stack.pop()
89             stack.append("t(%s)" %x)
90             print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format(i,op1,"(-)", " t(%s)" %x))
91             x = x+1
92         if stack != []:
93             op2 = stack.pop()
94             op1 = stack.pop()
95             print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format("+",op1,op2, " t(%s)" %x))
96             stack.append("t(%s)" %x)
97             x = x+1
98         elif i == '-':
99             op2 = stack.pop()
100            op1 = stack.pop()
101            print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format(i,op2,"(-)",op1))
102        else:
103            op1 = stack.pop()
104            op2 = stack.pop()
105            print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format(i,op2,op1, " t(%s)" %x))
106            stack.append("t(%s)" %x)
107            x = x+1
108 print("The quadruple for the expression ")
109 print(" OP | ARG 1 |ARG 2 |RESULT  ")
110 Quadruple(pos)
111
112 def Triple(pos):
113     stack = []
114     op = []
```

Activate Windows
Go to Settings to activate Windows.

Activate W
Go to Settings

RA1911033010047

KOLLA SESHU KIRAN

```
115     x = 0
116     for i in pos:
117         if i not in OPERATORS:
118             stack.append(i)
119         elif i == '-':
120             op1 = stack.pop()
121             stack.append("(%s)" %x)
122             print("{0:^4s} | {1:^4s} | {2:^4s}".format(i,op1,"(-"))
123             x = x+1
124         if stack != []:
125             op2 = stack.pop()
126             op1 = stack.pop()
127             print("{0:^4s} | {1:^4s} | {2:^4s}".format("+",op1,op2))
128             stack.append("(%s)" %x)
129             x = x+1
130         elif i == '=':
131             op2 = stack.pop()
132             op1 = stack.pop()
133             print("{0:^4s} | {1:^4s} | {2:^4s}".format(i,op1,op2))
134     else:
135         op1 = stack.pop()
136         if stack != []:
137             op2 = stack.pop()
138             print("{0:^4s} | {1:^4s} | {2:^4s}".format(i,op2,op1))
139             stack.append("(%s)" %x)
140             x = x+1
141     print("The triple for given expression")
142     print(" OP | ARG 1 |ARG 2 ")
143 Triple(pos)
```

Activate Windows
Go to Settings to activate Windows.

Output:

```
INPUT THE EXPRESSION: (a+b)*(a-b)
PREFIX: *+ab-ab-
POSTFIX: ab+ab-
### THREE ADDRESS CODE GENERATION ###
t1 := a + b
t2 := a - b
t3 := t1 * t2
The quadruple for the expression
OP | ARG 1 |ARG 2 |RESULT
+ | a | b | t(1)
- | b | (-) | t(2)
+ | a | t(2)| t(3)
* | t(1) | t(3)| t(4)
The triple for given expression
OP | ARG 1 |ARG 2
+ | a | b
- | b | (-)
+ | a | (1)
* | (0) | (2)

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

Result: The program was successfully implemented.

* Compiler Design *

Assignment 4

Date - 31/3/22

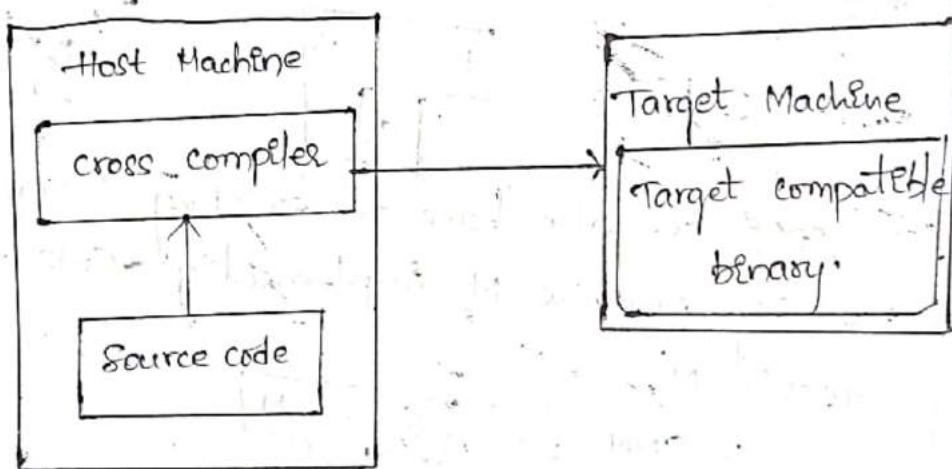
Cross Compiler:-

Cross compiler is also known as "retargetable compiler".
 GNU GCC is an example for cross compiler.

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

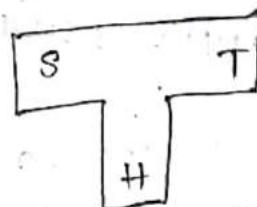
- For example, A compiler that runs on windows platform also generates a code that runs on Linux platform is a cross compiler.
- The process of creating executable code for different machines is also called "retargeting".

* cross compiler operation * :



- A host language is a language in which the compiler is written.

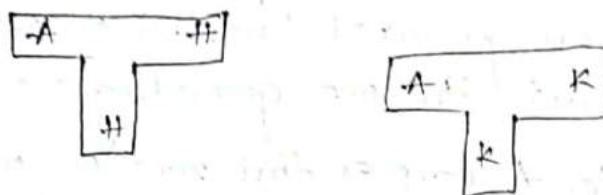
→ T-diagram.



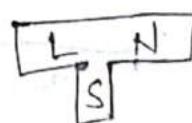
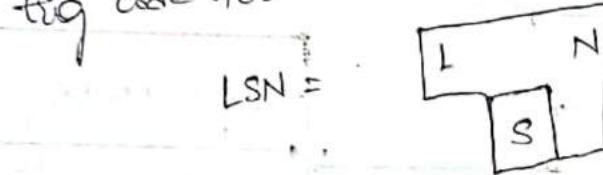
- Cross compilers are used very often in practice.

