18CSC304J COMPILER DESIGN

SEMESTER - VI

YEAR: 2022

NAME: Harsh Saxena

REG NO: RA1911033010060

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be **University** *u/s 3 of UGC Act, 1956)*

SCHOOL OF COMPUTING

SRM Institute of Science and Technology

(Under Section 3 of UGC Act, 1956)
S.R.M. NAGAR, KATTANKULATHUR- 603 203.

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

BONAFIDE CERTIFICATE

Register No: RA1911033010060

Certified to be the bonafide record of work done by Harsh Saxena of Computer Science B.Tech Degree course in the Practical Compiler Design in **SRM Institute of Science and Technology, Kattankulathur** during the academic year .

Lab Incharge

Date: **Head of the Department**

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

Date : **Examiner 1** **Examiner 2**

Table of Contents

Week 1
Lexical analysis

Aim:- the aim is to convert the input from a simple sequence of characters into a list of tokens of different kinds, such as numerical and string constants, variable identifiers, and programming language keywords.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
int isKeyword(char buffer[]){
char keywords[32][10] =
{"auto","break","case","char","const","continue","default",
"do","double","else","enum","extern","float","for","goto",
"if","int","long","register","return","short","signed",
"sizeof","static","struct","switch","typedef","union",
"unsigned","void","volatile","while"};
int i, flag = 0;
for(i = 0; i < 32; ++i){
if(strcmp(keywords[i], buffer) == 0){
flag = 1;
break;
}
}
return flag;
}
int main(){
char ch, buffer[15], operators[] = "+-*/%=";
FILE *fp;
int i,j=0;
fp = fopen("c.txt","r");
if(fp == NULL){
```

```
printf("error while opening the file\n");
exit(0);
}
while((ch = fgetc(fp)) != EOF){
for(i = 0; i < 6; ++i){
    if(ch == operators[i])
printf("%c is operator\n", ch);
}
if(isalnum(ch)){
buffer[j++] = ch;
}
else if((ch == ' ' || ch == '\n') && (j != 0)){
buffer[j] = '\0';
j = 0;
if(isKeyword(buffer) == 1)
printf("%s is keyword\n", buffer);
else
printf("%s is indentifier\n", buffer);
}
}
fclose(fp);
return 0;
}
```

```
TOKEN IDENTIFICATION
--------------
#include<iostream.h>  is an IDENTIFIER
#include<cstring.h>  is an IDENTIFIER
using  is a KEYWORD
namespace  is a KEYWORD
std  is a KEYWORD
;  is a SPECIAL SYMBOL
class  is an IDENTIFIER
Student  is an IDENTIFIER
{  is a SPECIAL SYMBOL
public:  is an IDENTIFIER
int  is a KEYWORD
id  is an IDENTIFIER
;  is a SPECIAL SYMBOL
string  is an IDENTIFIER
name  is an IDENTIFIER
;  is a SPECIAL SYMBOL
}  is a SPECIAL SYMBOL
;  is a SPECIAL SYMBOL
void  is a KEYWORD
ERROR:main() function missing brackets
-----------------------------


TOKEN COUNT
--------------
KEYWORDS: 5
OPERATORS: 0
Special Symbol: 6
NUMBERS: 0
IDENTIFIERS: 8
```

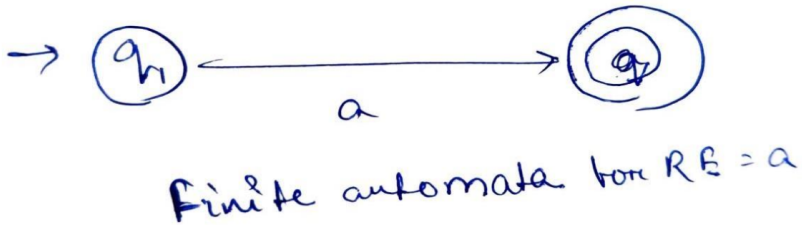
Result: During Lexical Analysis, all the tokens where identified separately and the error was found.

Week 2
RE to NFA

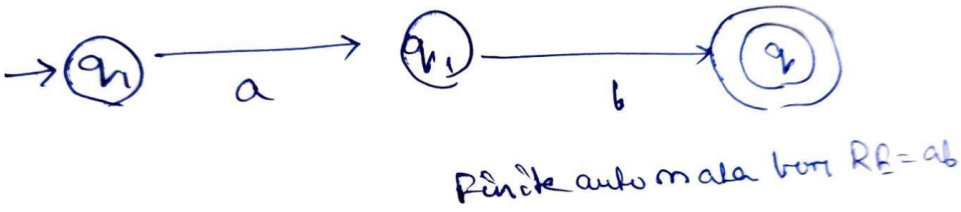
Aim: Program to convert the given regular expression to a transition table from which an NFA diagram can be drawn

Algorithm: We can use Thompson's Construction to find out a Finite Automaton from a Regular Expression. We will reduce the regular expression into the smallest regular expressions and convert these to NFA

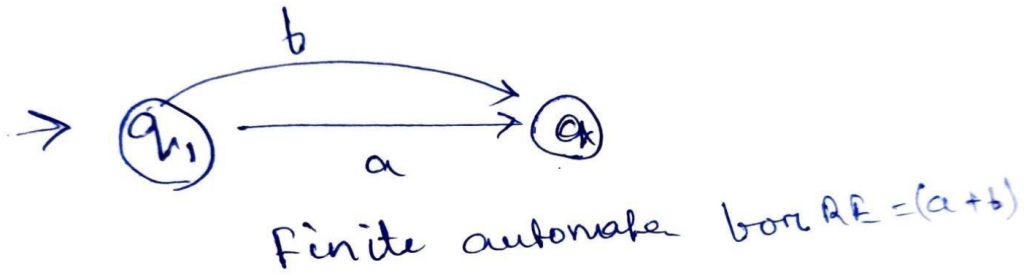
Case 1 – For a regular expression 'a', we can construct the following FA –



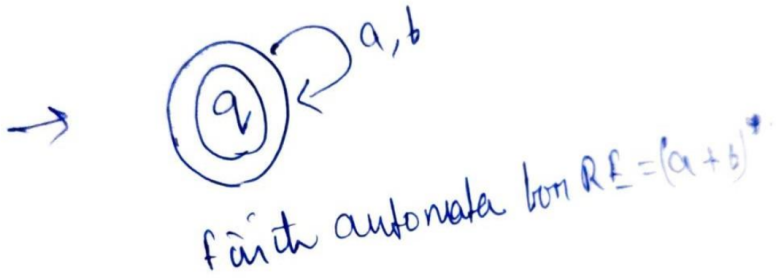
Case 2 – For a regular expression 'ab', we can construct the following FA –



Case 3 – For a regular expression (a+b), we can construct the following FA –



Case 4 – For a regular expression (a+b)*, we can construct the following FA –




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

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

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

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

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

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

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

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

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

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

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

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

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

def printTransitionTable(finite_automata):
```

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

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

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

```
PS C:\Users\harsh\codes> & C:/Users/harsh/AppData/Local/Programs/Python/Python39/python.exe c:/Users/harsh/codes/other/compiler/retonfa.py
Enter Regular Exp: a*b
State           Symbol                  Next state
q0              epsilon                 q1 q2
q1              a                       q3
q3              epsilon                 q1 q2
q2              epsilon                 q4
q4              b                       q5
```

RESULT - NFA transition table has been evaluated from the regular expression

Week 3
NFA to DFA

Aim : To convert a given NFA into its equivalent DFA

Algorithm :

1. Start

2. Get the input from the user

3. Set the only state in SDFA to “unmarked”.

4. while SDFA contains an unmarked state do: a. Let T be that unmarked state b. for each a in % do S = e-Closure(MoveNFA(T,a)) c. if S is not in SDFA already then, add S to SDFA (as an “unmarked” state) d. Set MoveDFA(T,a) to S

5. For each S in SDFA if any s & S is a final state in the NFA then, mark S an a final state in the DFA

6. Print the result.

7. Stop the program

```
import pandas as pd

nfa = {'0': {'a': ['0', '1'], 'b': ['0']}, '1': {'a': [], 'b': ['2']},
'2': {'a': [], 'b': []}}


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

nfa_final_state = '2'
```

```
new_states_list = []

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

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

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

    new_states_list.remove(new_states_list[0])

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

dfa_states_list = list(dfa.keys())
dfa_final_states = []
```

```
for x in dfa_states_list:
    for i in x:
        if i in nfa_final_state:
            dfa_final_states.append(x)
            break

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

```
NFA :-

{'0': {'a': ['0', '1'], 'b': ['0']}, '1': {'a': [], 'b': ['2']}, '2': {'a': [], 'b': []}}

Printing NFA table :-
        a    b
0  [0, 1]  [0]
1      []  [2]
2      []   []

DFA :-

{'0': {'a': '01', 'b': '0'}, '01': {'a': '01', 'b': '02'}, '02': {'a': '01', 'b': '0'}}

Printing DFA table :-
     a   b
0   01   0
01  01  02
02  01   0

Final states of the DFA are :  ['02']
```

RESULT - DFA transition table has been evaluated from the NFA

Week 4
Left Factoring and Left Recursion

Left Factoring

AIM: To develop a program to check and eliminate Left Factoring

ALGORITHM:

1. Start by initialization.

2. Get the number of productions and Grammar input from the user

3. Run a for loop till the number of productions and take input of the grammar.

4. Check if there are any common elements in before and after the '|'.

5. Store the common element in modified gram and store the other elements in newGram.

6. Store the other elements in X

7. Print A as the modified gram which will be of the format " A->SX" where S is the common element and X are the other elements which were attached with the common element.

8. If the entered Grammar is not a left factoring, then print Not aleft factoring.

9. End

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    long long int i, j, k, l, n, m = 9999999999, mini, ma = 0;
    string s[100], st, ch, sc = "", result, fs, maxi, rs = "";
    vector<string> ss;
    vector<string> sp;
    cin >> n;

    for (i = 1; i <= n; i++)
    {
        cin >> s[i];
    }
    for (i = 1; i <= n; i++)
    {
        st = s[i];
        sc = "";

        for (j = 0; j < st.length(); j++)
        {
            if (i == 1)
            {
                fs = st[0];
            }
            if (st[j] == '=')
            {
                l = j;
            }
        }
```

```
        if (i == 1)
        {

            for (k = l + 1; k < st.length(); k++)
            {
                if (st[k] == '|')
                {
                    ss.push_back(sc);
                    sc = "";
                }
                if (st[k] != '|')
                {
                    ch = st[k];
                    sc = sc + ch;
                }
            }
            ss.push_back(sc);
        }
        // cout<<sc<<endl;
    }

    for (k = 0; k < ss.size(); k++)
    {
        mini = ss[k].size();
        m = min(m, mini);
        maxi = ss[k];
        // cout<<ss[k]<<endl;
    }
    // cout<<maxi<<endl;

    for (k = 0; k < m; k++)
    {
        // cout<<ss[0][k]<<endl;
    }

    for (int i = 0; i < m; i++)
    {

        char current = ss[0][i];
```

```
    for (int j = 1; j < ss.size(); j++)
    {
        if (ss[j][i] != current)
        {
            break;
        }
        result.push_back(current);
    }
}

for (j = 0; j < ss.size(); j++)
{
    maxi = ss[j];
    // cout<<maxi<<result.length()<<endl;
    for (k = 0; k < maxi.length(); k++)
    {

        if (k >= result.length())
        {
            rs = rs + maxi[k];
        }

        // cout<<rs<<endl;
    }
    if (j != ss.size() - 1)
    {
        rs = rs + '|';
    }
}

cout << fs << "=" << result << fs << "'" << endl;
cout << fs << "'"
     << "=" << rs << endl;

for (i = 2; i <= n; i++)
{
    cout << s[i] << endl;
}
```

```
    return 0;
```

```
PS C:\Users\harsh\codes> cd "c:\Users\harsh\codes\other\compiler\" ; if ($?)
1
S=aSSbS
S=S'
S'=aSSbS
```

Left Recursion

AIM: To develop a program to check and eliminate Left Recursion

ALGORITHM:

1. Start by initialization.

2. Get the number of productions and Grammar input from the user

3. Run a for loop till the number of productions and take input of the grammar.

4. Check if the non-Terminal is present in the RHS and if yes print that it is a Left Recursion.

5. Remove the Left Recursion and Print the solution.

6. If the Left Recursion is non eliminable then print that it can’t be reduced.
7. If the entered Grammar is not a left recursion, then print Not a left recursion.

8. End

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
```

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

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

```
    return 0;
}
```

```
PS C:\Users\harsh\codes> cd "c:\Users\harsh\codes\other\compiler\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

Enter number of non terminals: 1

Enter non terminals:
Non terminal 1 : E

Enter '^' for null
Number of E productions: 2

One by one enter all E productions
RHS of production 1: E+T

RHS of production 2: T
1


New set of non-terminals: E E'

New set of productions:
E -> TE'
E' -> +TE'
E' -> ^
```

RESULT- The following grammar is left factored and left recursion is eliminated

Week 5
First and Follow and Predictive Parsing Table

AIM: To develop a program to find the First and Follow of Production Rules and create a Predictive Parsing Table

ALGORITHM:

1. Start by initialization.
2. Get the number of productions and Grammar input from the user
3. Run a for loop till the number of productions and take input of the grammar.
4. Check if the first and follow has been already calculated then put the production rule in the done list otherwise call the right function accordingly
5. In the FindFirst Function find the first of the production rule and keep calling it recursively unless all the elements have been iterated.
6. In the follow function check if the production rule is null otherwise check if the follow is calculated or not and then call the followfirst function to get the follow
7. In the FollowFirst Function find the first of the production rule and keep calling it recursively unless all the elements have been iterated.
8. After calculating the value of First and Follow store in calc_first and calc_follow.
9. For each production A –> α. (A tends to alpha) , Find First(α) and for each terminal in First(α), make entry A –> α in the table.

10. If First(α) contains ε (epsilon) as terminal than, find the Follow(A) and for each terminal in Follow(A), make entry A –> α in the table.

11. If the First(α) contains ε and Follow(A) contains $ as terminal, then make entry A –> α in the table for the $.

12. To construct the parsing table, we have two functions:

13. 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

First and Follow

```
import sys
sys.setrecursionlimit(60)

def first(string):
    first_ = set()
    if string in non_terminals:
        alternatives = productions_dict[string]

        for alternative in alternatives:
            first_2 = first(alternative)
            first_ = first_ |first_2

    elif string in terminals:
        first_ = {string}

    elif string=='' or string=='@':
        first_ = {'@'}

    else:
        first_2 = first(string[0])
        if '@' in first_2:
            i = 1
            while '@' in first_2:

                first_ = first_ | (first_2 - {'@'})
                if string[i:] in terminals:
                    first_ = first_ | {string[i:]}
                    break
```

```
                elif string[i:] == '':
                    first_ = first_ | {'@'}
                    break
                first_2 = first(string[i:])
                first_ = first_ | first_2 - {'@'}
                i += 1
        else:
            first_ = first_ | first_2


    return  first_


def follow(nT):
    follow_ = set()
    prods = productions_dict.items()
    if nT==starting_symbol:
        follow_ = follow_ | {'$'}
    for nt,rhs in prods:
        for alt in rhs:
            for char in alt:
                if char==nT:
                    following_str = alt[alt.index(char) + 1:]
                    if following_str=='':
                        if nt==nT:
                            continue
                        else:
                            follow_ = follow_ | follow(nt)
                    else:
                        follow_2 = first(following_str)
                        if '@' in follow_2:
                            follow_ = follow_ | follow_2-{'@'}
                            follow_ = follow_ | follow(nt)
                        else:
                            follow_ = follow_ | follow_2
    return follow_
```

```python
no_of_terminals=int(input("Enter no. of terminals: "))

terminals = []

print("Enter the terminals :")
for _ in range(no_of_terminals):
    terminals.append(input())

no_of_non_terminals=int(input("Enter no. of non terminals: "))

non_terminals = []

print("Enter the non terminals :")
for _ in range(no_of_non_terminals):
    non_terminals.append(input())

starting_symbol = input("Enter the starting symbol: ")

no_of_productions = int(input("Enter no of productions: "))

productions = []

print("Enter the productions:")
for _ in range(no_of_productions):
    productions.append(input())



productions_dict = {}

for nT in non_terminals:
    productions_dict[nT] = []



for production in productions:
```

```
    nonterm_to_prod = production.split("->")
    alternatives = nonterm_to_prod[1].split("/")
    for alternative in alternatives:
        productions_dict[nonterm_to_prod[0]].append(alternative)




FIRST = {}
FOLLOW = {}

for non_terminal in non_terminals:
    FIRST[non_terminal] = set()

for non_terminal in non_terminals:
    FOLLOW[non_terminal] = set()



for non_terminal in non_terminals:
    FIRST[non_terminal] = FIRST[non_terminal] | first(non_terminal)




FOLLOW[starting_symbol] = FOLLOW[starting_symbol] | {'$'}
for non_terminal in non_terminals:
    FOLLOW[non_terminal] = FOLLOW[non_terminal] | follow(non_terminal)



print("{: ^20}{: ^20}{: ^20}".format('Non Terminals','First','Follow'))
for non_terminal in non_terminals:
    print("{: ^20}{: ^20}{:
^20}".format(non_terminal,str(FIRST[non_terminal]),str(FOLLOW[non_terminal])))
```

```
PS C:\Users\harsh\codes> & C:/Users/harsh/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/harsh/codes/other/compiler/first and follow.py"
Enter no. of terminals: 5
Enter the terminals :
+
*
a
(
)
Enter no. of non terminals: 5
Enter the non terminals :
E
B
T
Y
F
Enter the starting symbol: E
Enter no of productions: 5
Enter the productions:
E->TB
B->+TB/@
T->FY
Y->*FY/@
F->a/(E)
   Non Terminals          First              Follow
         E              {'a', '('}         {'$', ')'}
         B              {'+', '@'}         {'$', ')'}
         T              {'a', '('}       {'$', ')', '+'}
         Y              {'*', '@'}       {'$', ')', '+'}
         F              {'a', '('}     {'$', '*', ')', '+'}
```

Predictive Parsing Table

```
gram = {
    "E":["E+T","T"],
    "T":["T*F","F"],
    "F":["(E)","i"],
    # "S":["CC"],
    # "C":["eC","d"],
}

def removeDirectLR(gramA, A):
    """gramA is dictonary"""
    temp = gramA[A]
    tempCr = []
    tempInCr = []
    for i in temp:
        if i[0] == A:
            #tempInCr.append(i[1:])
            tempInCr.append(i[1:]+[A+"'"])
        else:
            #tempCr.append(i)
            tempCr.append(i+[A+"'"])
    tempInCr.append(["e"])
    gramA[A] = tempCr
    gramA[A+"'"] = tempInCr
    return gramA


def checkForIndirect(gramA, a, ai):
    if ai not in gramA:
        return False
    if a == ai:
        return True
    for i in gramA[ai]:
        if i[0] == ai:
            return False
        if i[0] in gramA:
```

```
            return checkForIndirect(gramA, a, i[0])
    return False

def rep(gramA, A):
    temp = gramA[A]
    newTemp = []
    for i in temp:
        if checkForIndirect(gramA, A, i[0]):
            t = []
            for k in gramA[i[0]]:
                t=[]
                t+=k
                t+=i[1:]
                newTemp.append(t)

        else:
            newTemp.append(i)
    gramA[A] = newTemp
    return gramA

def rem(gram):
    c = 1
    conv = {}
    gramA = {}
    revconv = {}
    for j in gram:
        conv[j] = "A"+str(c)
        gramA["A"+str(c)] = []
        c+=1

    for i in gram:
        for j in gram[i]:
            temp = []
            for k in j:
                if k in conv:
                    temp.append(conv[k])
                else:
                    temp.append(k)
            gramA[conv[i]].append(temp)
```

```
#print(gramA)
for i in range(c-1,0,-1):
    ai = "A"+str(i)
    for j in range(0,i):
        aj = gramA[ai][0][0]
        if ai!=aj :
            if aj in gramA and checkForIndirect(gramA,ai,aj):
                gramA = rep(gramA, ai)

for i in range(1,c):
    ai = "A"+str(i)
    for j in gramA[ai]:
        if ai==j[0]:
            gramA = removeDirectLR(gramA, ai)
            break

op = {}
for i in gramA:
    a = str(i)
    for j in conv:
        a = a.replace(conv[j],j)
    revconv[i] = a

for i in gramA:
    l = []
    for j in gramA[i]:
        k = []
        for m in j:
            if m in revconv:
                k.append(m.replace(m,revconv[m]))
            else:
                k.append(m)
        l.append(k)
    op[revconv[i]] = l

return op
```

```
result = rem(gram)
terminals = []
for i in result:
    for j in result[i]:
        for k in j:
            if k not in result:
                terminals+=[k]
terminals = list(set(terminals))
#print(terminals)

def first(gram, term):
    a = []
    if term not in gram:
        return [term]
    for i in gram[term]:
        if i[0] not in gram:
            a.append(i[0])
        elif i[0] in gram:
            a += first(gram, i[0])
    return a

firsts = {}
for i in result:
    firsts[i] = first(result,i)
#   print(f'First({i}):',firsts[i])

def follow(gram, term):
    a = []
    for rule in gram:
        for i in gram[rule]:
            if term in i:
                temp = i
                indx = i.index(term)
                if indx+1!=len(i):
                    if i[-1] in firsts:
                        a+=firsts[i[-1]]
                    else:
                        a+=[i[-1]]
                else:
```

```
                    a+=["e"]
                if rule != term and "e" in a:
                    a+= follow(gram,rule)
    return a

follows = {}
for i in result:
    follows[i] = list(set(follow(result,i)))
    if "e" in follows[i]:
        follows[i].pop(follows[i].index("e"))
    follows[i]+=["$"]
#   print(f'Follow({i}):',follows[i])

resMod = {}
for i in result:
    l = []
    for j in result[i]:
        temp = ""
        for k in j:
            temp+=k
        l.append(temp)
    resMod[i] = l