T compiler

- if we want a compiler from language A to language B on a machine with language F
 - write one with F.
- construct a compiler between a source and a target language using one host language from another host language.



- if we have to implement, from scratch, a compiler from a high-level language A to a machine which is also a host language.
- suppose we have cross compilers for a new language L in implementation languages genera ting code for machine N.

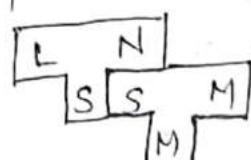


- suppose we also have an existing S compiler running machine M implementing code for machine M.



Now, run LSN through SMM to produce LMN

$$LMN = LSN + SMM$$



The first T describes a compiler from L to N written in S. Second T → S to M written in M.

RA1911033010047

KOLLA SESHU KIRAN

HACKERRANK REGEX SOLUTIONS(21 - 31)

Date: 31-03-2022

The screenshot shows a HackerRank challenge interface for a problem titled "Forward References".

Header: The top navigation bar includes links for "PREPARE", "CERTIFY", and "COMPETE". On the right, there's a search bar, a profile icon for "RA1911033010047", and a notification bell.

Breadcrumbs: The path "Prepare > Regex > Backreferences > Forward References" is displayed.

User Info: Points: 710 Rank: 1

Problem Overview: A message says "You made this submission 5 months ago." and shows a score of "Score: 20.00 Status: Accepted".

Submitted Code: The code is written in PHP and defines a variable \$Regex_Pattern with the value '/^(\\2tic|(tac))+\$/'. A note below the code says "Replace _____ with your regex."

Test Cases: The test cases section shows the following results:

- Test case 0: Compiler Message (Success)
- Test case 1: Success
- Test case 2: Input (stdin) tactactic, Download
- Test case 3: tactactic
- Test case 4: Expected Output (true), Download
- Test case 5: true

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE

Search Profile RA1911033010047

Prepare > Regex > Assertions > Positive Lookahead

Positive Lookahead ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

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

Submitted Code

Language: PHP Open in editor

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

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input(stdin) Download
1 gooooo!

Test case 3

Expected Output Download
1 Number of matches : 3

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search RA1911033010047

Prepare > Regex > Assertions > Negative Lookahead

Negative Lookahead ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.

Score: 20.00 Status: Accepted

Submitted Code

Language: PHP [Open in editor](#)

```
2
3
4 $Regex_Pattern = '/(\$)(?!\\1)\/'; //Do not delete '/'. Replace
----- with your regex.
5
6
```

NEED HELP?

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

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input (stdin) Download

```
1 gooooo
```

Test case 3

Expected Output Download

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

RA1911033010047

KOLLA SESHU KIRAN

The screenshot shows a challenge titled "Positive Lookbehind" on the HackerRank platform. The top navigation bar includes links for PREPARE, CERTIFY, and COMPETE, along with a search bar and user profile information (RA1911033010047). The challenge title is "Positive Lookbehind" with a difficulty rating of ★.

Below the title, it says "You made this submission 5 months ago." and "Score: 20.00 Status: Accepted".

The "Submitted Code" section shows the following PHP code:

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

The "Test cases" section displays the following results:

Test case	Compiler Message	Status
Test case 0	Compiler Message	Success
Test case 1	Input(stdin)	Success
Test case 2	Input(stdin)	Success
Test case 3	Input(stdin)	Success

For Test case 2, the input was "123Go!" and the output was "Number of matches : 1".

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search Profile RA1911033010047

Prepare > Regex > Assertions > Negative Lookbehind

Negative Lookbehind ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 5 months ago.

Score: 20.00 Status: Accepted

Submitted Code

Language: PHP Open in editor

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

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 1ols
```

Test case 3 Expected Output Download

```
1 Number of matches : 3
```

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search Search Notifications Notifications RA1911033010047

Prepare > Regex > Applications > Detect HTML links

Detect HTML links ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission 3 months ago.

Score: 10.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 import re
2 n = int(input())
3 l = []
4 for i in range(n):
5     x = str(input())
6     l.append(x)
7 pattern = re.compile('<a href="(.+?)".*?>(.+?)</a>')
8 for x in l:
9     a = re.findall(pattern, x)
10    for z in a:
11        y1 = z[0].strip()
12        y2 = z[1].strip()
13        p2 = re.compile('(<.*>)*(.+?)<.*>')
14        y2 = re.findall(p2, y2)
15        if(len(y2) != 0):
16            y2 = str(y2)
17            y2 = y2[2:len(y2)-2]
18            zz = y2.find('>')
19            y2 = y2[zz+1 : len(y2)]
20            print(y1+', '+y2)
21        else:
22            print(y1+', '+z[1].strip())
23
```

Test case 0

Input (stdin) Download

```
1 2
2 <p><a href="http://www.quackit.com/html/tutorial/html_links.cfm">Example Link</a></p>
3 <div class="more-info"><a href="http://www.quackit.com/html/examples/html_links_examples.cfm">More Link Examples...</a></div>
```

Expected Output Download

```
1 http://www.quackit.com/html/tutorial/html_links.cfm,Example Link
2 http://www.quackit.com/html/examples/html_links_examples.cfm,More Link Examples...
```

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search RA1911033010047

Prepare > Regex > Applications > Detect HTML Tags

Detect HTML Tags ★ Points: 710 Rank: 1

Problem Submissions Leaderboard | Discussions

You made this submission 3 months ago.
Score: 10.00 Status: Accepted

Submitted Code

Language: Python 3

```
1 import re
2 pattern = re.compile('<\s*([a-zA-Z0-9]+)\s*>?')
3 n = int(input())
4 tag_list = list()
5 for i in range(n):
6     x = str(input())
7     y = re.findall(pattern, x)
8     for z in y:
9         if(len(z) != 0 and z not in tag_list):
10             tag_list.append(z)
11 tag_list.sort()
12 for i in range(len(tag_list)):
13     if(i == len(tag_list) - 1):
14         print(tag_list[i])
15     else:
16         print(tag_list[i].end=":")
```

NEED HELP?

View discussions View top submissions

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input (stdin)

```
1 2
2 <p><a href="http://www.quackit.com/html/tutorial/html_links.cfm">Example Link</a></p>
```

Test case 3

Test case 4

Test case 5

Test case 6 Expected Output

```
1 a;div;p
```

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search Notifications RA1911033010047

Prepare > RegEx > Applications > Find A Sub-Word

Find A Sub-Word ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 2 months ago.

Score: 10.00 Status: Accepted

Submitted Code

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

```
1 import re
2 data = []
3 for i in range(int(input().strip())):
4     data.append(input().strip())
5 for i in range(int(input().strip())):
6     query = input().strip()
7     matches_count = 0
8     for e in data:
9         matches = re.findall(r'[A-Za-z_]+'+query+r'[A-Za-z_]', e)
10        matches_count += len(matches)
11    print(matches_count)
12
```

NEED HELP?

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

Input(stdin) Download

```
1
2 existing pessimist optimist this is
3
4 is
```

Expected Output Download

```
1 3
```

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE

Search RA1911033010047

Prepare > Regex > Applications > Alien Username

Alien Username ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 3 months ago.

Score: 10.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 import re
2 pattern = re.compile('^\[_\.\][0-9]+[a-zA-Z]*[_]?$')
3 n = int(input())
4 for i in range(n):
5     x = str(input())
6     if(len(re.findall(pattern,x)) != 0):
7         print("VALID")
8     else:
9         print("INVALID")
10
```

NEED HELP?

View discussions View editorial View top submissions

Test case 0 Compiler Message Success

Test case 1 Input (stdin) Download

```
1 3
2 _0898989811abcd_
3 _abce
4 _09090909abcD0
```

Test case 2 Expected Output Download

```
1 VALID
2 INVALID
```

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search RA1911033010047

Prepare > Regex > Applications > IP Address Validation

IP Address Validation ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission 3 months ago.

Score: 10.00 Status: Accepted

Submitted Code

Language: Python 3

```
1 import re
2 p1 = re.compile('^(?:(2[0-5][0-5])|(1[0-9][0-9])|([1-9][0-9])|(\d))\.(?
3 (?:(2[0-5][0-5])|(1[0-9][0-9])|([1-9][0-9])|(\d))$')
4 p2 = re.compile('^(?:(a-f0-9){1,4}:){7}(a-f0-9){4}$')
5 n = int(input())
6 for i in range(n):
7     x = str(input())
8     if(len(re.findall(p1, x)) != 0):
9         print("IPv4")
10    elif(len(re.findall(p2, x)) != 0):
11        print("IPv6")
12    else:
13        print("Neither")
```

NEED HELP?

View discussions
 View editorial
 View top submissions

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input(stdin) Download

```
1 7
```

Test case 3

```
2 1050:1000:1000:a000:5:600:300c:326b
3 1050:1000:2000:ab00:5:600:300c:326a
4 1050:1000:3000:abc0:5:600:300c:326c
5 1051:1000:4000:abcd:5:600:300c:326b
6 22.231.113.64
7 22.231.113.164
8 222.231.113.64
```

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search · Notifications · RA1911033010047

Prepare > Regex > Applications > Find a Word

Find a Word ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

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

```
1 import re
2 n = int(input())
3 sen = list()
4 for i in range(n):
5     x = str(input())
6     sen.append(x)
7 t = int(input())
8 for i in range(t):
9     x = str(input())
10    p = re.compile('(?:\W|\A)'+x+'(?:\W|\Z)')
11    c = 0
12    for y in sen:
13        c += len(re.findall(p, y))
14    print(c)
15
```

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input (stdin) Download

```
1
2 foo bar (foo) bar foo-bar foo_bar foo'bar bar~foo bar, foo.
```

Test case 3

Test case 4

Test case 5 Expected Output Download

```
1 6
```

Test case 6

EXPERIMENT - 11

Title: Implementation of DAG

Aim: To implement the DAG

Algorithm:

- Case (i) $x := y \text{ OP } z$
- Case (ii) $x := \text{OP } y$
- Case (iii) $x := y$
- Step 1:
 - If y operand is undefined then create node(y).
 - If z operand is undefined then for case(i) create node(z).
- Step 2:
 - For case(i), create node(OP) whose right child is node(z) and left child is node(y).
 - For case(ii), check whether there is node(OP) with one child node(y).
 - For case(iii), node n will be node(y).
- Output:
 - For node(x) delete x from the list of identifiers. Append x to the attached identifiers list for the node n found in step 2. Finally set node(x) to n.

Code:

```
1 //RA1911033010047
2 #include<iostream>
3 #include<string>
4 #include<unordered_map>
5 using namespace std;
6 class DAG
7 { public:
8     char label;
9     char data;
10    DAG* left;
11    DAG* right;
12
13    DAG(char x){
14        label='_';
15        data=x;
16        left=NULL;
17        right=NULL;
18    }
19    DAG(char lb, char x, DAG* lt, DAG* rt){
20        label=lb;
21        data=x;
22        left=lt;
23        right=rt;
24    }
25 };
26
27 int main(){
28     int n;
29     n=3;
30     string st[n];
31     st[0] = "A=x+y";
32     st[1] = "B=A*z";
33     st[2] = "C=B/x";
34     unordered_map<char, DAG*> labelDAGNode;
35 }
```

Activate Windows