# create predictive parsing table
tterm = list(terminals)
tterm.pop(tterm.index("e"))
tterm+=["d"]
pptable = {}
for i in result:
    for j in tterm:
        if j in firsts[i]:
            pptable[(i,j)]=resMod[i[0]][0]
        else:
            pptable[(i,j)]=""
    if "e" in firsts[i]:
        for j in tterm:
            if j in follows[i]:
                pptable[(i,j)]="e"
pptable[("F","i")] = "i"
```

```
toprint = f'{"": <10}'
for i in tterm:
    toprint+= f'|{i: <10}'
print(toprint)
for i in result:
    toprint = f'{i: <10}'
    for j in tterm:
        if pptable[(i,j)]!="":
            toprint+=f'|{i+"->"+pptable[(i,j)]: <10}'
        else:
            toprint+=f'|{pptable[(i,j)]: <10}'
    print(f'{"-":-<76}')
    print(toprint)
```

```
PS C:\Users\harsh\codes> & C:/Users/harsh/AppData/Local/Programs/Python/Python39/python.e
          |*         |)         |+         |i         |(         |d
----------------------------------------------------------------------------
E         |          |          |          |E->TE'    |E->TE'    |
----------------------------------------------------------------------------
T         |          |          |          |T->FT'    |T->FT'    |
----------------------------------------------------------------------------
F         |          |          |          |F->i      |F->(E)    |
----------------------------------------------------------------------------
E'        |          |E'->e     |E'->TE'   |          |          |
----------------------------------------------------------------------------
T'        |T'->FT'   |T'->e     |T'->e     |          |          |
```

RESULT- First and Follow and Predictive Parsing Table of the given grammar has been evaluated.

Week 6
Shift Reduce Parser

AIM: To develop a program to get the SHIFT REDUCE PARSING

ALGORITHM:

1. Starttheprogram.
2. Initialize the required variables.
3. Entertheinputsymbol.
4. Perform the following: for top-of-stack symbol, s, and next input symbol, a Shift x: (x is a STATE number)
5. Pusha,thenxonthetopofthestackAdvanceiptopointtothenext input symbol.
6. Reducey:(yisaPRODUCTIONnumber)Assumethattheproduction is of the form A→ß Pop 2 * |ß| symbols of the stack.
7. Atthispointthetopofthestackshouldbeastatenumber,says'.Push A, then goto of T[s',A] (a state number) on the top of the stack.
8. Output the production A→ß.
9. Printifstringisacceptedornot.
10. Stop the program

```
gram = {
    "S":["S+S","S*S","i"]
}
starting_terminal = "S"
inp = "i+i*i"

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

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

while True:
    action = True
    i = 0
    while i<len(gram[starting_terminal]):
        if gram[starting_terminal][i] in stack:
            stack = stack.replace(gram[starting_terminal][i],starting_terminal)
            print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Reduce
S->{gram[starting_terminal][i]}')
            i=-1
            action = False
        i+=1
    if len(inp)>1:
        stack+=inp[0]
        inp=inp[1:]
        print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Shift')
        action = False

    if inp == "$" and stack == ("$"+starting_terminal):
        print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Accepted')
        break

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

```
PS C:\Users\harsh\codes> & C:/Users/harsh/AppData/Local/Programs/Python/Python39/
Stack          |Input Buffer   |Parsing Action
-----------------------------------------------
$i             |+i*i           |Shift
$S             |+i*i           |Reduce S->i
$S+            |i*i            |Shift
$S+i           |*i             |Shift
$S+S           |*i             |Reduce S->i
$S             |*i             |Reduce S->S+S
$S*            |i              |Shift
$S*            |i              |Rejected
```

RESULT- Shift Reduce Parsing Table of the given grammar has been evaluated.

Week 7
Computation of Leading and Trailing

AIM : Implementation of Leading and Trailing Program

ALGORITHM :

- 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)
  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.
- 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.

```
#include <iostream>
#include <string.h>
#include <conio.h>

using namespace std;
int nt, t, top = 0;
char s[50], NT[10], T[10], st[50], l[10][10], tr[50][50];
int searchnt(char a)
{
    int count = -1, i;
    for (i = 0; i < nt; i++)
    {
        if (NT[i] == a)
```

```
            return i;
    }
    return count;
}
int searchter(char a)
{
    int count = -1, i;
    for (i = 0; i < t; i++)
    {
        if (T[i] == a)
            return i;
    }
    return count;
}
void push(char a)
{
    s[top] = a;
    top++;
}
char pop()
{
    top--;
    return s[top];
}
void installl(int a, int b)

{
    if (l[a][b] == 'f')
    {
        l[a][b] = 't';
        push(T[b]);
        push(NT[a]);
    }
}
void installt(int a, int b)
{
    if (tr[a][b] == 'f')
    {
        tr[a][b] = 't';
```

```
        push(T[b]);
        push(NT[a]);
    }
}

int main()
{
    int i, s, k, j, n;
    char pr[30][30], b, c;

    cout << "Enter the no of productions:";
    cin >> n;
    cout << "Enter the productions one by one\n";
    for (i = 0; i < n; i++)
        cin >> pr[i];
    nt = 0;
    t = 0;
    for (i = 0; i < n; i++)
    {
        if ((searchnt(pr[i][0])) == -1)
            NT[nt++] = pr[i][0];
    }
    for (i = 0; i < n; i++)
    {
        for (j = 3; j < strlen(pr[i]); j++)
        {
            if (searchnt(pr[i][j]) == -1)
            {
                if (searchter(pr[i][j]) == -1)
                    T[t++] = pr[i][j];
            }
        }
    }
    for (i = 0; i < nt; i++)
    {
        for (j = 0; j < t; j++)
            l[i][j] = 'f';
    }
    for (i = 0; i < nt; i++)
```

```
{
    for (j = 0; j < t; j++)

        tr[i][j] = 'f';
}
for (i = 0; i < nt; i++)
{
    for (j = 0; j < n; j++)
    {
        if (NT[(searchnt(pr[j][0]))] == NT[i])
        {
            if (searchter(pr[j][3]) != -1)
                installl(searchnt(pr[j][0]), searchter(pr[j][3]));
            else
            {
                for (k = 3; k < strlen(pr[j]); k++)
                {
                    if (searchnt(pr[j][k]) == -1)
                    {
                        installl(searchnt(pr[j][0]), searchter(pr[j][k]));
                        break;
                    }
                }
            }
        }
    }
}
while (top != 0)
{
    b = pop();
    c = pop();
    for (s = 0; s < n; s++)
    {
        if (pr[s][3] == b)
            installl(searchnt(pr[s][0]), searchter(c));
    }
}
for (i = 0; i < nt; i++)
{
```

```
        cout << "Leading[" << NT[i] << "]"
             << "\t{";
        for (j = 0; j < t; j++)
        {
            if (l[i][j] == 't')
                cout << T[j] << ",";
        }
        cout << "}\n";
    }

    top = 0;
    for (i = 0; i < nt; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (NT[searchnt(pr[j][0])] == NT[i])
            {
                if (searchter(pr[j][strlen(pr[j]) - 1]) != -1)
                    installt(searchnt(pr[j][0]), searchter(pr[j][strlen(pr[j]) -
1]));
                else
                {
                    for (k = (strlen(pr[j]) - 1); k >= 3; k--)
                    {
                        if (searchnt(pr[j][k]) == -1)
                        {
                            installt(searchnt(pr[j][0]), searchter(pr[j][k]));
                            break;
                        }
                    }
                }
            }
        }
    }
    while (top != 0)
    {
        b = pop();
        c = pop();
        for (s = 0; s < n; s++)
```

```
        {
            if (pr[s][3] == b)
                installt(searchnt(pr[s][0]), searchter(c));
        }
    }
    for (i = 0; i < nt; i++)
    {
        cout << "Trailing[" << NT[i] << "]"
             << "\t{";
        for (j = 0; j < t; j++)
        {
            if (tr[i][j] == 't')
                cout << T[j] << ",";
        }
        cout << "}\n";
    }
    getch();
}
```

```
PS C:\Users\harsh\codes> cd "c:\Users\harsh\codes\other\compiler\" ; if ($?) { g++ leading.cpp -o leading } ; if ($?) { .\leading }
Enter the no of productions:2
Enter the productions one by one
S->(L)/a
S->L,S/S
Leading[S]      {(,L,}
Trailing[S]     {/,a,}
```

RESULT- Thus Leading and Trailing functions are evaluated for the given grammar

Week 8
LR(0) Parser

AIM: To develop a program to get the LR(0) item sets and parsing table

ALGORITHM:

1. Start.

2. Create structure for production with LHS and RHS.

3. Open file and read input from file.

4. Build state 0 from extra grammar Law S' -> S $ that is all start symbol of grammar and one Dot (.) before S symbol.

5. If Dot symbol is before a non-terminal, add grammar laws that this non-terminal is in Left Hand Side of that Law and set Dot in before of first part of Right Hand Side.

6. If state exists (a state with this Laws and same Dot position), use that instead.

7. Now find set of terminals and non-terminals in which Dot exist in before.

8. If step 7 Set is non-empty go to 9, else go to 10.

9. For each terminal/non-terminal in set step 7 create new state by using all grammar law that Dot position is before of that terminal/non-terminal in reference state by increasing Dot point to next part in Right Hand Side of that laws.

10. Go to step 5.

11. End of state building. 12. Create the parsing table 13. Display the output.

14. End.

```
gram = {
    "S":["CC"],
    "C":["aC","d"]
}
start = "S"
terms = ["a","d","$"]

non_terms = []
for i in gram:
    non_terms.append(i)
gram["S'"]= [start]


new_row = {}
for i in terms+non_terms:
    new_row[i]=""


non_terms += ["S'"]
# each row in state table will be dictionary {nonterms ,term,$}
stateTable = []
# I = [(terminal, closure)]
# I = [("S","A.A")]

def Closure(term, I):
    if term in non_terms:
        for i in gram[term]:
            I+=[(term,"."+i)]
    I = list(set(I))
    for i in I:
        # print("." != i[1][-1],i[1][i[1].index(".")+1])
        if "." != i[1][-1] and i[1][i[1].index(".")+1] in non_terms and
i[1][i[1].index(".")+1] != term:
            I += Closure(i[1][i[1].index(".")+1], [])
```

```
    return I

Is = []
Is+=set(Closure("S'", []))


countI = 0
omegaList = [set(Is)]
while countI<len(omegaList):
    newrow = dict(new_row)
    vars_in_I = []
    Is = omegaList[countI]
    countI+=1
    for i in Is:
        if i[1][-1]!=".":
            indx = i[1].index(".")
            vars_in_I+=[i[1][indx+1]]
    vars_in_I = list(set(vars_in_I))
    # print(vars_in_I)
    for i in vars_in_I:
        In = []
        for j in Is:
            if "."+i in j[1]:
                rep = j[1].replace("."+i,i+".")
                In+=[(j[0],rep)]
        if (In[0][1][-1]!="."):
            temp = set(Closure(i,In))
            if temp not in omegaList:
                omegaList.append(temp)
            if i in non_terms:
                newrow[i] = str(omegaList.index(temp))
            else:
                newrow[i] = "s"+str(omegaList.index(temp))
            print(f'Goto(I{countI-1},{i}):{temp} That is
I{omegaList.index(temp)}')
        else:
            temp = set(In)
            if temp not in omegaList:
                omegaList.append(temp)
```

```
            if i in non_terms:
                newrow[i] = str(omegaList.index(temp))
            else:
                newrow[i] = "s"+str(omegaList.index(temp))
            print(f'Goto(I{countI-1},{i}):{temp} That is
I{omegaList.index(temp)}')

    stateTable.append(newrow)
print("\n\nList of I's\n")
for i in omegaList:
    if(omegaList.index(i)==0):
        print("Here 2")
    print(f'I{omegaList.index(i)}: {i}')


#populate replace elements in state Table
I0 = []
for i in list(omegaList[0]):
    I0 += [i[1].replace(".","")]
print(I0)

print("HERE")

for i in omegaList:
    for j in i:
        if "." in j[1][-1]:
            if j[1][-2]=="S":
                stateTable[omegaList.index(i)]["$"] = "Accept"
                break
            for k in terms:
                stateTable[omegaList.index(i)][k] =
"r"+str(I0.index(j[1].replace(".","")))
print("\nStateTable")

print(f'{" ": <9}',end="")
for i in new_row:
    print(f'|{i: <11}',end="")

print(f'\n{"-":-<66}')
```

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

```
PS C:\Users\harsh\codes> & C:/Users/harsh/AppData/Local/Programs/Python/Python39/python.exe c:/Users/harsh/codes/other/compiler/lr0.py
Goto(I0,d):{('C', 'd.')} That is I1
Goto(I0,C):{('C', '.d'), ('S', 'C.C'), ('C', '.aC')} That is I2
Goto(I0,S):{("S'", 'S.')} That is I3
Goto(I0,a):{('C', '.d'), ('C', '.aC'), ('C', 'a.C')} That is I4
Goto(I2,d):{('C', 'd.')} That is I1
Goto(I2,C):{('S', 'CC.')} That is I5
Goto(I2,a):{('C', '.d'), ('C', '.aC'), ('C', 'a.C')} That is I4
Goto(I4,d):{('C', 'd.')} That is I1
Goto(I4,C):{('C', 'aC.')} That is I6
Goto(I4,a):{('C', '.d'), ('C', '.aC'), ('C', 'a.C')} That is I4


List of I's

Here 2
I0: {("S'", '.S'), ('C', '.aC'), ('C', '.d'), ('S', '.CC')}
I1: {('C', 'd.')}
I2: {('C', '.d'), ('S', 'C.C'), ('C', '.aC')}
I3: {("S'", 'S.')}
I4: {('C', '.d'), ('C', '.aC'), ('C', 'a.C')}
I5: {('S', 'CC.')}
I6: {('C', 'aC.')}
['S', 'aC', 'd', 'CC']
HERE

StateTable
        |a          |d          |$          |S          |C
------------------------------------------------------------------
I(0)    |s4         |s1         |           |3          |2
I(1)    |r2         |r2         |r2         |           |
I(2)    |s4         |s1         |           |           |5
I(3)    |           |           |Accept     |           |
I(4)    |s4         |s1         |           |           |6
I(5)    |r3         |r3         |r3         |           |
```

Result- LR(0) Parsing Table is computed for the given grammar.

Week 9
Prefix Postfix and Three Address Code

AIM: To develop a program to get the prefix, postfix and Three Address code

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 understoodby the machine).

The benefits of using machine independent intermediate code are:

Because of the machine independent intermediate code, portability will be enhanced. For ex, suppose, 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 n 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 ofthe expression (a - b)* (c +d) + (a - b) is : ab -cd + *ab -+.

1.

2. Three-Address Code –

A statement involving no more than three references(two for operands and one for result) is known as three address statement. A sequence of three address statements is known as three address code. Three address statement is of the form x = y 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 statement.

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

T 1 =b * c

T2=a+T1

T3=T2+d

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

Read the Prefix expression in reverse order (from right to left)
If the symbol is an operand, then push it onto the Stack
If the symbol is an operator, then pop two operands from the Stack
Create a string by concatenating the two operands and the operator after them.
string = operand1 + operand2 + operator

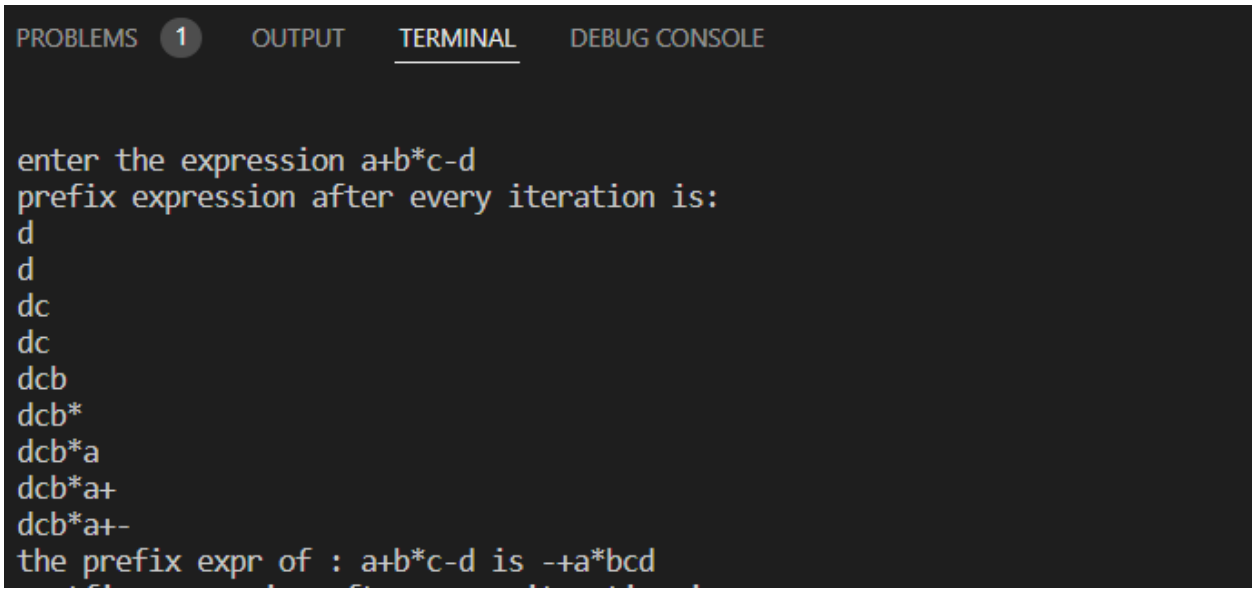
And push the resultant string back to Stack
Repeat the above steps until end of Prefix expression.

```
class infix_to_prefix:
    precedence={'^':5,'*':4,'/':4,'+':3,'-':3,'(':2,')':1}
    def __init__(self):
        self.items=[]
        self.size=-1
    def push(self,value):
        self.items.append(value)
        self.size+=1
    def pop(self):
        if self.isempty():
            return 0
        else:
            self.size-=1
            return self.items.pop()
    def isempty(self):
        if(self.size==-1):
            return True
        else:
            return False
    def seek(self):
        if self.isempty():
            return False
        else:
            return self.items[self.size]
    def is0perand(self,i):
        if i.isalpha() or i in '1234567890':
            return True
        else:
            return False
    def reverse(self,expr):
        rev=""
        for i in expr:
```

```
            if i == '(':
                i=')'
            elif i == ')':
                i='('
            rev=i+rev
        return rev
    def infixtoprefix (self,expr):
        prefix=""
        print('prefix expression after every iteration is:')
        for i in expr:
            if(len(expr)%2==0):
                print("Incorrect infix expr")
                return False
            elif(self.is0perand(i)):
                prefix +=i
            elif(i in '+-*/^'):
                while(len(self.items)and self.precedence[i] <
self.precedence[self.seek()]):
                    prefix+=self.pop()
                self.push(i)
            elif i == '(':
                self.push(i)
            elif i == ')':
                o=self.pop()
                while o!='(':
                    prefix +=o
                    o=self.pop()
            print(prefix)
                #end of for
        while len(self.items):
            if(self.seek()=='('):
                self.pop()
            else:
                prefix+=self.pop()
                print(prefix)
        return prefix

    def infixtopostfix (self,expr):
        postfix=""
```

```
        print('postfix expression after every iteration is:')
        for i in expr:
            if(len(expr)%2==0):
                print("Incorrect infix expr")
                return False
            elif(self.isOperand(i)):
                postfix +=i
            elif(i in '+-*/^'):
                while(len(self.items)and
self.precedence[i]<=self.precedence[self.seek()]):
                    postfix+=self.pop()
                self.push(i)
            elif i == '(':
                self.push(i)
            elif i == ')':
                o=self.pop()
                while o!='(':
                    postfix +=o
                    o=self.pop()
            print(postfix)
                #end of for
        while len(self.items):
            if(self.seek()=='('):
                self.pop()
            else:
                postfix+=self.pop()
        return postfix

s=infix_to_prefix()
expr=input('enter the expression ')
rev=""
rev=s.reverse(expr)
result=s.infixtoprefix(rev)

if (result!=False):

    prefix=s.reverse(result)
    print("the prefix expr of :",expr,"is",prefix)
```

```
result=s.infixtopostfix(expr)
if (result!=False):
    print("the postfix expr of :",expr,"is",result)
```



RESULT- Hence the program Prefix, Postfix and Three Address Code was created successfully

Week 10
Quadruple, Triple and Indirect Triple

Aim : To implement quadruple , triple and indirect triple for a given expression Algorithm

Algorithm :

In quadruples representation, each instruction is splitted into the following 4 different fields-

op, arg1, arg2, result

Here-

In triples representation,

- References to the instructions are made.
- Temporary variables are not used.

In indirect-triples ,

- This representation is an enhancement over triples representation.
- It uses an additional instruction array to list the pointers to the triples in the desired order.
- Thus, instead of position, pointers are used to store the results.
- It allows the optimizers to easily reposition the sub-expression for producing the optimized code.

In indirect-triples ,
● This representation is an enhancement over triples representation.
● It uses an additional instruction array to list the pointers to the triples in the desired order.
● Thus, instead of position, pointers are used to store the results.
● It allows the optimizers to easily reposition the sub-expression for producing the optimized code.

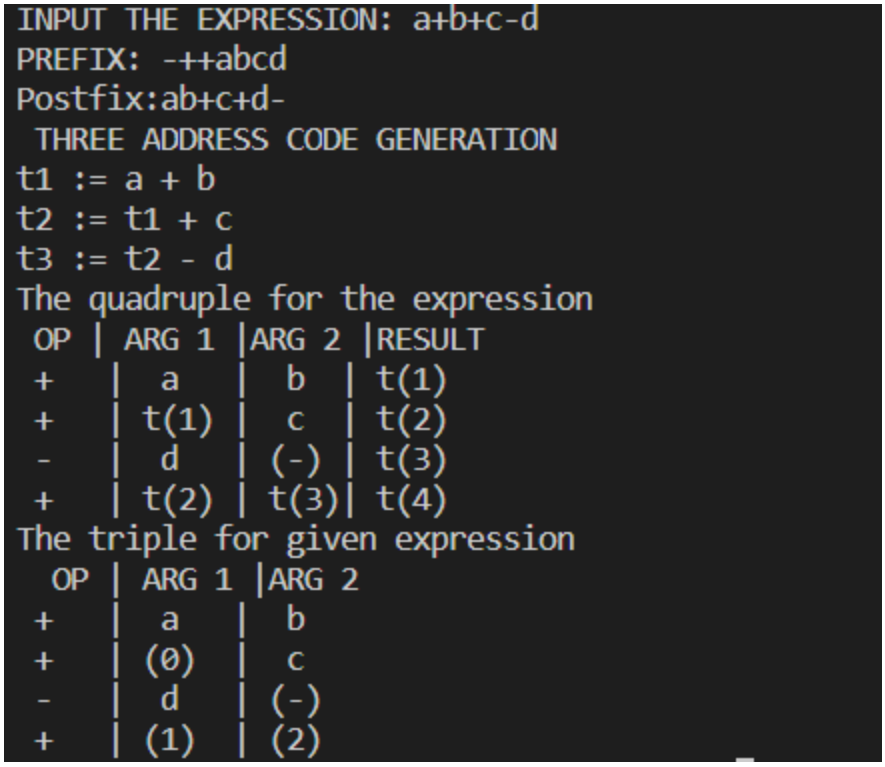
```
OPERATORS = set(['+', '-', '*', '/', '(', ')'])
PRI = {'+':1, '-':1, '*':2, '/':2}

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

```
    output = ''
    for ch in formula:
        if ch not in OPERATORS:
            output += ch
        elif ch == '(':
            stack.append('(')
        elif ch == ')':
            while stack and stack[-1] != '(':
                output += stack.pop()
            stack.pop() # pop '('
        else:
            while stack and stack[-1] != '(' and PRI[ch] <= PRI[stack[-1]]:
                output += stack.pop()
            stack.append(ch)
    # leftover
    while stack:
        output += stack.pop()
    print(f"Postfix:{output}")
    return output

def infix_to_prefix(formula):
    op_stack = []
    exp_stack = []
    for ch in formula:
        if not ch in OPERATORS:
            exp_stack.append(ch)
        elif ch == '(':
            op_stack.append(ch)
        elif ch == ')':
            while op_stack[-1] != '(':
                op = op_stack.pop()
                a = exp_stack.pop()
                b = exp_stack.pop()
                exp_stack.append( op+b+a )
            op_stack.pop() # pop '('
        else:
            while op_stack and op_stack[-1] != '(' and PRI[ch] <=
PRI[op_stack[-1]]:
                op = op_stack.pop()
```

```
                a = exp_stack.pop()
                b = exp_stack.pop()
                exp_stack.append( op+b+a )
            op_stack.append(ch)

    # leftover
    while op_stack:
        op = op_stack.pop()
        a = exp_stack.pop()
        b = exp_stack.pop()
        exp_stack.append( op+b+a )
    print(f'PREFIX: {exp_stack[-1]}')
    return exp_stack[-1]

def generate3AC(pos):
    print(" THREE ADDRESS CODE GENERATION ")
    exp_stack = []
    t = 1

    for i in pos:
        if i not in OPERATORS:
            exp_stack.append(i)
        else:
            print(f't{t} := {exp_stack[-2]} {i} {exp_stack[-1]}')
            exp_stack=exp_stack[:-2]
            exp_stack.append(f't{t}')
            t+=1

expres = input("INPUT THE EXPRESSION: ")
pre = infix_to_prefix(expres)
pos = infix_to_postfix(expres)
generate3AC(pos)


def Quadruple(pos):
    stack = []
    #op = []
    x = 1
    for i in pos:
```

```
        if i not in OPERATORS:
            stack.append(i)
        elif i == '-':
            op1 = stack.pop()
            stack.append("t(%s)" % x)
            print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format(i, op1, "(-)", "
t(%s)" % x))
            x = x + 1
            if stack != []:
                op2 = stack.pop()
                op1 = stack.pop()
                print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format("+", op1, op2,
" t(%s)" % x))
                stack.append("t(%s)" % x)
                x = x + 1
        elif i == '=':
            op2 = stack.pop()
            op1 = stack.pop()
            print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format(i, op2, "(-)",
op1))
        else:
            op1 = stack.pop()
            op2 = stack.pop()
            print("{0:^4s} | {1:^4s} | {2:^4s}|{3:4s}".format(i, op2, op1, "
t(%s)" % x))
            stack.append("t(%s)" % x)
            x = x + 1


print("The quadruple for the expression ")
print(" OP | ARG 1 |ARG 2 |RESULT  ")
Quadruple(pos)


def Triple(pos):
    stack = []
    #op = []
    x = 0
    for i in pos:
```

```
        if i not in OPERATORS:
            stack.append(i)
        elif i == '-':
            op1 = stack.pop()
            stack.append("(%s)" % x)
            print("{0:^4s} | {1:^4s} | {2:^4s}".format(i, op1, "(-)"))
            x = x + 1
            if stack != []:
                op2 = stack.pop()
                op1 = stack.pop()
                print("{0:^4s} | {1:^4s} | {2:^4s}".format("+", op1, op2))
                stack.append("(%s)" % x)
                x = x + 1
        elif i == '=':
            op2 = stack.pop()
            op1 = stack.pop()
            print("{0:^4s} | {1:^4s} | {2:^4s}".format(i, op1, op2))
        else:
            op1 = stack.pop()
            if stack != []:
                op2 = stack.pop()
                print("{0:^4s} | {1:^4s} | {2:^4s}".format(i, op2, op1))
                stack.append("(%s)" % x)
                x = x + 1


print("The triple for given expression")
print("  OP | ARG 1 |ARG 2  ")
Triple(pos)
```



Result : The code for different forms of intermediate code generation is successfully implemented.

Week 11
Implementation of ICG

Aim: To write a program to implement intermediate code generation

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

```c
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
void small();
void dove(int i);
int p[5] = {0, 1, 2, 3, 4}, c = 1, i, k, l, m, pi;
char sw[5] = {'=', '-', '+', '/', '*'}, j[20], a[5], b[5], ch[2];
void main()
{
    printf("Enter the expression:");
    scanf("%s", j);
    printf("\tThe Intermediate code is:\n");
    small();
}
void dove(int i)
{
    a[0] = b[0] = '\0';
```

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

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

```
PROBLEMS 1   OUTPUT   TERMINAL   DEBUG CONSOLE

PS C:\Users\harsh\codes\other\compiler> cd "c:\Users\har
Enter the expression:a=b+c-d
        The Intermediate code is:
        t1=b+c
        t2=t1-d
        a=t2
```

Result: The program was successfully compiled and run.

Week 12
Implementation OF DAG

Aim: A program to implement DAG

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

```c
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
void small();
void dove(int i);
int p[5] = {0, 1, 2, 3, 4}, c = 1, i, k, l, m, pi;
char sw[5] = {'=', '-', '+', '/', '*'}, j[20], a[5], b[5], ch[2];
void main()
{
    printf("Enter the expression:");
    scanf("%s", j);
    printf("\tThe Intermediate code is:\n");
    small();
}
void dove(int i)
{
    a[0] = b[0] = '\0';
    if (!isdigit(j[i + 2]) && !isdigit(j[i - 2]))
```

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

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

```
PS C:\Users\harsh\codes\other\compiler> cd "c:\Users\harsh\codes\other\compiler\"
Enter the expression:t=a+b
        The Intermediate code is:
        t=t0    t1=t0+b
PS C:\Users\harsh\codes\other\compiler>
```

Result: The program was successfully compiled and run.

Week 13
Implementation OF VARIOUS STORAGE ALLOCATION TECHNIQUES ALGORITHMS

Aim: To implement various storage allocation techniques algorithms

Algorithm:

Static storage allocation
● In static allocation, names are bound to storage locations.
● If memory is created at compile time then the memory will be created in static area and only once.
● Static allocation supports the dynamic data structure that means memory is created only at compile time and deallocated after program completion.
● The drawback with static storage allocation is that the size and position of data objects should be known at compile time.
● Another drawback is restriction of the recursion procedure.

Stack Storage Allocation
● In static storage allocation, storage is organized as a stack.
● An activation record is pushed into the stack when activation begins and it is popped when the activation ends.
● Activation record contains the locals so that they are bound to fresh storage 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.

1. Stack

```c
#include <conio.h>
#include<stdio.h>
int i, stk[100], top = -1, n;
void show()
{
    for (i = 0; i <= top; i++)
        printf("%d\t", stk[i]);
}
void push()
{
    int item;
    if (top == n - 1)
        printf("\nStack is full.");
    else
    {
        printf("\nEnter the item: ");
        scanf("%d", &item);
        stk[++top] = item;
    }
}
void pop()
{
    if (top == -1)
        printf("Stack is empty.");
    else
    {
        printf("%d is popped.", stk[top]);
        top--;
    }
}
int main()
{
    int i, op;
    printf("Enter the size of the stack: ");
    scanf("%d", &n);
    do
```

```
    {
        printf("\n1 : Push");
        printf("\n2 : Pop");
        printf("\n3 : Display");
        printf("\n4 : Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &op);
        switch (op)
        {
        case 1:
            push();
            break;
        case 2:
            pop();
            break;
        case 3:
            show();
            break;
        }
    } while (op != 4);
    getch();
}
```

```
PS C:\Users\harsh\codes\other\compiler> cd "c:\Users\harsh\codes\other\compiler\" ; if ($?)
Enter the size of the stack: 4

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: 1

Enter the item: 7

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

Enter the item: 8

1 : Push
2 : Pop
3 : Display
4 : Exit
Enter your choice: 3
5       7       8
1 : Push
2 : Pop
3 : Display
4 : Exit
Enter your choice: 2
8 is popped.
1 : Push
2 : Pop
3 : Display
4 : Exit
Enter your choice: 3
5       7
```

2. Heap

```cpp
#include <iostream>
#include <cstdlib>
#include <vector>
#include <iterator>
using namespace std;
/*
```

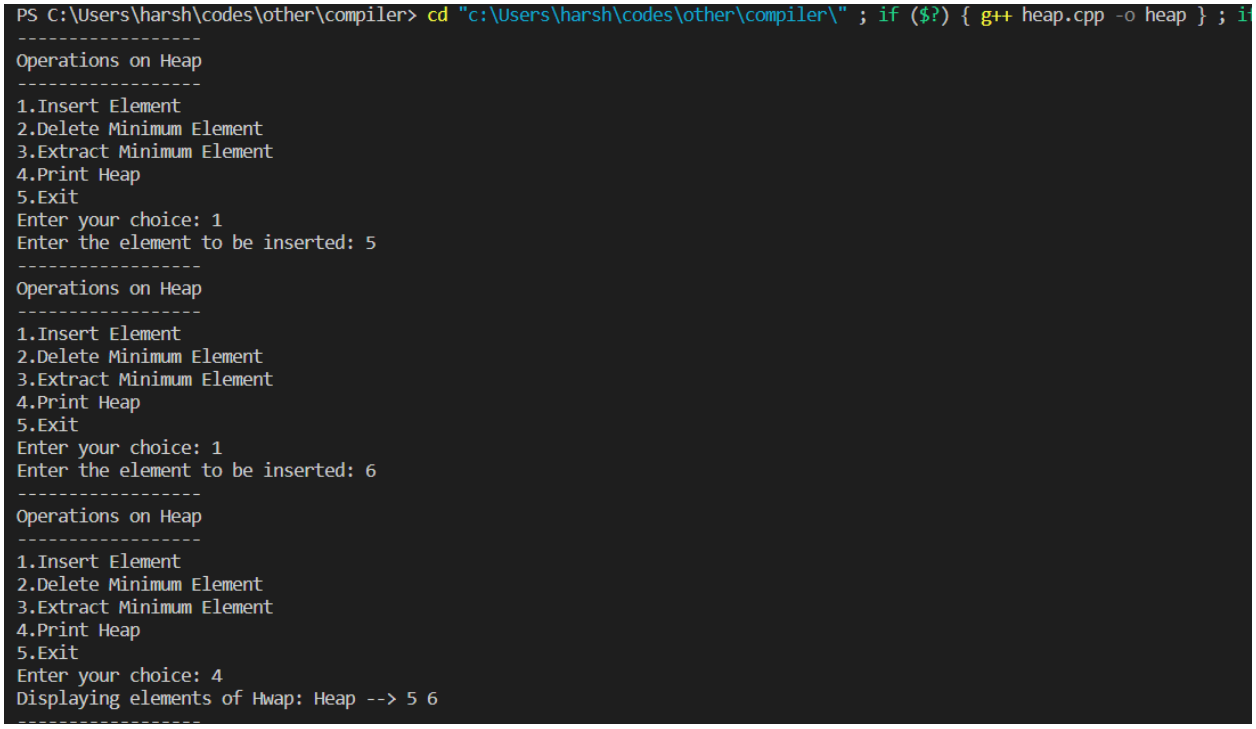
```
 * Class Declaration
 */
class Heap
{
private:
    vector<int> heap;
    int left(int parent);
    int right(int parent);
    int parent(int child);
    void heapifyup(int index);
    void heapifydown(int index);

public:
    Heap()
    {
    }
    void Insert(int element);
    void DeleteMin();
    int ExtractMin();
    void DisplayHeap();
    int Size();
};
/*
 * Return Heap Size
 */
int Heap::Size()
{
    return heap.size();
}
/*
 * Insert Element into a Heap
 */
void Heap::Insert(int element)
{
    heap.push_back(element);
    heapifyup(heap.size() - 1);
}
/*
 * Delete Minimum Element
```

```
 */
void Heap::DeleteMin()
{
    if (heap.size() == 0)
    {
        cout << "Heap is Empty" << endl;
        return;
    }
    heap[0] = heap.at(heap.size() - 1);
    heap.pop_back();
    heapifydown(0);
    cout << "Element Deleted" << endl;
}
/*
 * Extract Minimum Element
 */
int Heap::ExtractMin()
{
    if (heap.size() == 0)
    {
        return -1;
    }
    else
        return heap.front();
}
/*
 * Display Heap
 */
void Heap::DisplayHeap()
{
    vector<int>::iterator pos = heap.begin();
    cout << "Heap --> ";
    while (pos != heap.end())
    {
        cout << *pos << " ";
        pos++;
    }
    cout << endl;
}
```

```
/*
 * Return Left Child
 */
int Heap::left(int parent)
{
    int l = 2 * parent + 1;
    if (l < heap.size())
        return l;
    else
        return -1;
}
/*
 * Return Right Child
 */
int Heap::right(int parent)
{
    int r = 2 * parent + 2;
    if (r < heap.size())
        return r;
    else
        return -1;
}
/*
 * Return Parent
 */
int Heap::parent(int child)
{
    int p = (child - 1) / 2;
    if (child == 0)
        return -1;
    else
        return p;
}
/*
 * Heapify- Maintain Heap Structure bottom up
 */
void Heap::heapifyup(int in)
{
    if (in >= 0 && parent(in) >= 0 && heap[parent(in)] > heap[in])
```

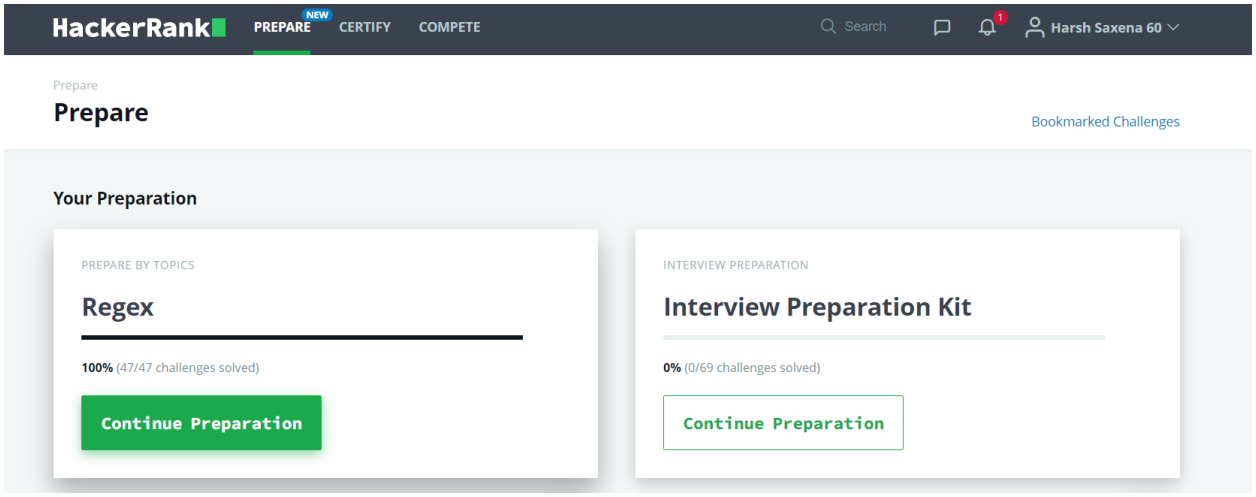
```
    {
        int temp = heap[in];
        heap[in] = heap[parent(in)];
        heap[parent(in)] = temp;
        heapifyup(parent(in));
    }
}
/*
 * Heapify- Maintain Heap Structure top down
 */
void Heap::heapifydown(int in)
{
    int child = left(in);
    int child1 = right(in);
    if (child >= 0 && child1 >= 0 && heap[child] > heap[child1])
    {
        child = child1;
    }
    if (child > 0)
    {
        int temp = heap[in];
        heap[in] = heap[child];
        heap[child] = temp;
        heapifydown(child);
    }
}
/*
 * Main Contains Menu
 */
int main()
{
    Heap h;
    while (1)
    {
        cout << "------------------" << endl;
        cout << "Operations on Heap" << endl;
        cout << "------------------" << endl;
        cout << "1.Insert Element" << endl;
        cout << "2.Delete Minimum Element" << endl;
```