```
36    for(int i=0;i<3;i++){
37        string stTemp=st[i];
38        for(int j=0;j<5;j++){
39            char tempLabel = stTemp[0];
40            char tempLeft = stTemp[2];
41            char tempData = stTemp[3];
42            char tempRight = stTemp[4];
43            DAG* leftPtr;
44            DAG* rightPtr;
45            if(labelDAGNode.count(tempLeft) == 0){
46                leftPtr = new DAG(tempLeft);
47            }
48            else{
49                leftPtr = labelDAGNode[tempLeft];
50            }
51            if(labelDAGNode.count(tempRight) == 0){
52                rightPtr = new DAG(tempRight);
53            }
54            else{
55                rightPtr = labelDAGNode[tempRight];
56            }
57            DAG* nn = new DAG(tempLabel,tempData,leftPtr,rightPtr);
58            labelDAGNode.insert(make_pair(tempLabel,nn));
59        }
60    }
61    cout<<"Label      ptr      leftPtr      rightPtr" << endl;
62    for(int i=0;i<n;i++){
63        DAG* x=labelDAGNode[st[i][0]];
64        cout<<st[i][0]<<"      "<<x->data<<"      ";
65        if(x->left->label=='_')cout<<x->left->data;
66        else cout<<x->left->label;
67        cout<<"      ";
68        if(x->right->label=='_')cout<<x->right->data;
69        else cout<<x->right->label;
70        cout<<endl;
71    }
72    return 0;
73 }
```

Activate Windows

Activate Windows

RA1911033010047

KOLLA SESHU KIRAN

Output:

```
Label      ptr      leftPtr      rightPtr
A          +
B          *
C          /
```

```
...Program finished with exit code 0
Press ENTER to exit console.[]
```

Result: The program for implementation of DAG is successfully implemented.

EXPERIMENT - 11

Title: Implementation of three address code

Aim: To write a program to implement three address code: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 has no next uses or does 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 registers will no longer contain y or z.

RA1911033010047

KOLLA SESHU KIRAN

Code:

```
1 //RA1911033010047
2 #include<stdio.h>
3 #include<ctype.h>
4 #include<stdlib.h>
5 #include<string.h>
6 void small();
7 void dove(int i);
8 int p[5]={0,1,2,3,4},c=1,i,k,l,m,pi;
9 char sw[5]={'=','-', '+', '/', '*'},j[20],a[5],b[5],ch[2];
10 void main()
11 {
12 printf("Enter the expression:");
13 scanf("%s",j);
14 printf("\tThe Intermediate code is:\n");
15 small();
16 }
17 void dove(int i)
18 {
19 a[0]=b[0]='\0';
20 if(!isdigit(j[i+2])&&!isdigit(j[i-2]))
21 {
22 a[0]=j[i-1];
23 b[0]=j[i+1];
24 }
25 if(isdigit(j[i+2])){
26 a[0]=j[i-1];
27 b[0]='t';
28 b[1]=j[i+2];
29 }
30 if(isdigit(j[i-2])){
31 {
32 b[0]=j[i+1];
33 a[0]='t';
34 a[1]=j[i-2];
35 b[1]='\0';
36 }
37 if(isdigit(j[i+2]) &&isdigit(j[i-2]))
38 {
39 a[0]='t';
40 b[0]='t';
41 a[1]=j[i-2];
42 b[1]=j[i+2];
43 sprintf(ch,"%d",c);
44 j[i+2]=j[i-2]=ch[0];
45 }
46 if(j[i]=='*')
47 printf("\tt%d=%s*s\n",c,a,b);
48 if(j[i]== '/')
49 printf("\tt%d=%s/%s\n",c,a,b);
50 if(j[i]== '+')
51 printf("\tt%d=%s+%s\n",c,a,b);if(j[i]==' -')
52 printf("\tt%d=%s-%s\n",c,a,b);
53 if(j[i]== '=')
54 printf("\ttc=t%d",j[i-1],--c);
55 sprintf(ch,"%d",c);
56 j[i]=ch[0];
57 c++;
58 small();
59 }
60 void small()
61 {
62 pi=0;l=0;
63 for(i=0;i<strlen(j);i++)
64 {
65 for(m=0;m<5;m++)
66 if(j[i]==sw[m])
67 if(pi<=p[m])
68 {
69 pi=p[m];
70 l=1;
71 k=i;
72 }
73 }
74 if(l==1)
75 dove(k);
76 else
```

Activate Windows

Go to Settings to activate Windows.

Activate Windows

Go to Settings to activate Windows

```
77 exit(0);}
```

Activate Windows

Output:

```
Enter the expression:(a+b)*(a-b)
The Intermediate code is:
t1=+
t2=a+b
t3=a-b

...Program finished with exit code 0
Press ENTER to exit console.[]
```

Result: The program was successfully implemented, compiled and run output is obtained.

EXPERIMENT - 13

Title:

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 a 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 in 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:

1) Stack

```
1 //RA1911033010047|
2 #include<stdio.h>
3 #include<conio.h>
4
5 int i, stk[100], top=-1, n;
6 void show()
7 {
8     for(i=0;i<=top;i++)
9         printf("%d\t",stk[i]);
10 }
11 void push()
12 {
13     int item;
14     if(top == n-1)
15         printf("\nStack is full.");
16     else
17     {
18         printf("\nEnter the item: ");
19         scanf("%d",&item);
20         stk[++top]=item;
21     }
22 void pop()
23 {
24     if(top== -1)
25         printf("Stack is empty.");
26     else
27     {
28         printf("%d is popped.",stk[top]);
29         top--;
30     }
31 int main()
32 {
33     int i,op;
34     printf("Enter the size of the stack: ");
35     scanf("%d",&n);
36     do
37     {
38         printf("\n1 : Push");

```

Activate Windows
Go to Settings to activate Windows.