```
        cout << "3.Extract Minimum Element" << endl;
        cout << "4.Print Heap" << endl;
        cout << "5.Exit" << endl;
        int choice, element;
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            cout << "Enter the element to be inserted: ";
            cin >> element;
            h.Insert(element);
            break;
        case 2:
            h.DeleteMin();
            break;
        case 3:
            cout << "Minimum Element: ";
            if (h.ExtractMin() == -1)
            {
                cout << "Heap is Empty" << endl;
            }
            else
                cout << "Minimum Element: " << h.ExtractMin() << endl;
            break;
        case 4:
            cout << "Displaying elements of Hwap: ";
            h.DisplayHeap();
            break;
        case 5:
            exit(1);
        default:
            cout << "Enter Correct Choice" << endl;
        }
    }
    return 0;
}
```

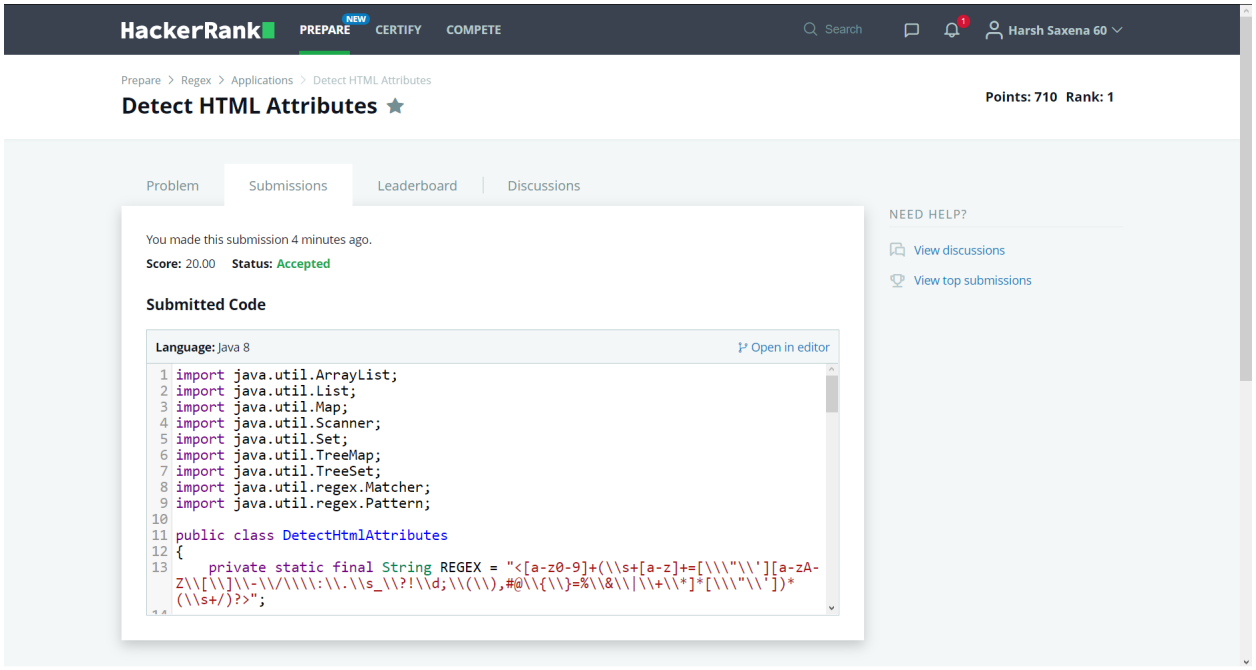


Result : Different Algorithms for various storage allocation algorithms were implemented successfully

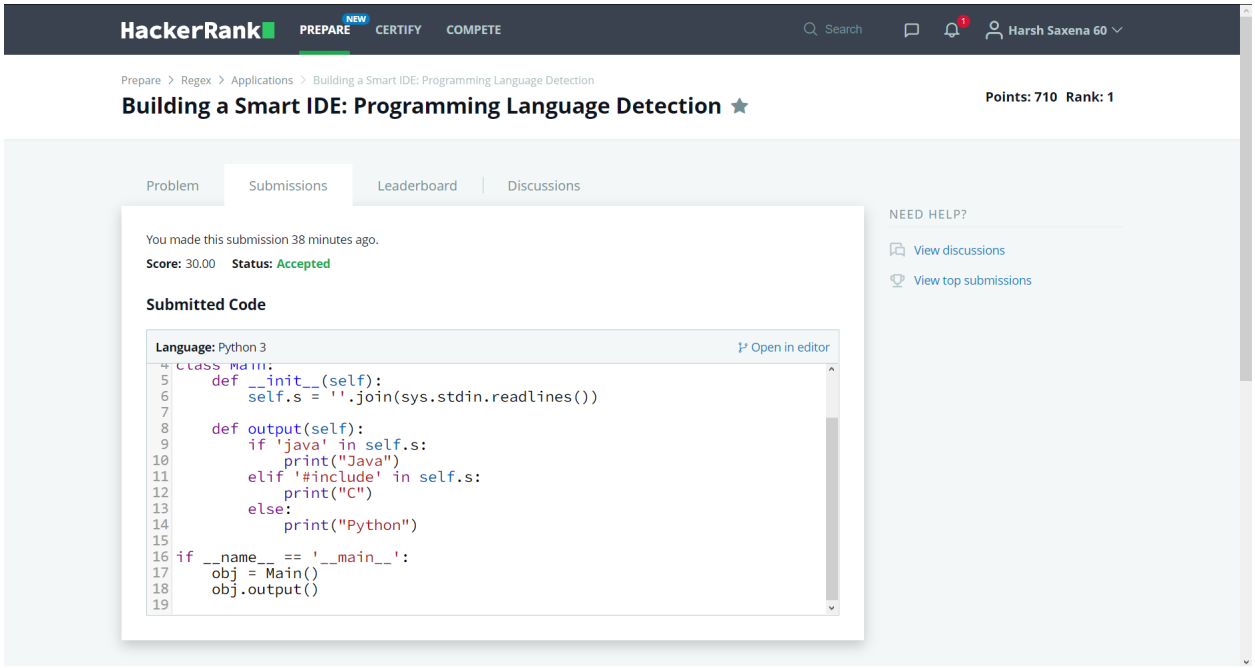
Hackerrank Regex Questions



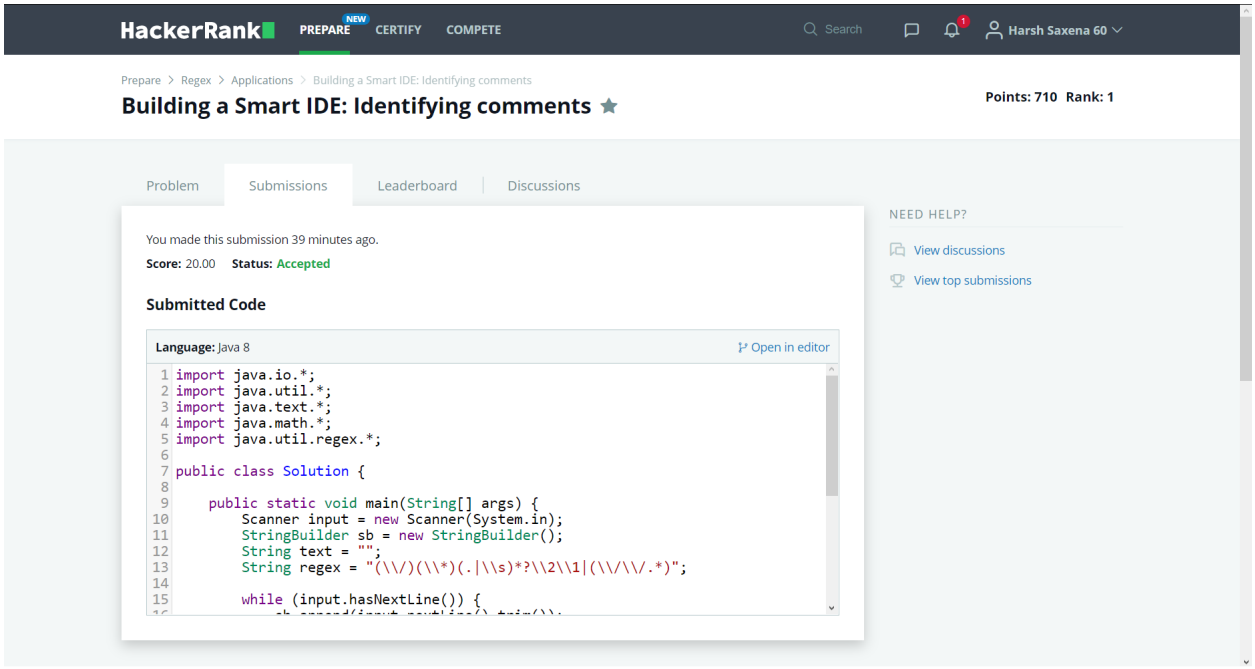
1.



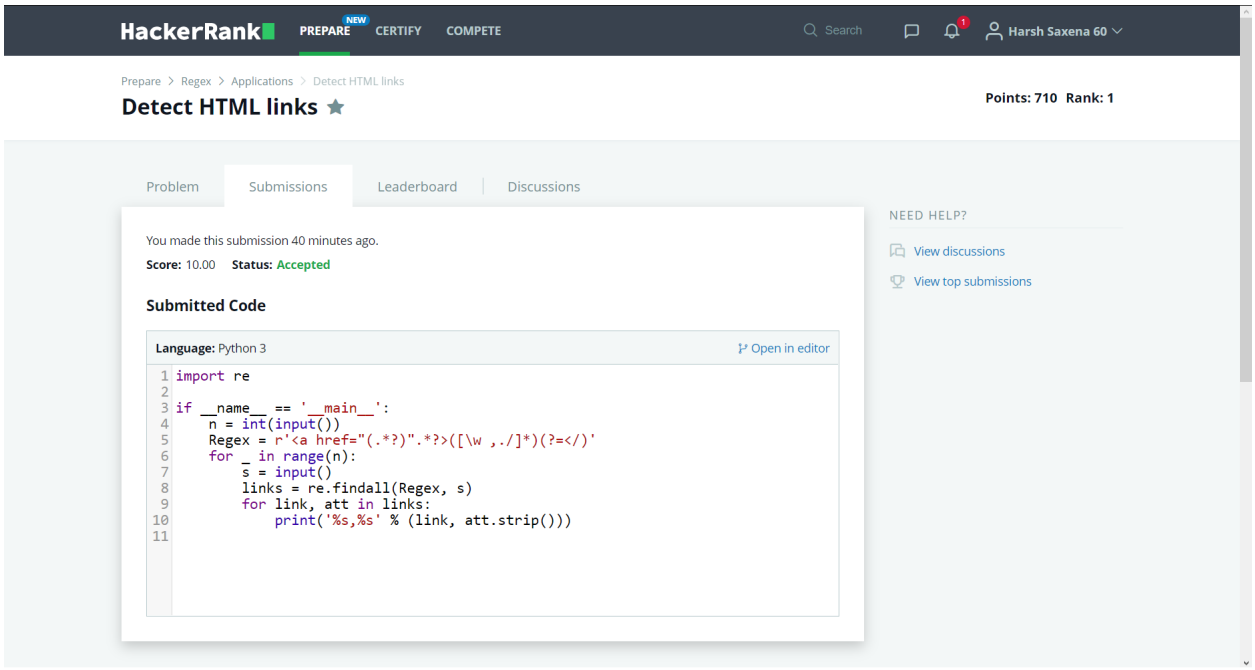
2.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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
You made this submission 38 minutes ago.
Score: 30.00 Status: Accepted
Submitted Code
Language: Python 3
Open in editor
class Main:
def __init__(self):
self.s = ''.join(sys.stdin.readlines())
def output(self):
if 'java' in self.s:
print("Java")
elif '#include' in self.s:
print("C")
else:
print("Python")
if __name__ == '__main__':
obj = Main()
obj.output()
NEED HELP?
View discussions
View top submissions

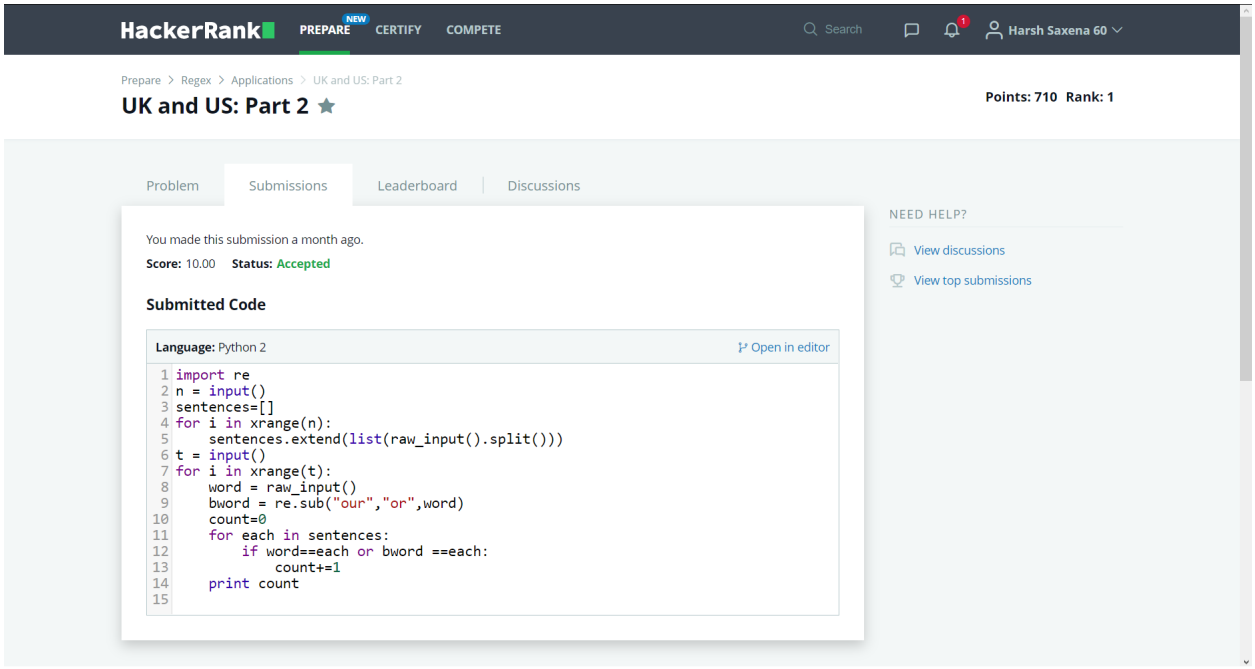
3.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > Building a Smart IDE: Identifying comments
Building a Smart IDE: Identifying comments
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
You made this submission 39 minutes ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
public class Solution {
public static void main(String[] args) {
Scanner input = new Scanner(System.in);
StringBuilder sb = new StringBuilder();
String text = "";
String regex = "(\\/)(*)(.|\\s)*?\\2\\1|(\\/\\/.*)";
while (input.hasNextLine()) {
NEED HELP?
View discussions
View top submissions

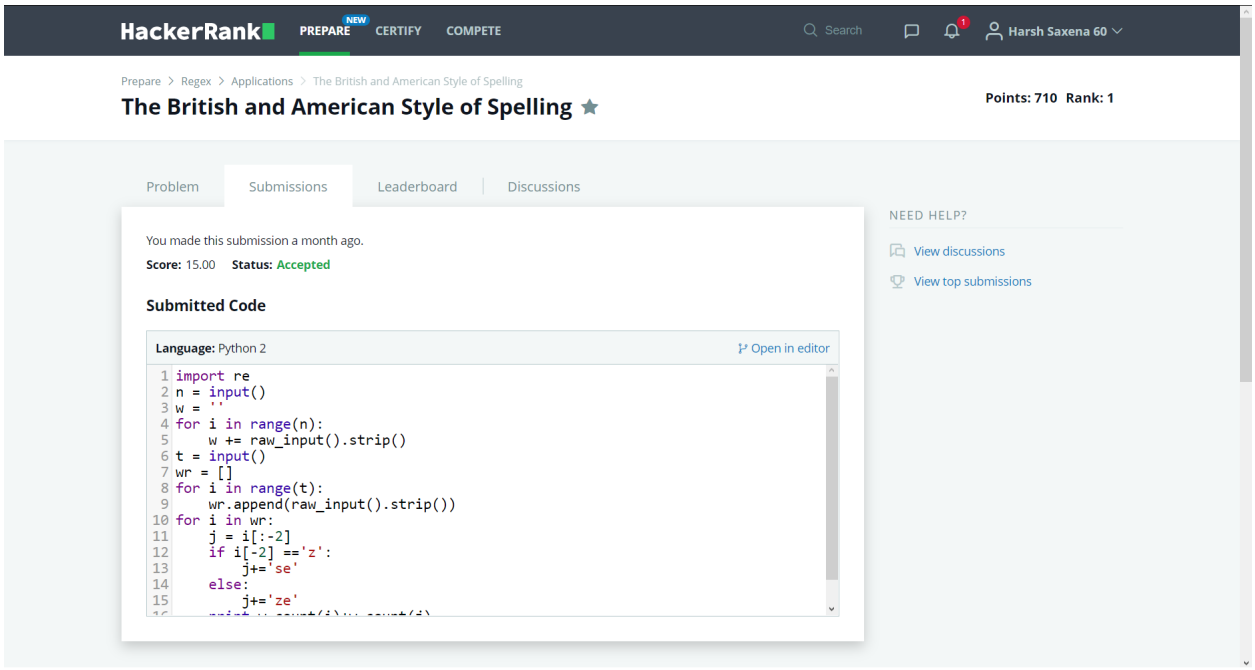
4.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > Detect HTML links
Detect HTML links
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
You made this submission 40 minutes ago.
Score: 10.00 Status: Accepted
Submitted Code
Language: Python 3
Open in editor
1 import re
2
3 if __name__ == '__main__':
4 n = int(input())
5 Regex = r'<a href="(.*?)".*?>([\w ,./]*)(?=</)'
6 for _ in range(n):
7 s = input()
8 links = re.findall(Regex, s)
9 for link, att in links:
10 print('%s,%s' % (link, att.strip()))
11
NEED HELP?
View discussions
View top submissions

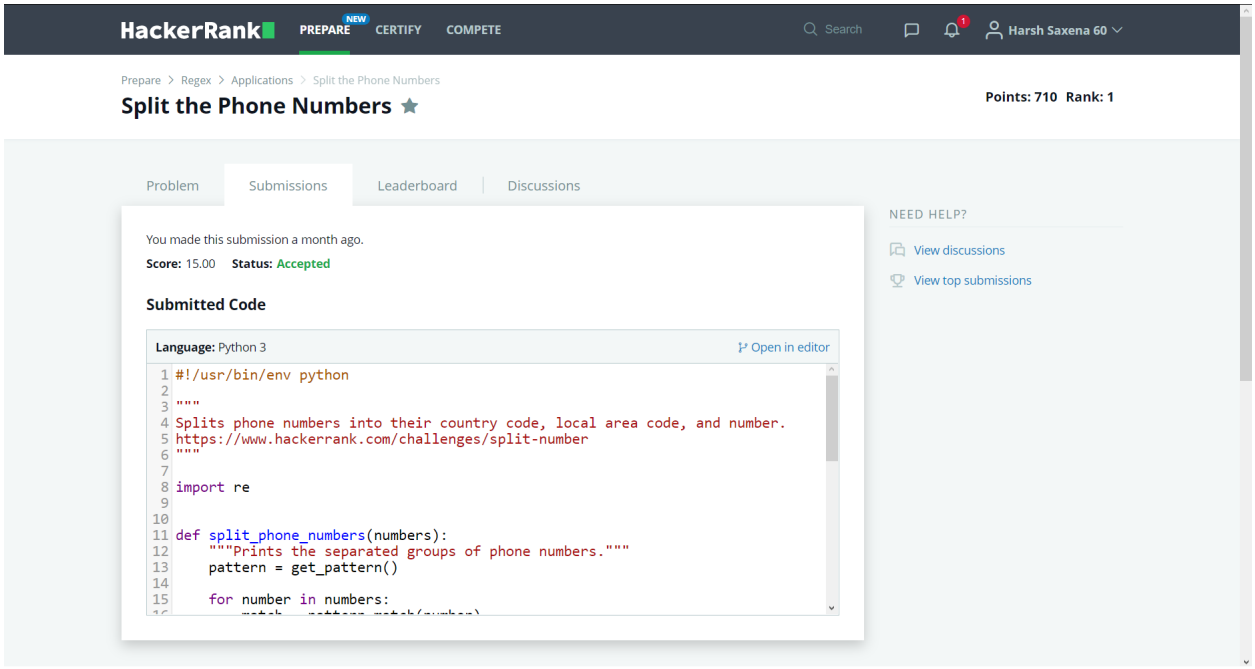
5.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > UK and US: Part 2
UK and US: Part 2
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