```
39         printf("\n2 : Pop");
40         printf("\n3 : Display");
41         printf("\n4 : Exit");
42         printf("\nEnter your choice: ");
43         scanf("%d",&op);
44         switch(op)
45         {
46             case 1:
47                 push();
48                 break;
49             case 2:
50                 pop();
51                 break;
52             case 3:
53                 show();
54                 break;
55         }
56     }while(op!=4);
57     getch();
58 }
59
60
```

Activate Windows
Go to Settings to activate Windows.

RA1911033010047

KOLLA SESHU KIRAN

Output:

```
Enter the size of the stack: 5
1 : Push
2 : Pop
3 : Display
4 : Exit
Enter your choice: 1

Enter the item: 2
1 : Push
2 : Pop
3 : Display
4 : Exit
Enter your choice: 1

Enter the item: 3
1 : Push
2 : Pop
3 : Display
4 : Exit
Enter your choice: 1

Enter the item: 5
1 : Push
2 : Pop
3 : Display
4 : Exit
Enter your choice: 2
5 is popped.
1 : Push
2 : Pop
3 : Display
4 : Exit
Enter your choice: 1
Enter the item: 7
```

Activate Windows
Go to Settings to activate Win

```
1 : Push
2 : Pop
3 : Display
4 : Exit
Enter your choice: 3
2      3      7
1 : Push
2 : Pop
3 : Display
4 : Exit
Enter your choice: 4
```

2)Heap:

```
1 //RA1911033010047
2 #include <iostream>
3 #include <cstdlib>
4 #include <vector>
5 #include <iterator>
6 using namespace std;
7 /*
8  * Class Declaration
9  */
10 class Heap
11 {
12     private:
13         vector <int> heap;
14         int left(int parent);
15         int right(int parent);
16         int parent(int child);
17         void heapifyup(int index);
18         void heapifydown(int index);
19     public:
20         Heap()
21     {}
22         void Insert(int element);
23         void DeleteMin();
24         int ExtractMin();
25         void DisplayHeap();
26         int Size();
27 };
28 /*
29  * Return Heap Size
30  */
31 int Heap::Size()
32 {
33     return heap.size();
34 }
35 /*
36  * Insert Element into a Heap
37  */
38 */
```

Activate Wi
Go to Settings !

RA1911033010047

KOLLA SESHU KIRAN

```
39 void Heap::Insert(int element)
40 {
41     heap.push_back(element);
42     heapifyup(heap.size() - 1);
43 }
44 /*
45 * Delete Minimum Element
46 */
47 void Heap::DeleteMin()
48 {
49     if (heap.size() == 0)
50     {
51         cout<<"Heap is Empty"<<endl;
52         return;
53     }
54     heap[0] = heap.at(heap.size() - 1);
55     heap.pop_back();
56     heapifydown(0);
57     cout<<"Element Deleted"<<endl;
58 }
59
60 /*
61 * Extract Minimum Element
62 */
63 int Heap::ExtractMin()
64 {
65     if (heap.size() == 0)
66     {
67         return -1;
68     }
69     else
70         return heap.front();
71 }
72
73 /*
74 * Display Heap
75 */
76 void Heap::DisplayHeap()

77 {
78     vector <int>::iterator pos = heap.begin();
79     cout<<"Heap --> ";
80     while (pos != heap.end())
81     {
82         cout<<*pos<<" ";
83         pos++;
84     }
85     cout<<endl;
86 }
87
88 /*
89 * Return Left Child
90 */
91 int Heap::left(int parent)
92 {
93     int l = 2 * parent + 1;
94     if(l < heap.size())
95         return l;
96     else
97         return -1;
98 }
99
100 /*
101 * Return Right Child
102 */
103 int Heap::right(int parent)
104 {
105     int r = 2 * parent + 2;
106     if(r < heap.size())
107         return r;
108     else
109         return -1;
110 }
111
112 /*
113 * Return Parent
114 */
```

Activate Windows
Go to Settings to activate Windows

Activate Windows
Go to Settings to activate Windows

RA1911033010047

KOLLA SESHU KIRAN

```
115 int Heap::parent(int child)
116 {
117     int p = (child - 1)/2;
118     if(child == 0)
119         return -1;
120     else
121         return p;
122 }
123
124 /*
125 * Heapify- Maintain Heap Structure bottom up
126 */
127 void Heap::heapifyup(int in)
128 {
129     if (in >= 0 && parent(in) >= 0 && heap[parent(in)] > heap[in])
130     {
131         int temp = heap[in];
132         heap[in] = heap[parent(in)];
133         heap[parent(in)] = temp;
134         heapifyup(parent(in));
135     }
136 }
137
138 /*
139 * Heapify- Maintain Heap Structure top down
140 */
141 void Heap::heapifydown(int in)
142 {
143
144     int child = left(in);
145     int child1 = right(in);
146     if (child >= 0 && child1 >= 0 && heap[child] > heap[child1])
147     {
148         child = child1;
149     }
150     if (child > 0)
151     {
152         int temp = heap[in];
153
154         heap[in] = heap[child];
155         heap[child] = temp;
156         heapifydown(child);
157     }
158
159 /*
160 * Main Contains Menu
161 */
162 int main()
163 {
164     Heap h;
165     while (1)
166     {
167         cout<<"-----"<<endl;
168         cout<<"Operations on Heap"<<endl;
169         cout<<"-----"<<endl;
170         cout<<"1.Insert Element"<<endl;
171         cout<<"2.Delete Minimum Element"<<endl;
172         cout<<"3.Extract Minimum Element"<<endl;
173         cout<<"4.Print Heap"<<endl;
174         cout<<"5.Exit"<<endl;
175         int choice, element;
176         cout<<"Enter your choice: ";
177         cin>>choice;
178         switch(choice)
179         {
180             case 1:
181                 cout<<"Enter the element to be inserted: ";
182                 cin>>element;
183                 h.Insert(element);
184                 break;
185             case 2:
186                 h.DeleteMin();
187                 break;
188             case 3:
189                 cout<<"Minimum Element: ";
190                 if (h.ExtractMin() == -1)
```

Activate Windows
Go to Settings to activate Win

Activate Windows
Go to Settings to activate Windo

```

191 -     {
192     cout<<"Heap is Empty"<<endl;
193   }
194   else
195     cout<<"Minimum Element: "<<h.ExtractMin()<<endl;
196   break;
197 case 4:
198   cout<<"Displaying elements of Hwap: ";
199   h.DisplayHeap();
200   break;
201 case 5:
202   exit(1);
203 default:
204   cout<<"Enter Correct Choice"<<endl;
205 }
206 }
207 return 0;
208 }
```

Activate Windows
Go to Settings to activate Wind

Output:

```

-----
Operations on Heap
-----
1.Insert Element
2.Delete Minimum Element
3.Extract Minimum Element
4.Print Heap
5.Exit
Enter your choice: 1
Enter the element to be inserted: 2
-----
Operations on Heap
-----
1.Insert Element
2.Delete Minimum Element
3.Extract Minimum Element
4.Print Heap
5.Exit
Enter your choice: 4
Displaying elements of Hwap: Heap --> 2
-----
Operations on Heap
-----
1.Insert Element
2.Delete Minimum Element
3.Extract Minimum Element
4.Print Heap
5.Exit
```

Activate V
Go to Setting

Result : The programs for different Algorithms for various storage allocation techniques were implemented successfully.

RA1911033010047

Kolla Seshu Kiran

Compiler Design

18/4/22

* Assignment - 5 *

* Global Data-Flow Analysis *

- To apply global optimizations on basic blocks, data-flow information is collected by solving systems of data-flow equations.
- Suppose we need to determine the live-variables for a sequence of statements S .

$$in[S] = use[S] \cup (out[s] - def[s])$$

B1 : $i := m - 1$ solution:

$$j := n \quad in[B1] = \{m, n\} \cup (\{i, j\} - \{i, j\}) = \{m, n\}$$

B2 : $j := j - 1$ $out[B1] = in[B2] = \{i, j\}$

B3 : $i := i + j$ $in[B2] = \{j\} \cup (\{i, j\} - \{j\}) = \{i, j\}$

$$out[B2] = in[B3] = \{i, j\}$$

- Suppose we need to determine the reaching definitions for a sequence of statements S .

$$out[S] = gen[S] \cup (in[S] - kill[S])$$

$$\begin{array}{ll}
 \text{B1: } d_1 : i := m-1 & \text{out}[B1] = \text{gen}[B1] = \{d_1, d_2\} \\
 d_2 : j := n & \text{out}[B2] = \text{gen}[B2] \cup \{d_1\} \\
 \text{B2: } d_3 : j := j-1 & \Rightarrow \{d_1, d_3\}
 \end{array}$$

B3:

d_1 reaches B2 and B3 and
 d_2 reaches B2, but not B3
because d_2 is killed in B2.

- To effectively optimize the code compiler collects all the information about the program and distributes the information to each block of the flow graph. This process is known as data-flow graph analysis.

- A framework for proving facts about program

Gen & kill sets:-

Gen sets:- A current basic block or instruction is said to be in the GEN set if it creates the definition. In any case, the must be in the output.

- Block B generates def d if d is in B and is not followed in B by any unambiguous defs of the same variable as d.

- $\text{GEN}(B)$ is the set of defs generated by B.

The transfer equation for B is :-

$$\text{OUT}(B) = \text{IN}(B) - \text{KILL}(B) \cup \text{GEN}(B)$$

- Kill sets:- A current basic block or instruction is said to be in the KILL set if it redefines a variable in the expression. After that, the expression is invalid.

- Say def d kills def e if d and e define the same variable, and d is unambiguous. Ambiguous defs cannot kill other defs.

- $\text{kill}(B)$ is the set of sets def not in B that are killed by some def in B.

RA1911033010047

KOLLA SESHU KIRAN

HACKERRANK REGEX PROBLEMS(32 - 47)

Date: 18-04-2022

The screenshot shows the HackerRank platform interface for the 'Detect the Email Addresses' challenge. At the top, there's a navigation bar with 'PREPARE NEW', 'CERTIFY', and 'COMPETE' buttons. On the right, there's a search bar, a notification icon, and a user profile for 'RA1911033010047'. Below the navigation, the challenge title 'Detect the Email Addresses' is displayed with a star icon, and the user's current stats: 'Points: 710 Rank: 1'.

The main content area has tabs for 'Problem', 'Submissions', 'Leaderboard', and 'Discussions'. Under 'Problem', it says 'You made this submission 3 months ago.' and shows a 'Score: 15.00 Status: Accepted'. The 'Submitted Code' section shows Python 3 code:

```
Language: Python 3 Open in editor
1 import re
2 p = re.compile('[_0-9a-zA-Z]+[_\.0-9a-zA-Z]*@[0-9a-zA-Z]+[\.0-9a-zA-Z_]+[0-9a-zA-Z_]*')
3 n = int(input())
4 l = []
5 for i in range(n):
6     x = str(input())
7     r = re.findall(p, x)
8     for z in r:
9         if(z not in l):
10             l.append(z)
11 l.sort()
12 for i in range(len(l)):
13     if(i == len(l)-1):
14         print(l[i])
```

The 'Test cases' section lists four successful test cases:

- Test case 0: Compiler Message Success
- Test case 1: Success
- Test case 2: Input(stdin) Success
- Test case 3: Success
- Test case 4: Success

Test case 2 shows the input '36' and the output 'Finally this phone is testimony to our quest and ever open ears for hearing from our customers since 1921. We look forward to hearing from you today.' Test case 4 shows the input 'All India National Toll Free Number: 180 0425 0426 Working Hours: 10:00 am to 6:00 pm (Monday ~ Friday), 10:00 am to 2:00 pm (Saturday). To report ATM Card Lost, Kindly contact: +91 (44) 2622 3106 / 2622 3109. TMB Customer Care: +91 9842 461 461 For all your queries. on any of our services in any branch in' and the output 'Finally this phone is testimony to our quest and ever open ears for hearing from our customers since 1921. We look forward to hearing from you today.'.

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search Search Notifications RA1911033010047

Prepare > Regex > Applications > Detect the Domain Name

Detect the Domain Name ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 import re
2 pattern = '(http|https)://(www.|ww2.|)([a-zA-Z0-9\\-\\.]+)(\\.[a-
3 regex = re.compile(pattern)
4 s = set()
5 for i in range(int(input())):
6     string = input()
7     iterator = regex.finditer(string)
8     if iterator:
9         for match in iterator:
10             s.add(match.group(3)+match.group(4))
11 print(''.join(t for t in sorted(s)))
```