You made this submission a month ago.
Score: 10.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 import re
2 n = input()
3 sentences=[]
4 for i in xrange(n):
5 sentences.extend(list(raw_input().split()))
6 t = input()
7 for i in xrange(t):
8 word = raw_input()
9 bword = re.sub("our","or",word)
10 count=0
11 for each in sentences:
12 if word==each or bword ==each:
13 count+=1
14 print count
15
NEED HELP?
View discussions
View top submissions

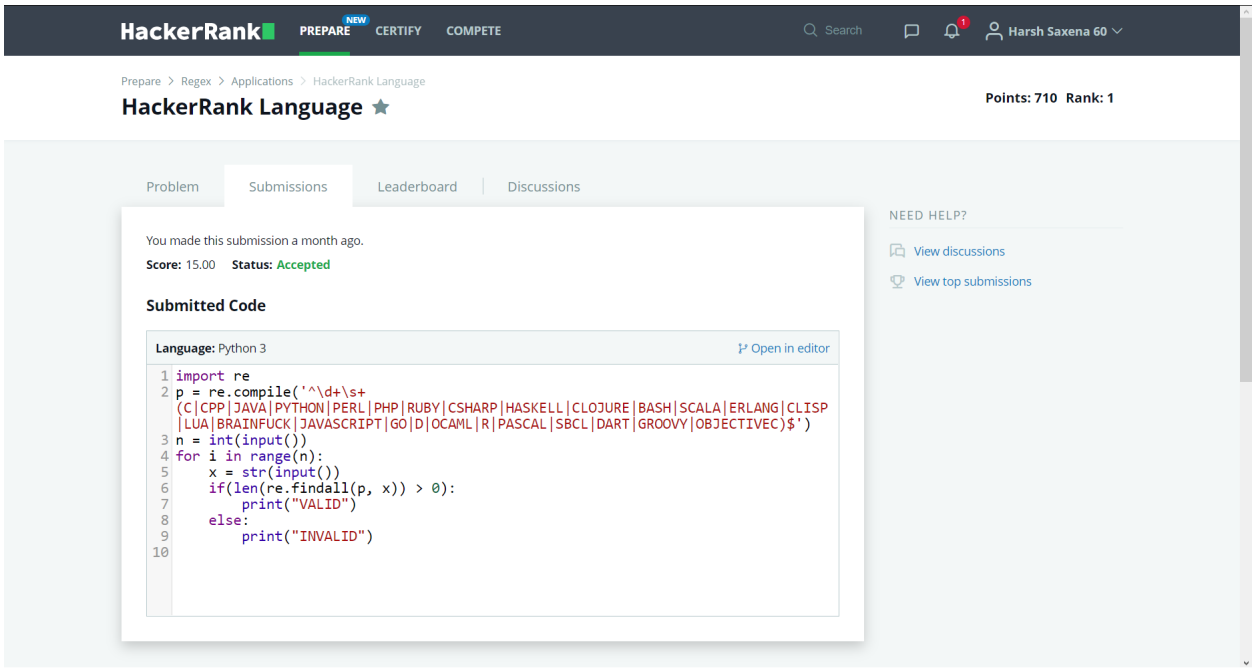
6.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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
You made this submission a month ago.
Score: 15.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 import re
2 n = input()
3 w = ''
4 for i in range(n):
5 w += raw_input().strip()
6 t = input()
7 wr = []
8 for i in range(t):
9 wr.append(raw_input().strip())
10 for i in wr:
11 j = i[:-2]
12 if i[-2] =='z':
13 j+='se'
14 else:
15 j+='ze'
NEED HELP?
View discussions
View top submissions

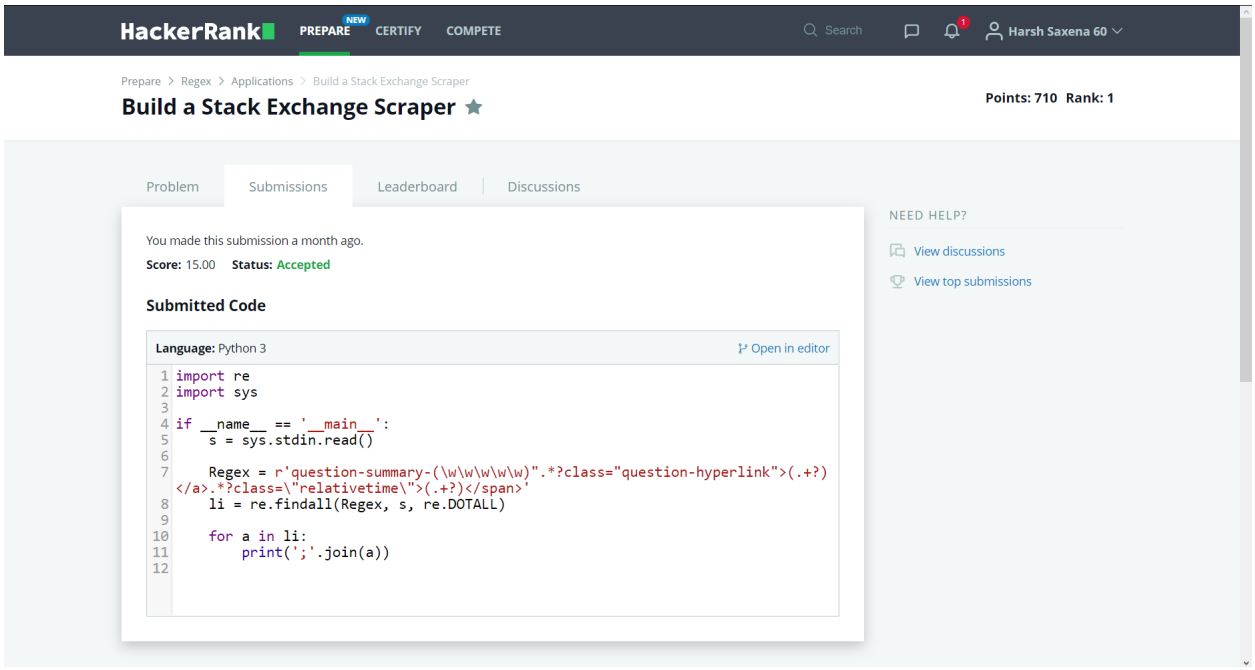
7.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > Split the Phone Numbers
Split the Phone Numbers
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
#!/usr/bin/env python
"""
Splits phone numbers into their country code, local area code, and number.
https://www.hackerrank.com/challenges/split-number
"""
import re
def split_phone_numbers(numbers):
"""Prints the separated groups of phone numbers."""
pattern = get_pattern()
for number in numbers:
NEED HELP?
View discussions
View top submissions

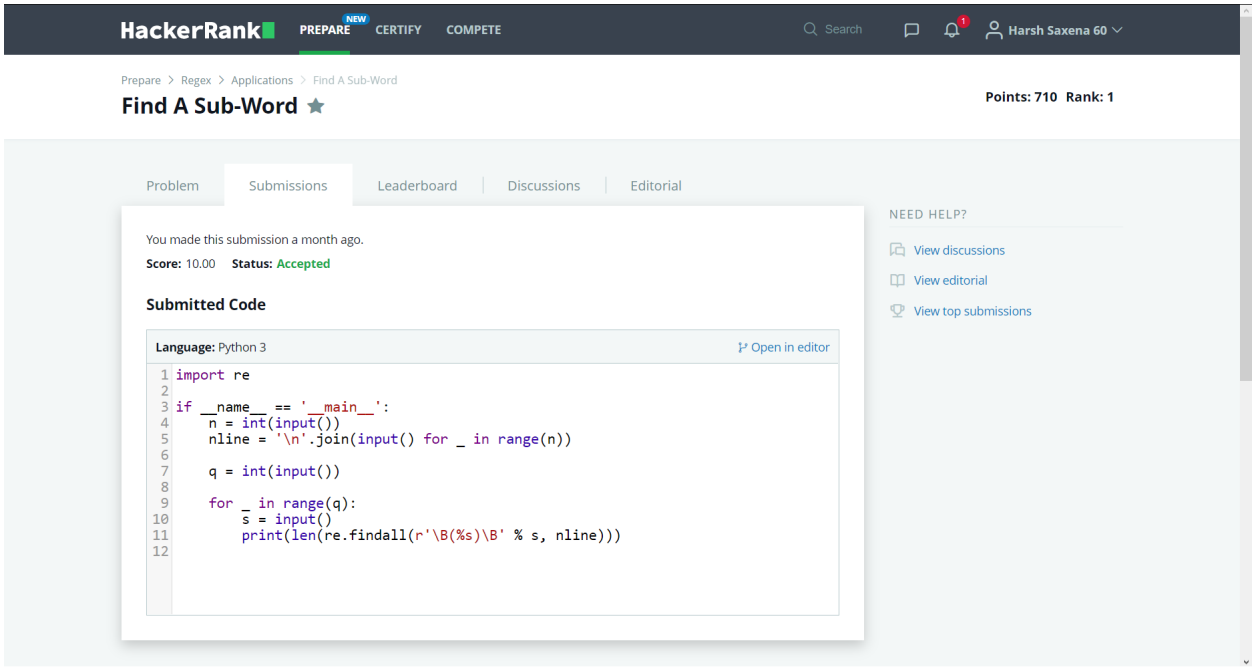
8.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > HackerRank Language
HackerRank Language
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
import re
p = re.compile('^\d+\s+
(C|CPP|JAVA|PYTHON|PERL|PHP|RUBY|CSHARP|HASKELL|CLOJURE|BASH|SCALA|ERLANG|CLISP
|LUA|BRAINFUCK|JAVASCRIPT|GO|D|OCAML|R|PASCAL|SBCL|DART|GROOVY|OBJECTIVEC)$')
n = int(input())
for i in range(n):
x = str(input())
if(len(re.findall(p, x)) > 0):
print("VALID")
else:
print("INVALID")
NEED HELP?
View discussions
View top submissions

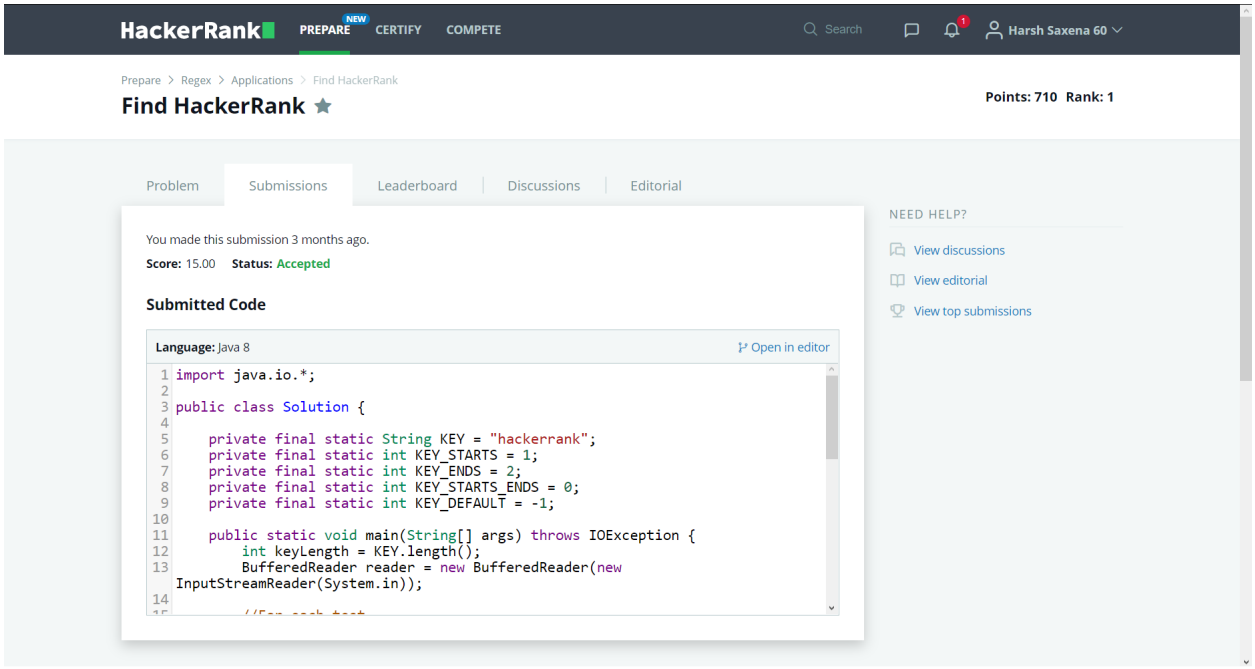
9.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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
import re
import sys
if __name__ == '__main__':
s = sys.stdin.read()
Regex = r'question-summary-(\w\w\w\w\w)".*?class="question-hyperlink">(.+?).*?class=\"relativetime\">(.+?)'
li = re.findall(Regex, s, re.DOTALL)
for a in li:
print(';'.join(a))
NEED HELP?
View discussions
View top submissions

10.

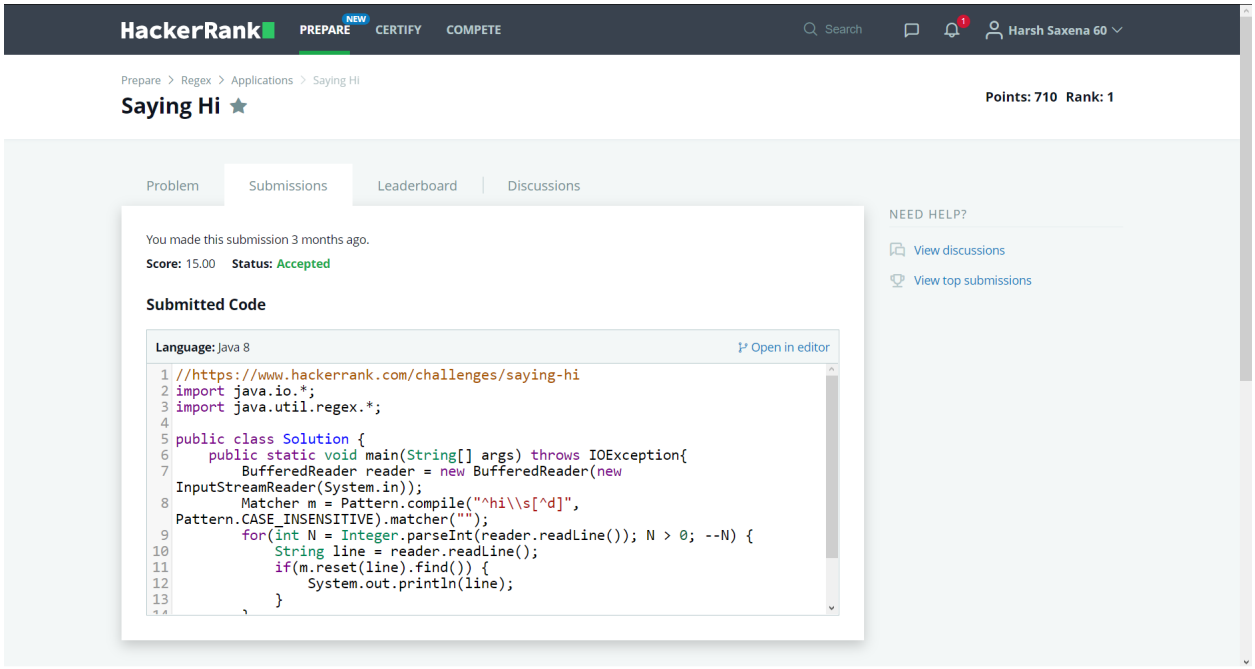

HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > Find A Sub-Word
Find A Sub-Word
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission a month ago.
Score: 10.00 Status: Accepted
Submitted Code
Language: Python 3
Open in editor
import re
if __name__ == '__main__':
n = int(input())
nline = '\n'.join(input() for _ in range(n))
q = int(input())
for _ in range(q):
s = input()
print(len(re.findall(r'\B(%s)\B' % s, nline)))
NEED HELP?
View discussions
View editorial
View top submissions

11.

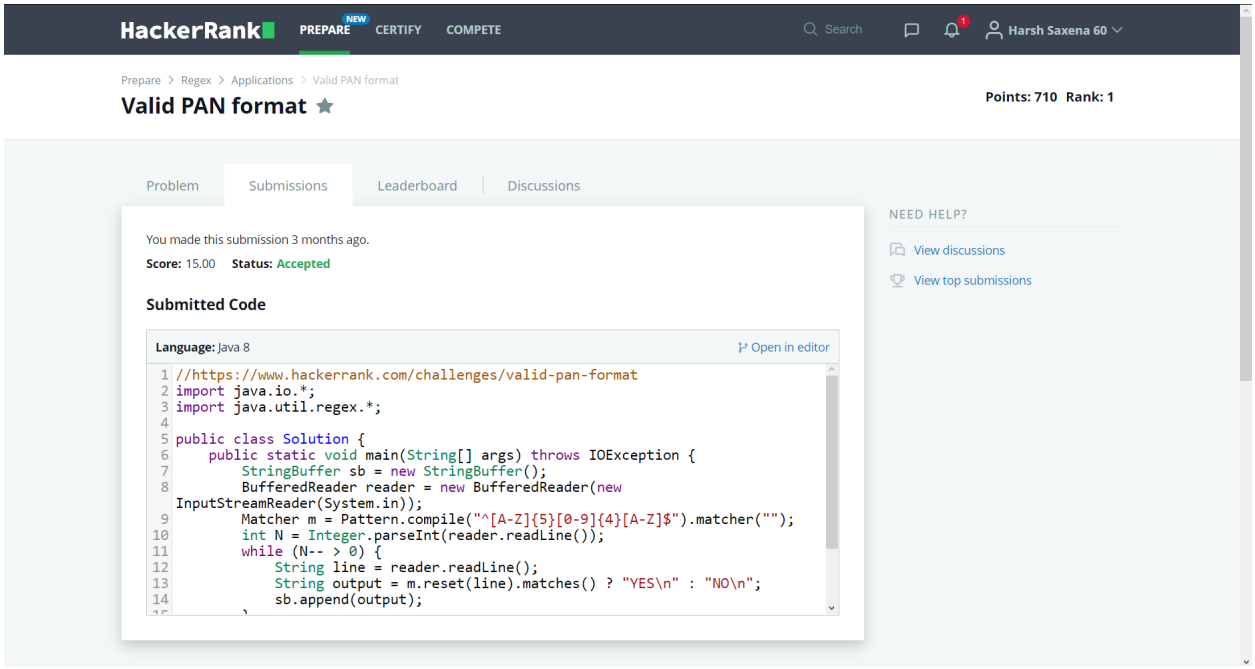



```java
import java.io.*;

public class Solution {

    private final static String KEY = "hackerrank";
    private final static int KEY_STARTS = 1;
    private final static int KEY_ENDS = 2;
    private final static int KEY_STARTS_ENDS = 0;
    private final static int KEY_DEFAULT = -1;

    public static void main(String[] args) throws IOException {
        int keyLength = KEY.length();
        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
```

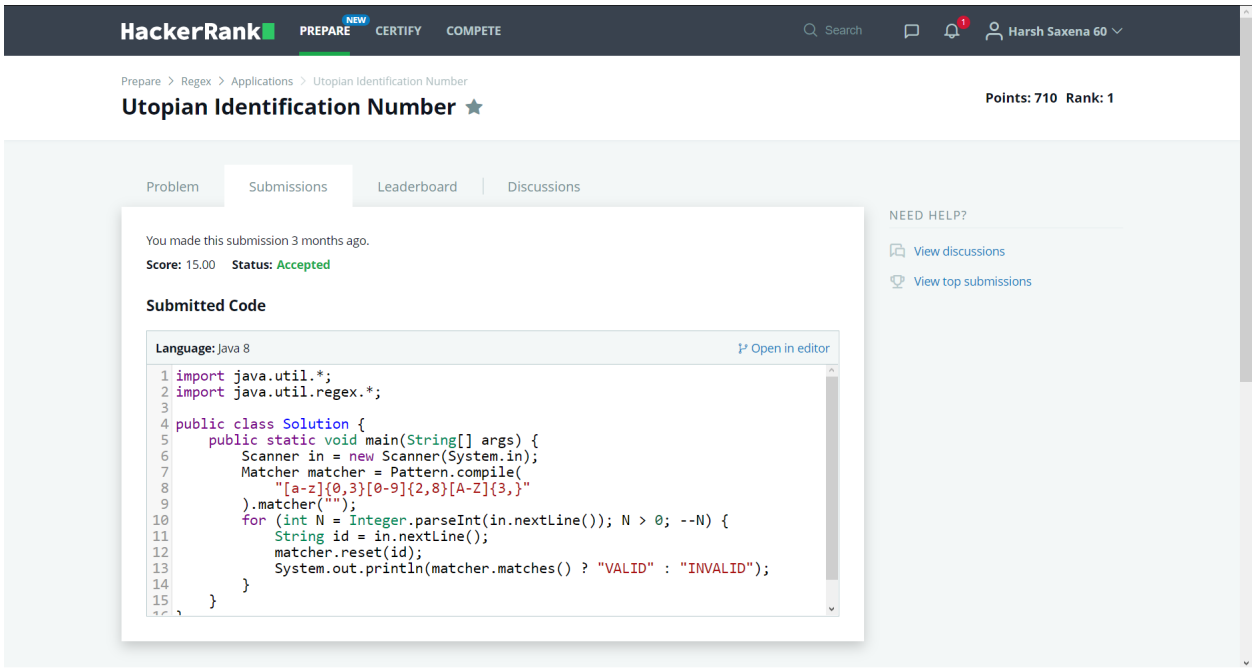
12.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > Saying Hi
Saying Hi
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
You made this submission 3 months ago.
Score: 15.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
1 //https://www.hackerrank.com/challenges/saying-hi
2 import java.io.*;
3 import java.util.regex.*;
4
5 public class Solution {
6 public static void main(String[] args) throws IOException{
7 BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
8 Matcher m = Pattern.compile("^hi\\s[^d]",
Pattern.CASE_INSENSITIVE).matcher("");
9 for(int N = Integer.parseInt(reader.readLine()); N > 0; --N) {
10 String line = reader.readLine();
11 if(m.reset(line).find()) {
12 System.out.println(line);
13 }
NEED HELP?
View discussions
View top submissions

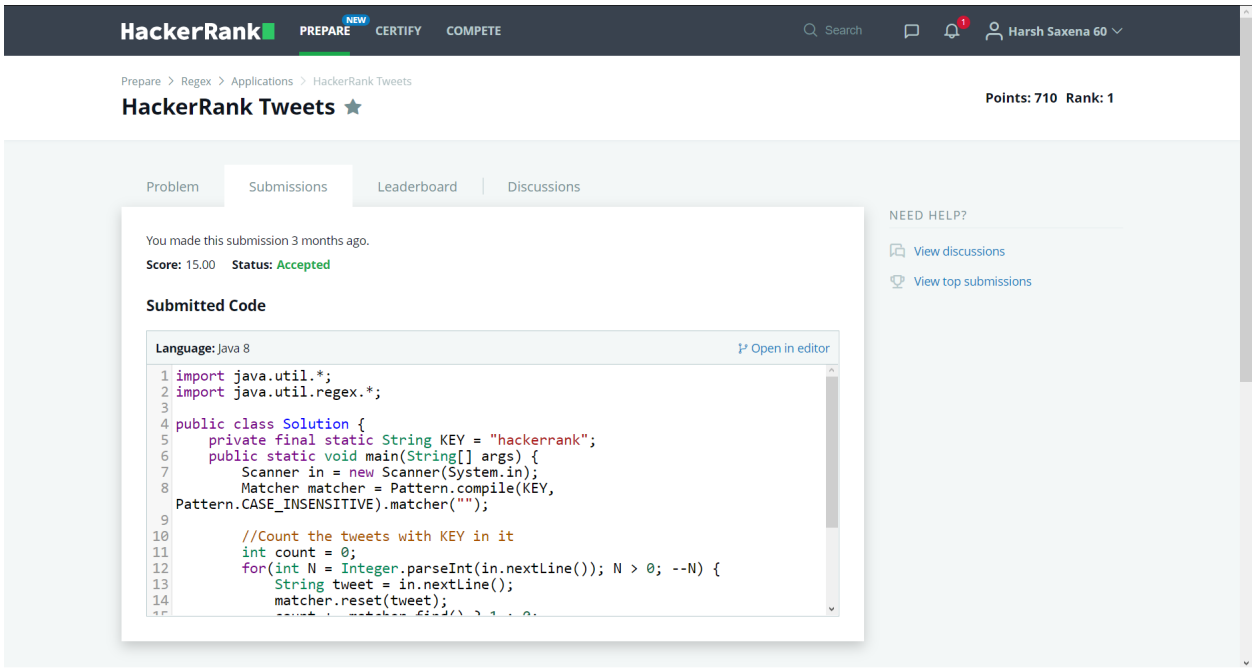
13.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > Valid PAN format
Valid PAN format
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
You made this submission 3 months ago.
Score: 15.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
NEED HELP?
View discussions
View top submissions
//https://www.hackerrank.com/challenges/valid-pan-format
import java.io.*;
import java.util.regex.*;
public class Solution {
public static void main(String[] args) throws IOException {
StringBuffer sb = new StringBuffer();
BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
Matcher m = Pattern.compile("^[A-Z]{5}[0-9]{4}[A-Z]$").matcher("");
int N = Integer.parseInt(reader.readLine());
while (N-- > 0) {
String line = reader.readLine();
String output = m.reset(line).matches() ? "YES\n" : "NO\n";
sb.append(output);

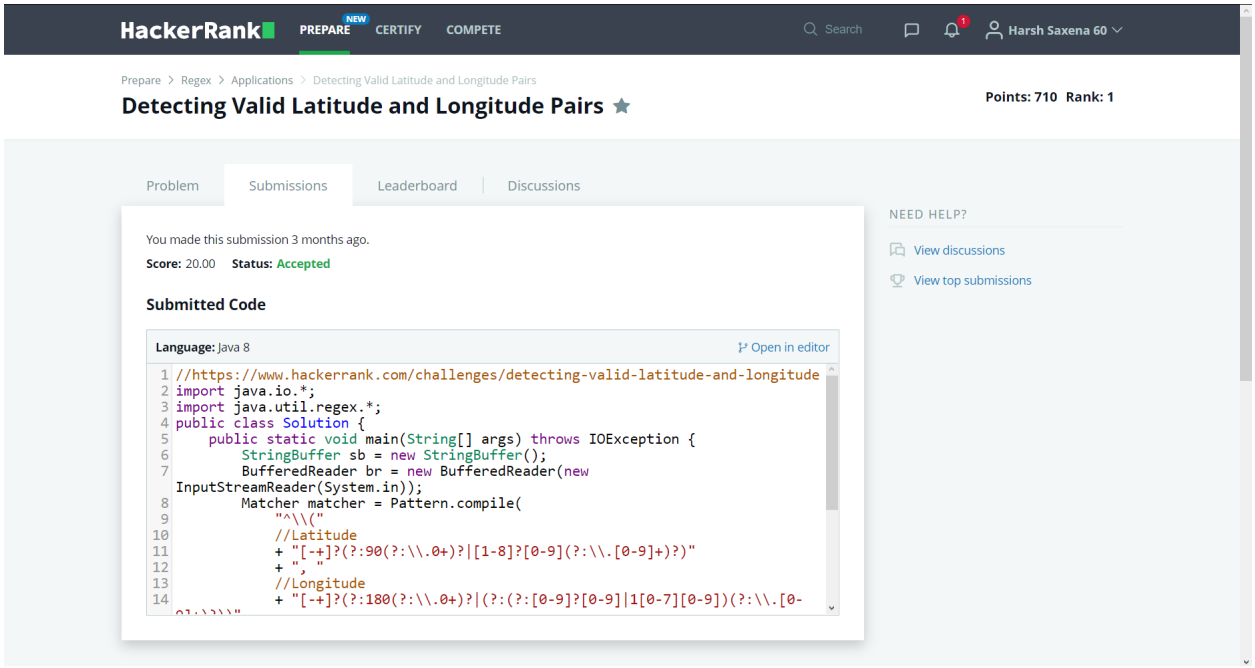
14.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > Utopian Identification Number
Utopian Identification Number
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
You made this submission 3 months ago.
Score: 15.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
import java.util.*;
import java.util.regex.*;
public class Solution {
public static void main(String[] args) {
Scanner in = new Scanner(System.in);
Matcher matcher = Pattern.compile(
"[a-z]{0,3}[0-9]{2,8}[A-Z]{3,}"
).matcher("");
for (int N = Integer.parseInt(in.nextLine()); N > 0; --N) {
String id = in.nextLine();
matcher.reset(id);
System.out.println(matcher.matches() ? "VALID" : "INVALID");
}
}
NEED HELP?
View discussions
View top submissions

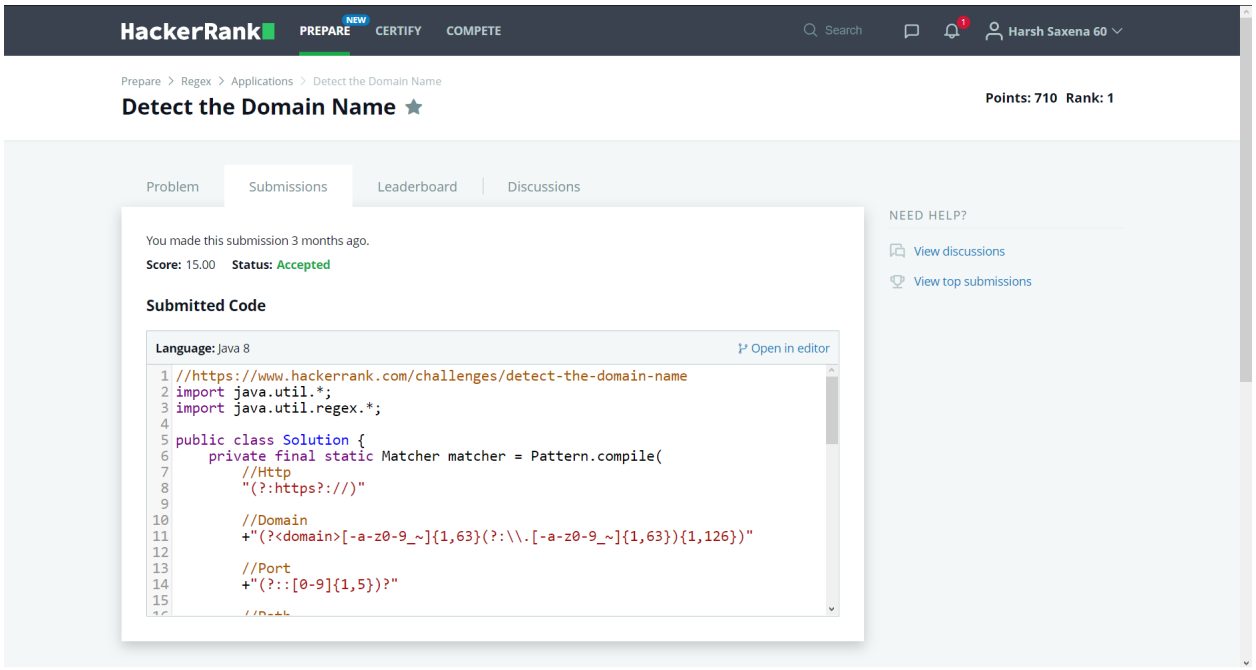
15.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > HackerRank Tweets
HackerRank Tweets
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
You made this submission 3 months ago.
Score: 15.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
1 import java.util.*;
2 import java.util.regex.*;
3
4 public class Solution {
5 private final static String KEY = "hackerrank";
6 public static void main(String[] args) {
7 Scanner in = new Scanner(System.in);
8 Matcher matcher = Pattern.compile(KEY,
Pattern.CASE_INSENSITIVE).matcher("");
9
10 //Count the tweets with KEY in it
11 int count = 0;
12 for(int N = Integer.parseInt(in.nextLine()); N > 0; --N) {
13 String tweet = in.nextLine();
14 matcher.reset(tweet);
NEED HELP?
View discussions
View top submissions

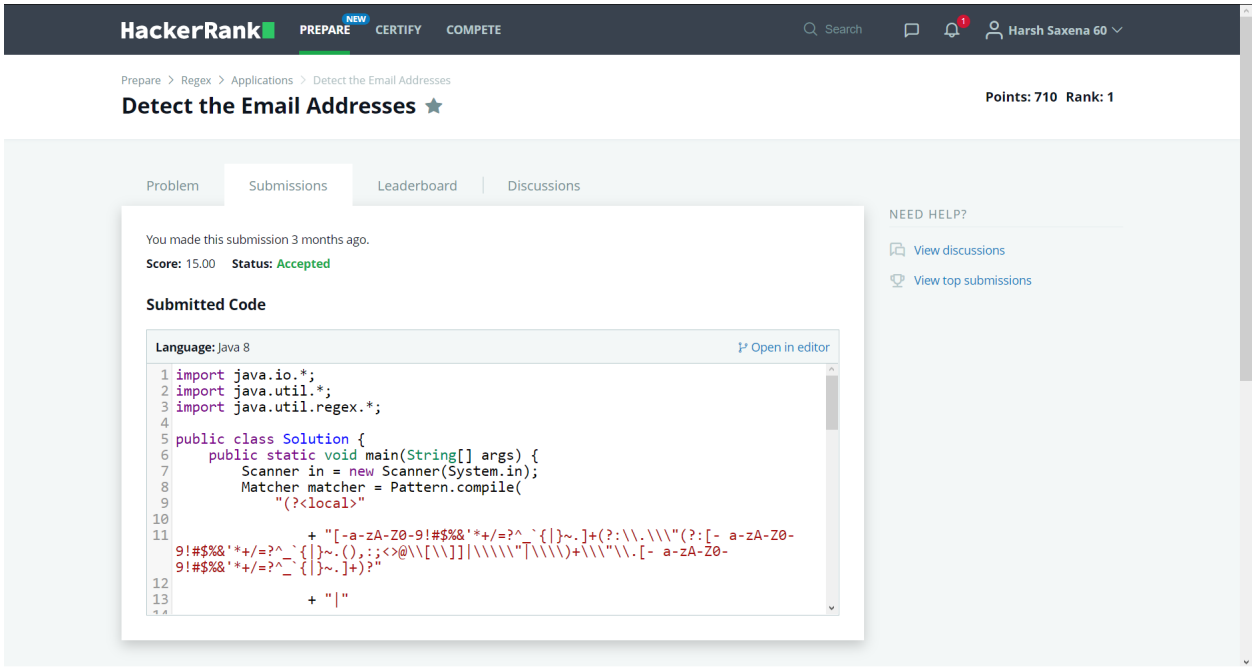
16.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > Detecting Valid Latitude and Longitude Pairs
Detecting Valid Latitude and Longitude Pairs
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
You made this submission 3 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
NEED HELP?
View discussions
View top submissions
//https://www.hackerrank.com/challenges/detecting-valid-latitude-and-longitude
import java.io.*;
import java.util.regex.*;
public class Solution {
public static void main(String[] args) throws IOException {
StringBuffer sb = new StringBuffer();
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
Matcher matcher = Pattern.compile(
"^\\("
//Latitude
+ "[-+]?(?:90(?:\\.0+)?|[1-8]?[0-9](?:\\.[0-9]+)?)"
+ ", "
//Longitude
+ "[-+]?(?:180(?:\\.0+)?|(?:(?:[0-9]?[0-9]|1[0-7][0-9])(?:\\.[0-

17.

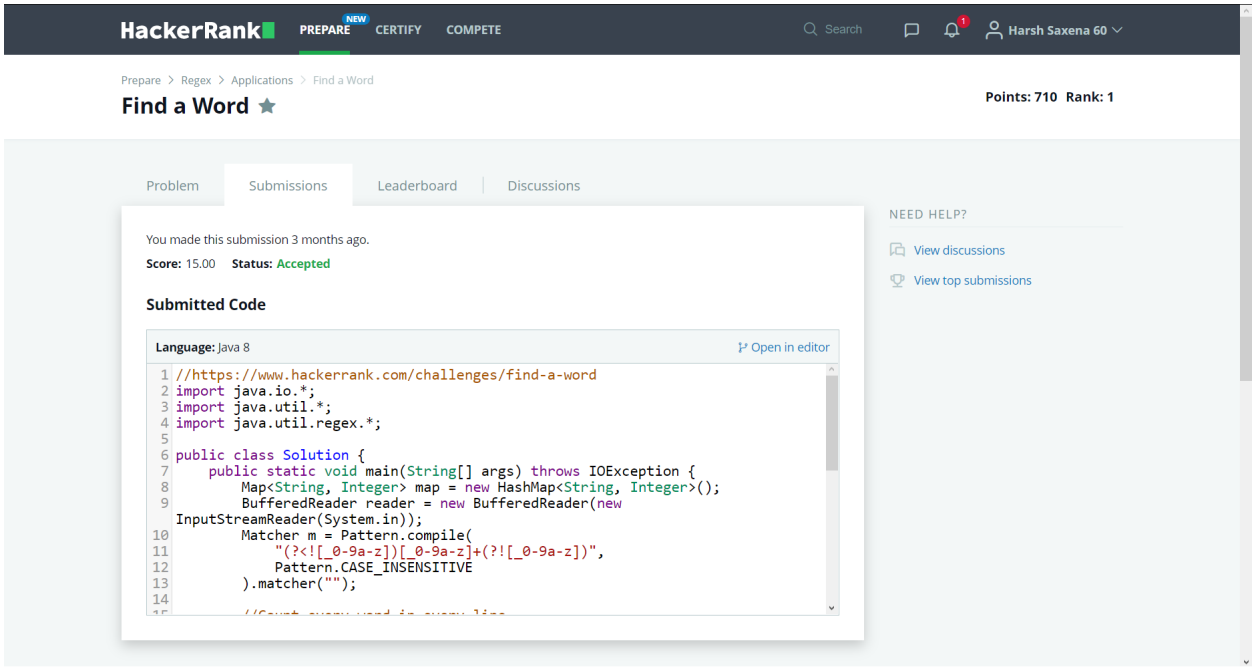

HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > Detect the Domain Name
Detect the Domain Name
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
You made this submission 3 months ago.
Score: 15.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
1 //https://www.hackerrank.com/challenges/detect-the-domain-name
2 import java.util.*;
3 import java.util.regex.*;
4
5 public class Solution {
6 private final static Matcher matcher = Pattern.compile(
7 //Http
8 "(?:https?://)"
9
10 //Domain
11 +"(?<domain>[-a-z0-9_~]{1,63}(?:\\.[-a-z0-9_~]{1,63}){1,126})"
12
13 //Port
14 +"(?::[0-9]{1,5})?"
15
NEED HELP?
View discussions
View top submissions

18.

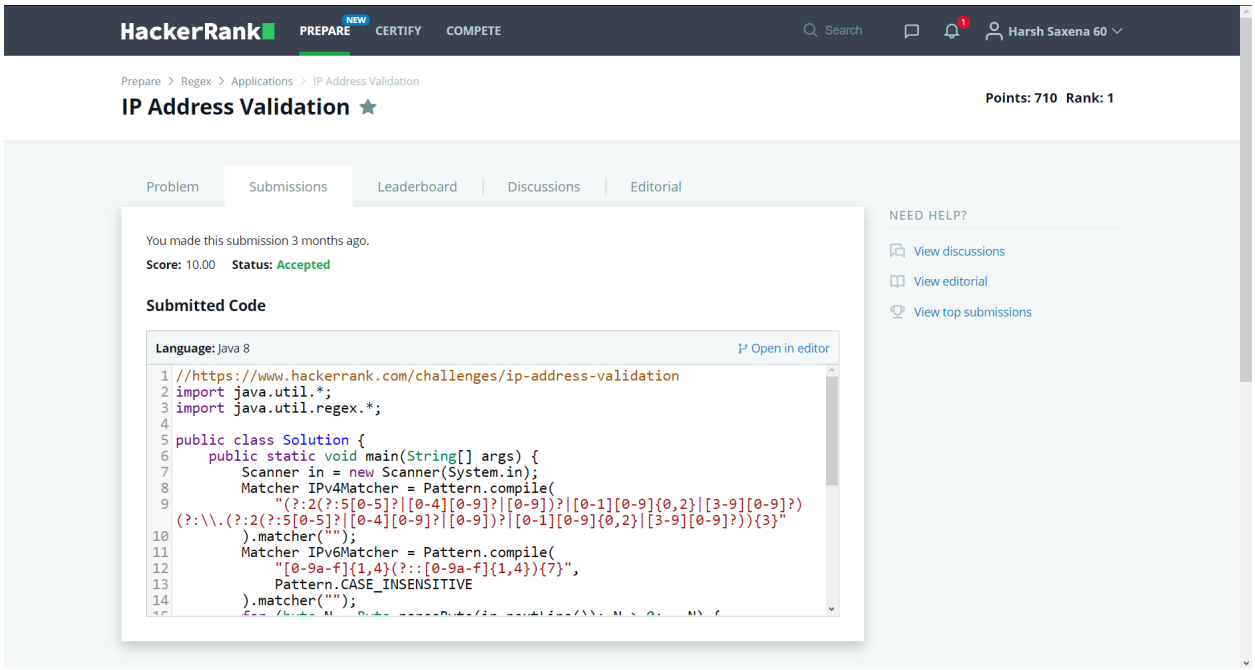



```
import java.io.*;
import java.util.*;
import java.util.regex.*;

public class Solution {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        Matcher matcher = Pattern.compile(
            "(?<local>"

                + "[-a-zA-Z0-9!#$%&'*+/=?^_`{|}~.]+(?:\\.\\\"(?:[- a-zA-Z0-
9!#$%&'*+/=?^_`{|}~.(),:;<>@\\[\\]]|\\\\\"|\\\\)+\\\"\\.[- a-zA-Z0-
9!#$%&'*+/=?^_`{|}~.]+)?"