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input(stdin) Download

```
1 1027
```

Test case 3 Input(stdin) Download

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml/DTD/xhtml-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
6 />
7 <meta name="format-detection" content="telephone=no" />
8 <title>Rediff.com - India, Business, Stock, Sports, Cricket,
9 Entertainment, Bollywood, Music, Video and Breaking news.
```

Test case 4 Input(stdin) Download

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Applications > Building a Smart IDE: Identifying comments

Building a Smart IDE: Identifying comments ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission a month ago.
Score: 20.00 Status: Accepted

Submitted Code

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

```
1 import re
2
3 import io
4 import sys
5 input_stream = io.TextIOWrapper(sys.stdin.buffer, encoding='utf-
8')
6
7 s = input_stream.readlines()
8 s = "\n".join(s)
9
10 pc = '(//.*|/*[\d\0]*?*/)'
11 mc = re.findall(pc, s)
12 for v in mc:
13     vs = v.split("\n\n")
14     for l in vs:
```

Test cases

Test case 0 Compiler Message Success

Test case 1

Test case 2

Test case 3

Test case 4 **Hidden Test Case** Unlock this testcase for 5 hackos. [Unlock](#)

Test case 5

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Applications > Detecting Valid Latitude and Longitude Pairs

Detecting Valid Latitude and Longitude Pairs ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission 3 months ago.

Score: 20.00 Status: Accepted

Submitted Code

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

```
1 import re
2 p = re.compile('([+-]?\d*\.\d*)', '([+-]?\d*\.\d*)')
3 t = int(input())
4 for i in range(t):
5     try:
6         z = str(input())
7         z = z[1:len(z)-1]
8         zz = re.findall(p, z)
9         zz = zz[0]
10        x = zz[0]
11        y = zz[1]
12        if(x[0] == '+' or x[0] == '-'):
13            x = x[1:]
14        if(y[0] == '+' or y[0] == '-'):
15            y = y[1:]
```

Test cases

Test case	Compiler Message
Test case 0	Success
Test case 1	
Test case 2	Input (stdin) Download
Test case 3	
Test case 4	
Test case 5	

Test case 0 Compiler Message Success

Test case 1

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

12

Test case 3

75, 180
(+90.0, -147.45)

Test case 4

77.11112223331, 149.99999999
(+90, +180)

Test case 5

90, 180
(-90.00000, -180.0000)
75, 280
(+190.0, -147.45)

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Applications > HackerRank Tweets

HackerRank Tweets ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 import re
2 p = re.compile('((h|H)(a|A)(c|C)(k|K)(e|E)(r|R)(a|R)(n|N)
3 (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)
```

Test case 0 Compiler Message Success

Test case 1 Input (stdin) Download

```
1 4
```

Test case 2 Input (stdin) Download

```
1 I love #hackerrank
```

Test case 3 Input (stdin) Download

```
1 I just scored 27 points in the Picking Cards challenge on
2 #HackerRank
```

Test case 4 Input (stdin) Download

```
1 I just signed up for summer cup @hackerrank
```

Test case 5 Input (stdin) Download

```
1 interesting talk by hari, co-founder of hackerrank
```

Test case 6 Expected Output Download

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE CERTIFY COMPETE

Search RA1911033010047

Prepare > Regex > Applications > Build a Stack Exchange Scraper

Build a Stack Exchange Scraper ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions

You made this submission a month ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 import re
2 import sys
3
4 if __name__ == '__main__':
5     s = sys.stdin.read()
6
7     Regex = r'question-summary-(\w\w\w\w)', *?class="question-
hyperlink">(.*?)</a>, *?class="relativetime">(.*?)</span>
8     li = re.findall(Regex, s, re.DOTALL)
9
10    for a in li:
11        print(''.join(a))
12
```

NEED HELP?

View discussions View top submissions

Test case 0 Compiler Message Success

Test case 1

Test case 2

Test case 3 Hidden Test Case Unlock this testcase for 5 hackos.

Test case 4 Unlock

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Applications > Utopian Identification Number ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission 3 months ago.
Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3 Open In editor

```
1 import re
2 p = re.compile('^[a-z]{0,3}[0-9]{2,8}[A-Z]{3,}')
3 n = int(input())
4 for i in range(n):
5     x = str(input())
6     if(len(re.findall(p, x)) > 0):
7         print("VALID")
8     else:
9         print("INVALID")
```

Test case 0 Compiler Message Success

Test case 1 Input(stdin) Download

```
1 2
2 abc012333ABCDEEEE
3 0123AB
```

Test case 2 Expected Output Download

```
1 VALID
2 INVALID
```

Test case 3

Test case 4

Test case 5

Test case 6

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search RA1911033010047

Prepare > Regex > Applications > Valid PAN format

Valid PAN format ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 import re
2 p = re.compile('^[A-Z]{5}[0-9]{4}[A-Z]$')
3 n = int(input())
4 for i in range(n):
5     x = str(input())
6     if(len(x) != 10):
7         print("NO")
8     elif(len(re.findall(p, x)) > 0):
9         print("YES")
10    else:
11        print("NO")
```

Test case 0 Compiler Message Success

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Input (stdin)

Download

1 3
2 ABCDS1234Y
3 ABAB12345Y
4 avCDS1234Y

Expected Output

Download

1 YES
2 NO

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search RA1911033010047

Prepare > Regex > Applications > Find HackerRank

Find HackerRank ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions Editorial

You made this submission a month ago.

Score: 15.00 Status: Accepted

Submitted Code

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

```
1 import re
2
3 n_lines = int(input())
4
5 for i in range(n_lines):
6     line = input()
7     match1 = re.search(r'^^(hackerrank)', line)
8     match2 = re.search(r'^(hackerrank)$', line)
9     match3 = re.search(r'^^(hackerrank)$', line)
10
11     if match3:
12         print(0)
13     elif match1:
14         print(1)
15     elif match2:
```

NEED HELP?