                + "|"
```

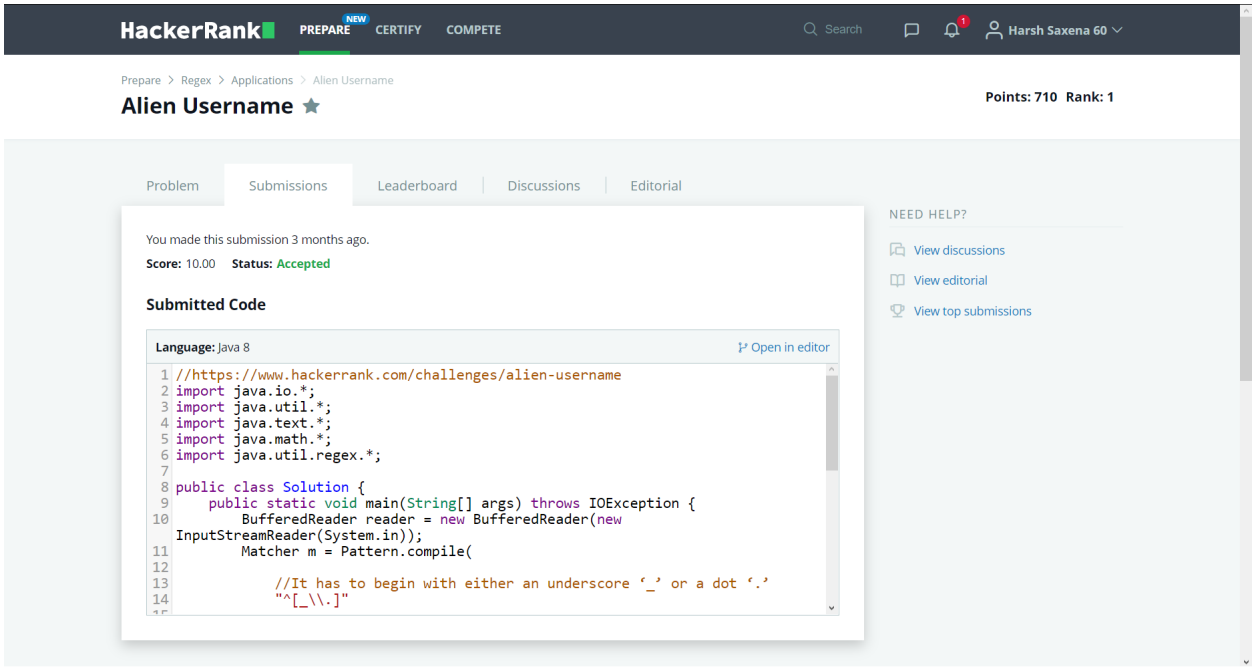
19.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Applications > Find a Word
Find a Word
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
You made this submission 3 months ago.
Score: 15.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
1 //https://www.hackerrank.com/challenges/find-a-word
2 import java.io.*;
3 import java.util.*;
4 import java.util.regex.*;
5
6 public class Solution {
7 public static void main(String[] args) throws IOException {
8 Map<String, Integer> map = new HashMap<String, Integer>();
9 BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
10 Matcher m = Pattern.compile(
11 "(?<![_0-9a-z])[_0-9a-z]+(?![_0-9a-z])",
12 Pattern.CASE_INSENSITIVE
13).matcher("");
14
NEED HELP?
View discussions
View top submissions

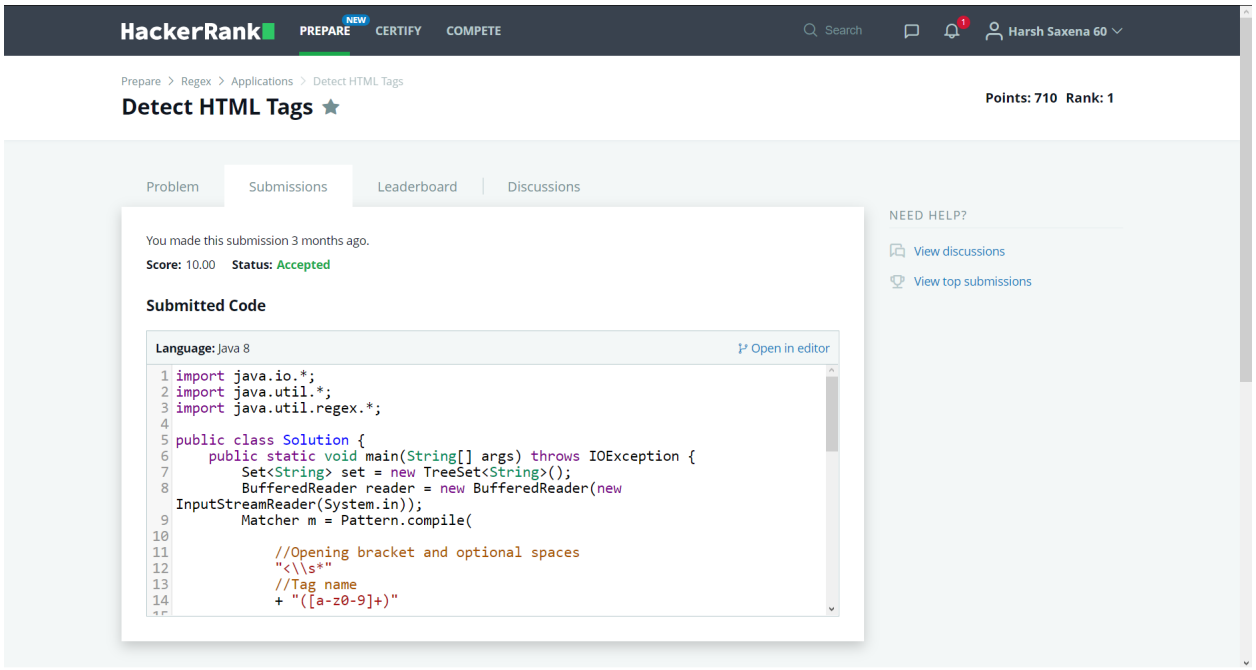
20.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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: Java 8
Open in editor
1 //https://www.hackerrank.com/challenges/ip-address-validation
2 import java.util.*;
3 import java.util.regex.*;
4
5 public class Solution {
6 public static void main(String[] args) {
7 Scanner in = new Scanner(System.in);
8 Matcher IPv4Matcher = Pattern.compile(
9 "(?:2(?:5[0-5]?|[0-4][0-9]?|[0-9])?|[0-1][0-9]{0,2}|[3-9][0-9]?)
(?:\\.(?:2(?:5[0-5]?|[0-4][0-9]?|[0-9])?|[0-1][0-9]{0,2}|[3-9][0-9]?)){3}"
10).matcher("");
11 Matcher IPv6Matcher = Pattern.compile(
12 "[0-9a-f]{1,4}(?::[0-9a-f]{1,4}){7}",
13 Pattern.CASE_INSENSITIVE
14).matcher("");
NEED HELP?
View discussions
View editorial
View top submissions

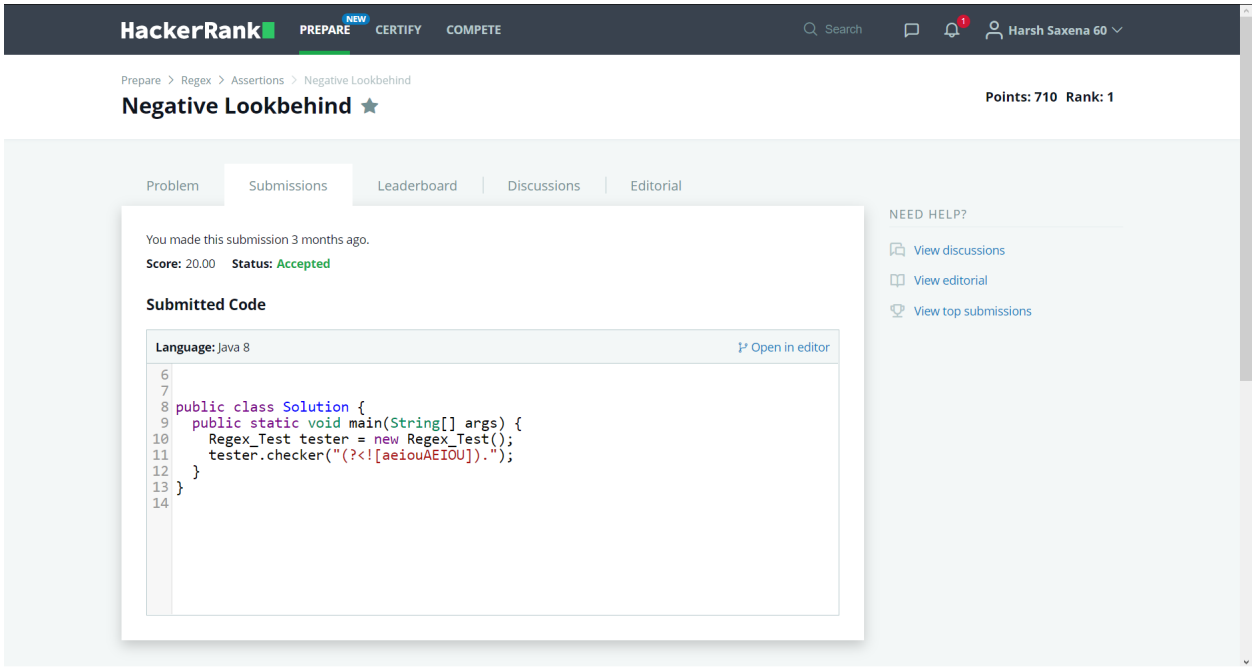
21.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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: Java 8
Open in editor
1 //https://www.hackerrank.com/challenges/alien-username
2 import java.io.*;
3 import java.util.*;
4 import java.text.*;
5 import java.math.*;
6 import java.util.regex.*;
7
8 public class Solution {
9 public static void main(String[] args) throws IOException {
10 BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
11 Matcher m = Pattern.compile(
12
13 //It has to begin with either an underscore '_' or a dot '.'
14 "^[_\\.]"
NEED HELP?
View discussions
View editorial
View top submissions

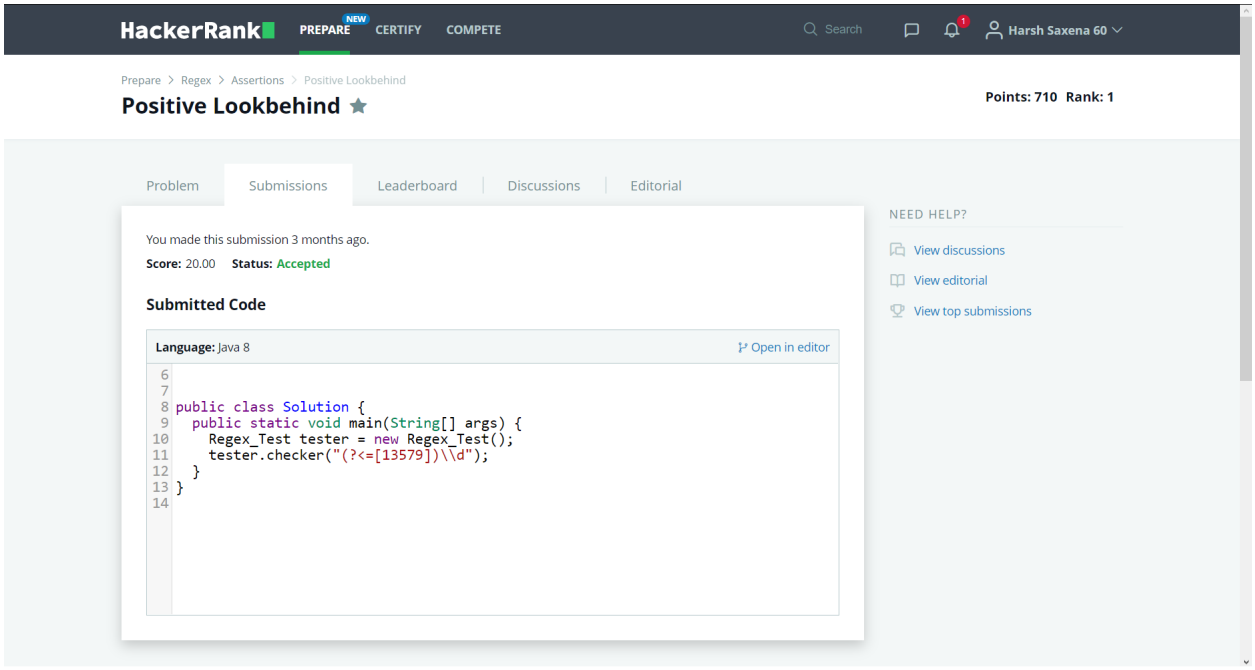
22.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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: Java 8
Open in editor
1 import java.io.*;
2 import java.util.*;
3 import java.util.regex.*;
4
5 public class Solution {
6 public static void main(String[] args) throws IOException {
7 Set<String> set = new TreeSet<String>();
8 BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
9 Matcher m = Pattern.compile(
10
11 //Opening bracket and optional spaces
12 "<\\s*"
13 //Tag name
14 + "([a-z0-9]+)"
NEED HELP?
View discussions
View top submissions

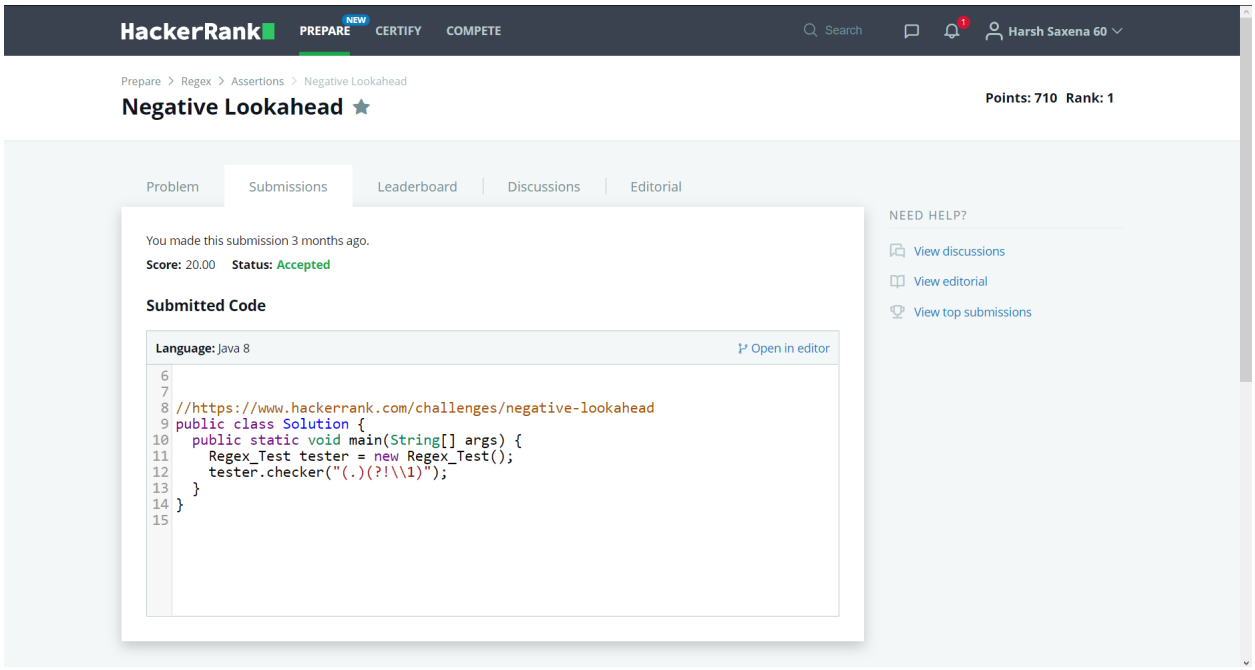
23.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Assertions > Negative Lookbehind
Negative Lookbehind
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 3 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
public class Solution {
public static void main(String[] args) {
Regex_Test tester = new Regex_Test();
tester.checker("(?<![aeiouAEIOU]).");
}
}
NEED HELP?
View discussions
View editorial
View top submissions

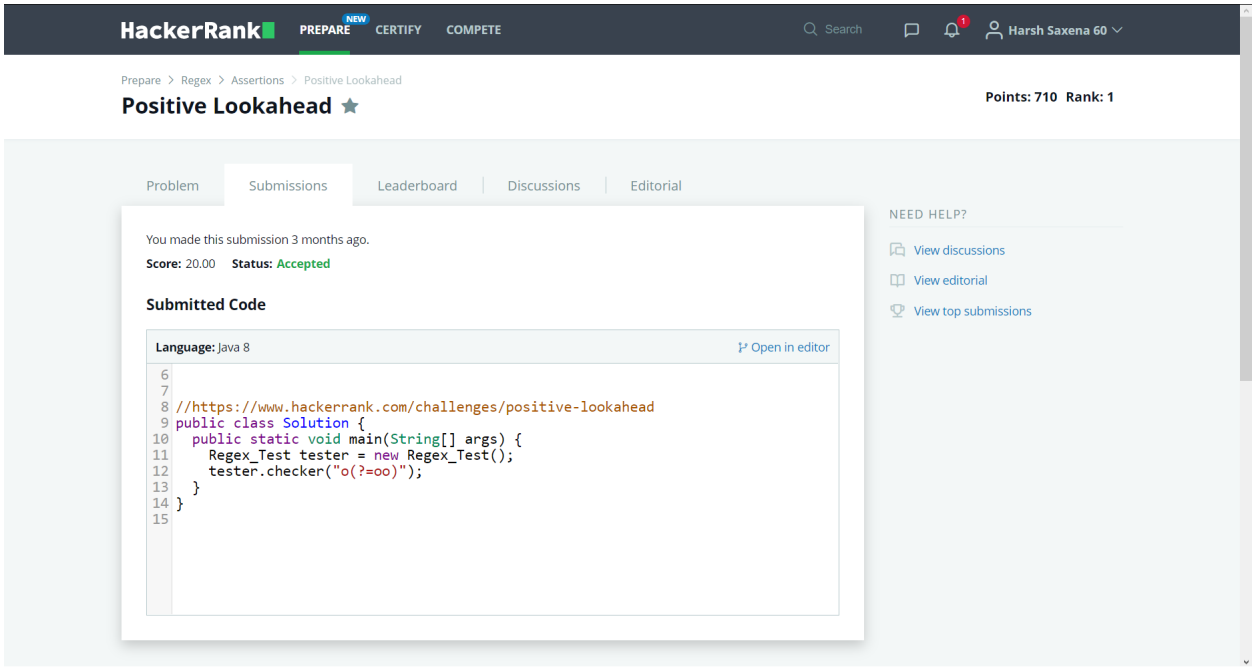
24.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Assertions > Positive Lookbehind
Positive Lookbehind
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 3 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
public class Solution {
public static void main(String[] args) {
Regex_Test tester = new Regex_Test();
tester.checker("(?<=[13579])\\d");
}
}
NEED HELP?
View discussions
View editorial
View top submissions

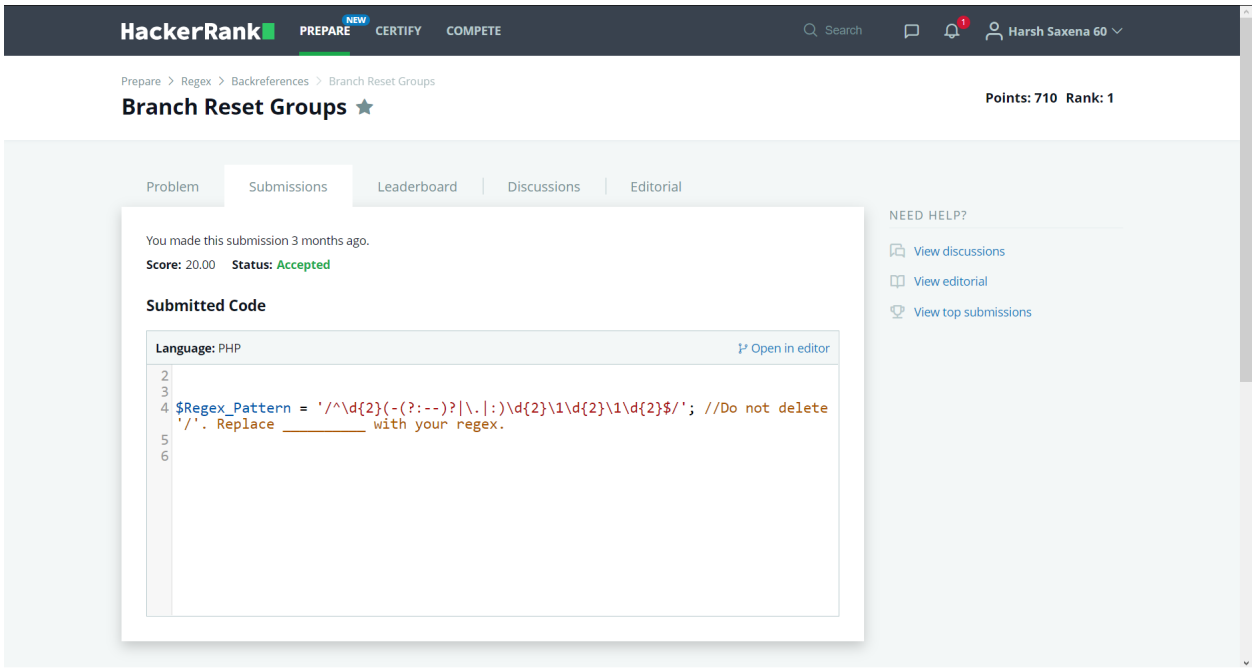
25.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Assertions > Negative Lookahead
Negative Lookahead
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 3 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
//https://www.hackerrank.com/challenges/negative-lookahead
public class Solution {
public static void main(String[] args) {
Regex_Test tester = new Regex_Test();
tester.checker("(.)(?!\\1)");
}
}
NEED HELP?
View discussions
View editorial
View top submissions

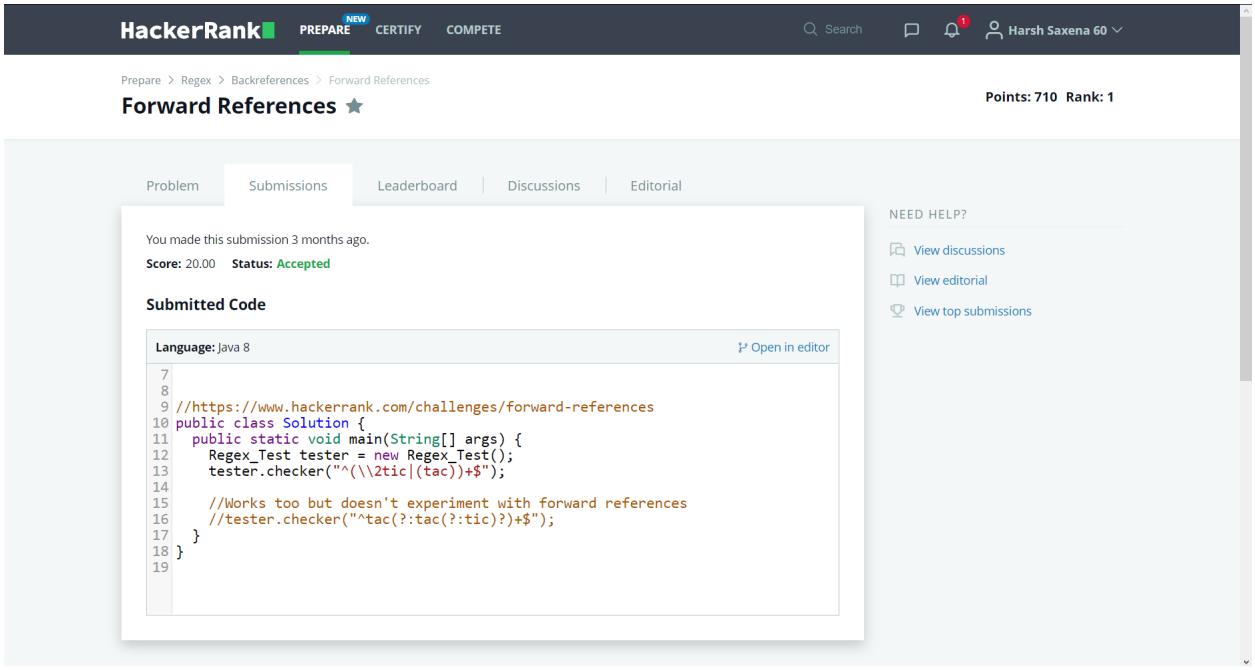