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

Test case 0 [Success](#)

Compiler Message Success

Input(stdin)

```
1 4
2 i love hackerrank
3 hackerrank is an awesome place for programmers
4 hackerrank
5 i think hackerrank is a great place to hangout
```

Download

Expected Output

```
1 2
```

Download

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE

Prepare > Regex > Applications -> Saying Hi

Saying Hi ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission 2 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 n = int(input())
2 import re
3 p = re.compile('^[Hh][i|i]\s[^dD].*')
4 for i in range(n):
5     x = str(input())
6     if(len(re.findall(p, x)) > 0):
7         print(x)
8
```

Test case 0 Compiler Message Success

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Input(stdin)

1 5
2 Hi Alex how are you doing
3 hi dave how are you doing
4 Good by Alex
5 hidden agenda
6 Alex greeted Martha by saying Hi Martha

Download

Expected Output

Download ▾

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Applications > HackerRank Language

HackerRank Language ★

Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions

You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 import re
2 p = re.compile('^\d+\s+
(C|CPP|JAVA|PYTHON|PERL|PHP|RUBY|CSHARP|HASKELL|CLOJURE|BASH|SCALA|ER
LANG|CLISP|LUA|BRAINFUCK|JAVASCRIPT|GO|D|OCAML|R|PASCAL|SBCL|DART|GRO
OVY|OBJECTIVEC)$')
3 n = int(input())
4 for i in range(n):
5     x = str(input())
6     if(len(re.findall(p, x)) > 0):
7         print("VALID")
8     else:
9         print("INVALID")
10
```

NEED HELP?

View discussions View top submissions

Test case 0 Compiler Message Success

Test case 1 Input(stdin) Download

```
1 3
```

Test case 2 Input(stdin) Download

```
1 11011 C
```

Test case 3 Input(stdin) Download

```
1 11022 CPP
```

Test case 4 Input(stdin) Download

```
1 11044 X
```

Test case 5 Expected Output Download

```
1 VALID
```

Test case 6 Expected Output Download

```
1 VALID
```

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE

Search RA1911033010047

Prepare > Regex > Applications > Building a Smart IDE: Programming Language Detection

Building a Smart IDE: Programming Language Detection ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission a month ago.

Score: 30.00 Status: Accepted

Submitted Code

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

```
1
2 import re
3 import sys
4
5 class Main:
6     def __init__(self):
7         self.s = ''.join(sys.stdin.readlines())
8
9     def output(self):
10        if 'java' in self.s:
11            print("Java")
12        elif '#include' in self.s:
13            print("C")
14        else:
15            print("Python")
```

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input (stdin) Download

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

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Applications > Split the Phone Numbers ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP? View discussions View top submissions

You made this submission 2 months ago.
Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

```
1 import re
2 p = re.compile('(\d{1,3})[-|\s](\d{1,3})[-|\s](\d{4,10})')
3 n = int(input())
4 for i in range(n):
5     x = str(input())
6     if(len(re.findall(p, x)) > 0):
7         y = re.findall(p, x)[0]
8
9     print("CountryCode="+y[0]+",LocalAreaCode="+y[1]+",Number="+y[2])
```

Test case 0 Compiler Message Success

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Hidden Test Case
Unlock this testcase for 5 hacks! Unlock

RA1911033010047

KOLLA SESHU KIRAN

HackerRank NEW PREPARE CERTIFY COMPETE

Search Search Notifications Notifications RA1911033010047

Prepare > Regex > Applications > Detect HTML Attributes

Detect HTML Attributes ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission a month ago.

Score: 20.00 Status: Accepted

Submitted Code

Language: Python 3 Open in editor

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

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input(stdin) Download

```
1 2
2: <p><a href="http://www.quackit.com/html/tutorial/html_links.cfm">Example Link</a></p>
3: <div class="more-info"><a href="http://www.quackit.com/html/examples/html_links_examples.cfm">More Link Examples...</a></div>
```

Test case 3

Test case 4

Test case 5

Test case 6 Expected Output Download

```
1 a:href
```

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Applications > The British and American Style of Spelling

The British and American Style of Spelling ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission 2 months ago.
Score: 15.00 Status: Accepted

Submitted Code

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

```
1 import re
2 data = " ".join([input().strip() for i in
3     range(int(input().strip()))])
4 for i in range(int(input().strip())):
5     print(len(re.findall(input()[:-2]+"(ze|se)", data)))
```

Test case 0 Compiler Message Success

Test case 1 Input (stdin) Download

```
1 2
2 hackerrank ui is easy to familiarise with
3 to familiarize oneself with ui of hackerrank is easy
4 1
5 familiarize
```

Test case 2 Expected Output Download

```
1 2
```

RA1911033010047

KOLLA SESHU KIRAN

HackerRank PREPARE NEW CERTIFY COMPETE Search RA1911033010047

Prepare > Regex > Applications > UK and US: Part 2

UK and US: Part 2 ★ Points: 710 Rank: 1

Problem Submissions Leaderboard Discussions NEED HELP?

You made this submission 2 months ago.

Score: 10.00 Status: Accepted

Submitted Code

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

```
1 import re
2 sent = []
3 n = int(input())
4 for i in range(n):
5     x = str(input())
6     sent.append(x)
7 t = int(input())
8 for i in range(t):
9     x = str(input())
10    y = x
11    c = 0
12    x = x.replace('our', 'or')
13    p = re.compile('(?:\s|\A)'+('+'x+'|'+y+')'+'(?\s|\Z)')
14    for sen in sent:
15        c += len(re.findall(p, sen))
```

Test case 0 Compiler Message Success

Test case 1

Test case 2 Input (stdin) Download

```
1 2
```

Test case 3

Test case 4

Test case 5

Test case 6 Expected Output Download

```
1 2
```