26.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Assertions > Positive Lookahead
Positive Lookahead
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 3 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
//https://www.hackerrank.com/challenges/positive-lookahead
public class Solution {
public static void main(String[] args) {
Regex_Test tester = new Regex_Test();
tester.checker("o(?=oo)");
}
}
NEED HELP?
View discussions
View editorial
View top submissions

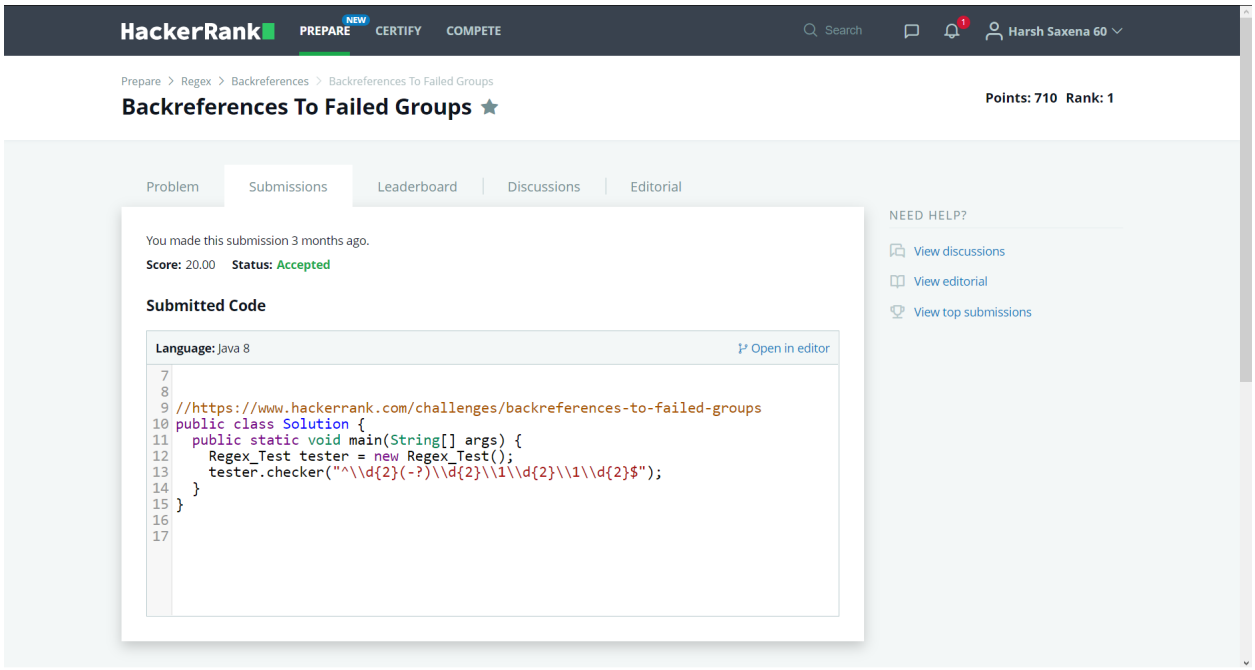
27.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Backreferences > Branch Reset Groups
Branch Reset Groups
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 3 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: PHP
Open in editor
$Regex_Pattern = '/^\d{2}(-(?:--)?|\.|:)\d{2}\1\d{2}\1\d{2}$/'; //Do not delete '/'. Replace _____ with your regex.
NEED HELP?
View discussions
View editorial
View top submissions

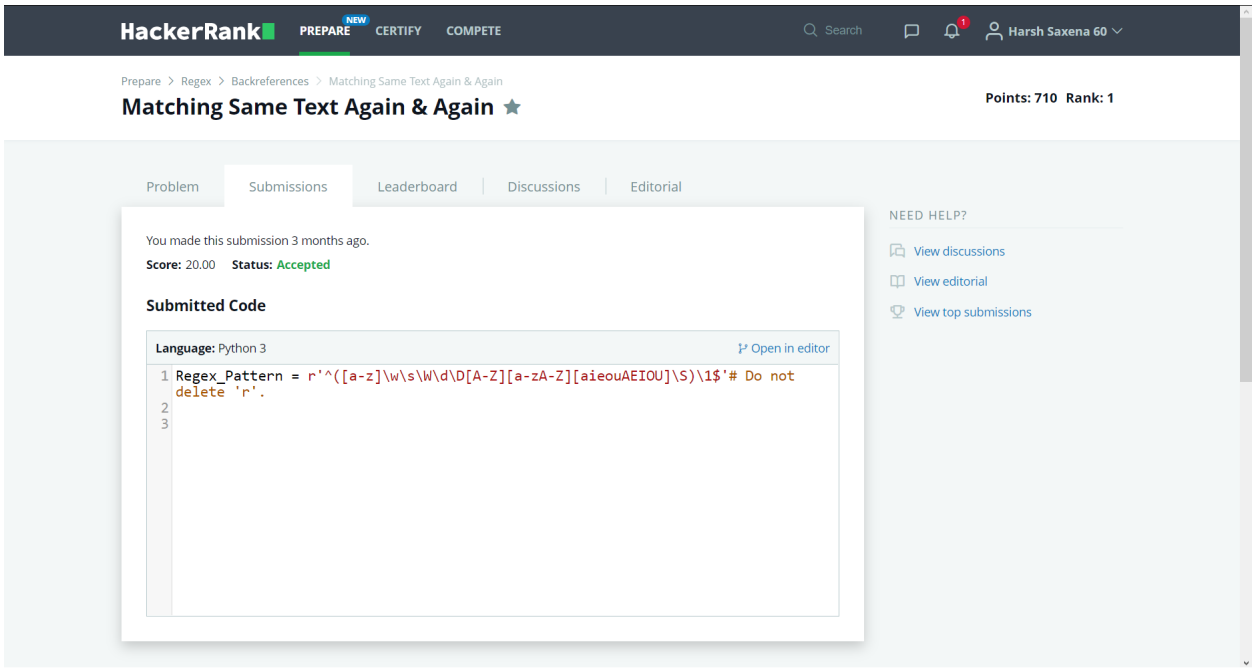
28.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Backreferences > Forward References
Forward References
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 3 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
//https://www.hackerrank.com/challenges/forward-references
public class Solution {
public static void main(String[] args) {
Regex_Test tester = new Regex_Test();
tester.checker("^(\\2tic|(tac))+$");
//Works too but doesn't experiment with forward references
//tester.checker("^tac(?:tac(?:tic)?)+$");
}
}
NEED HELP?
View discussions
View editorial
View top submissions

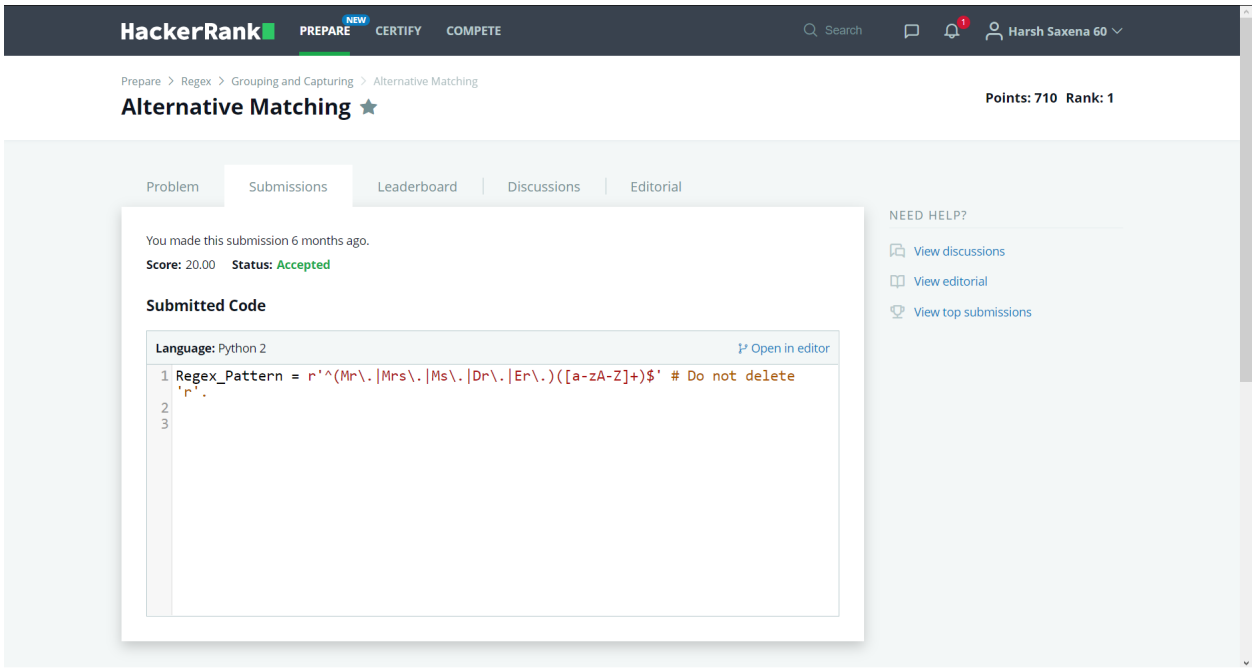
29.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Backreferences > Backreferences To Failed Groups
Backreferences To Failed Groups
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 3 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Java 8
Open in editor
//https://www.hackerrank.com/challenges/backreferences-to-failed-groups
public class Solution {
public static void main(String[] args) {
Regex_Test tester = new Regex_Test();
tester.checker("^\\d{2}(-?)\\d{2}\\1\\d{2}\\1\\d{2}$");
}
}
NEED HELP?
View discussions
View editorial
View top submissions

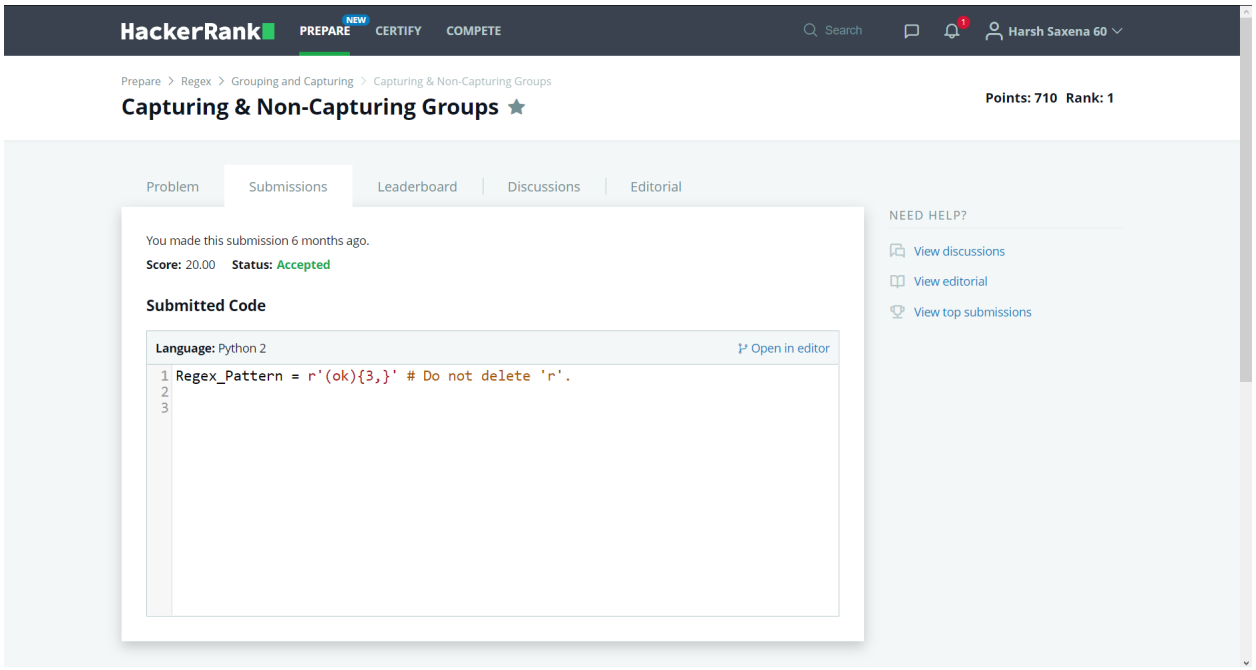
30.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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 3 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Python 3
Open in editor
Regex_Pattern = r'^([a-z]\w\s\W\d\D[A-Z][a-zA-Z][aieouAEIOU]\S)\1$'# Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions

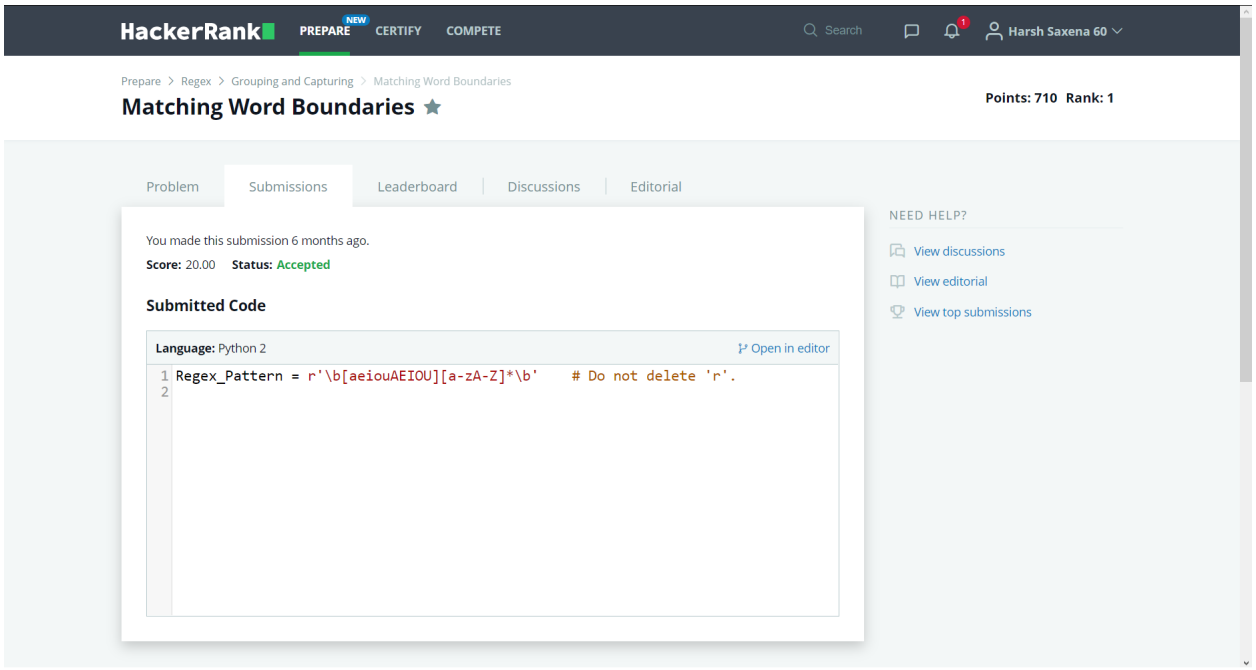
31.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Grouping and Capturing > Alternative Matching
Alternative Matching
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 6 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Python 2
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

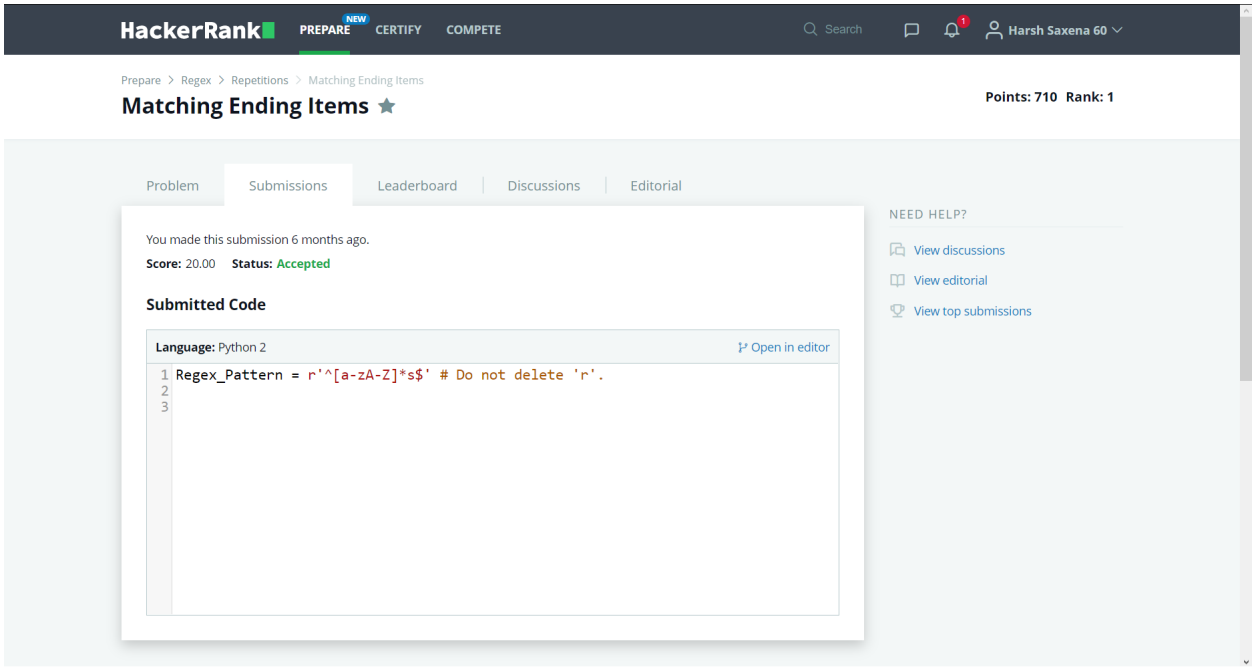
32.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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 6 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 Regex_Pattern = r'(ok){3,}' # Do not delete 'r'.
2
3
NEED HELP?
View discussions
View editorial
View top submissions

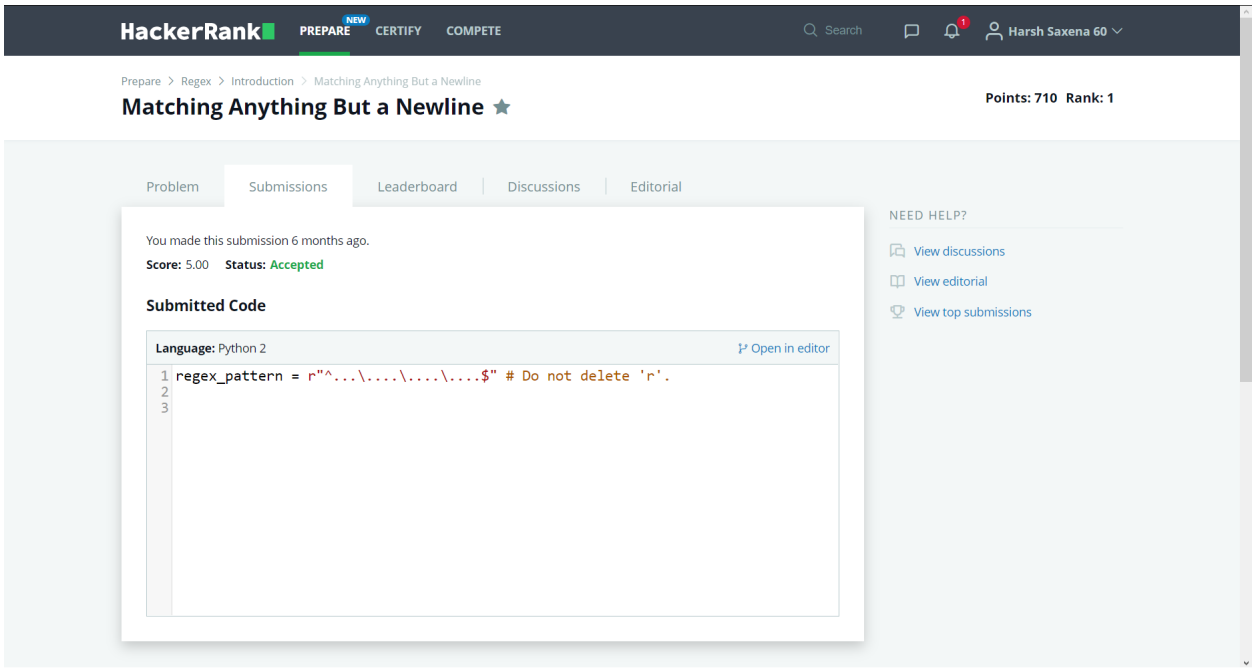
33.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Grouping and Capturing > Matching Word Boundaries
Matching Word Boundaries
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 6 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 Regex_Pattern = r'\b[aeiouAEIOU][a-zA-Z]*\b' # Do not delete 'r'.
2
NEED HELP?
View discussions
View editorial
View top submissions

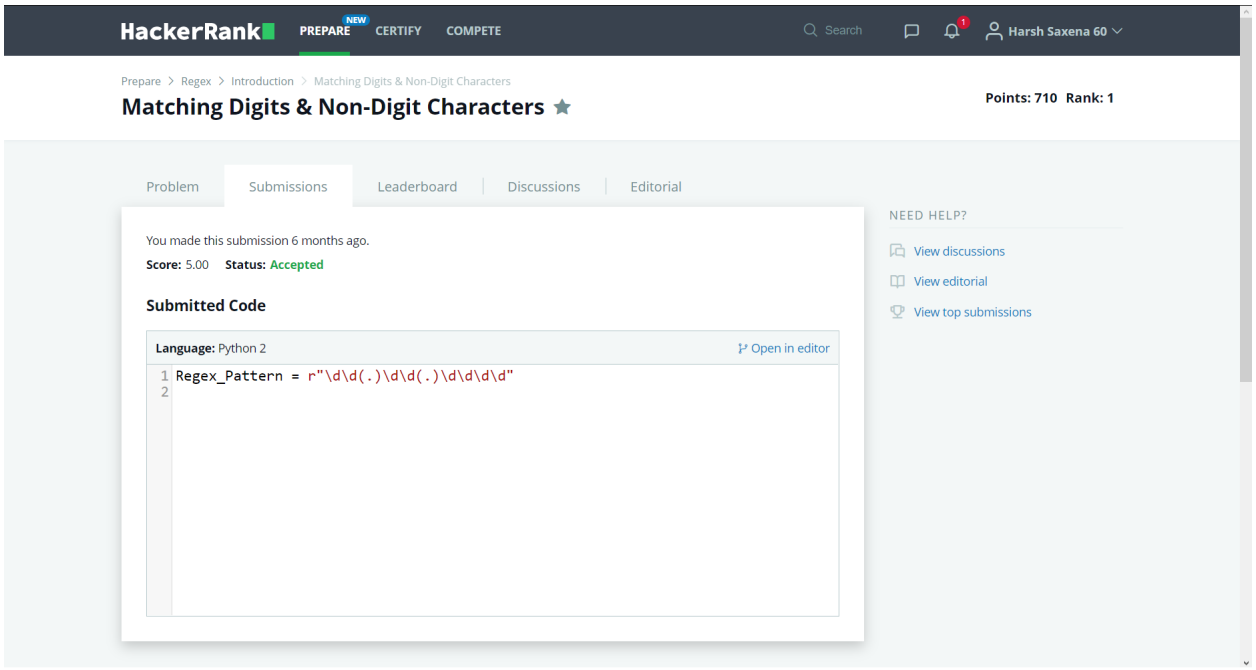
34.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Repetitions > Matching Ending Items
Matching Ending Items
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 6 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 Regex_Pattern = r'^[a-zA-Z]*s$' # Do not delete 'r'.
2
3
NEED HELP?
View discussions
View editorial
View top submissions

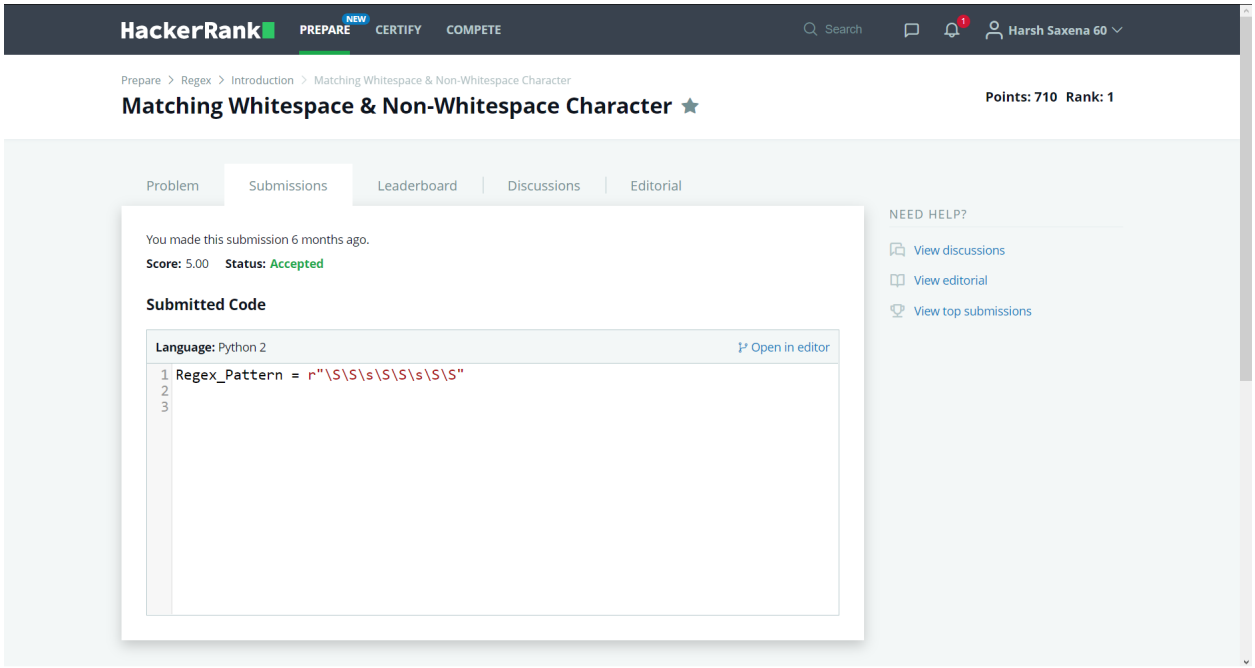
35.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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 6 months ago.
Score: 5.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 regex_pattern = r"^...\....\....\....$" # Do not delete 'r'.
2
3
NEED HELP?
View discussions
View editorial
View top submissions

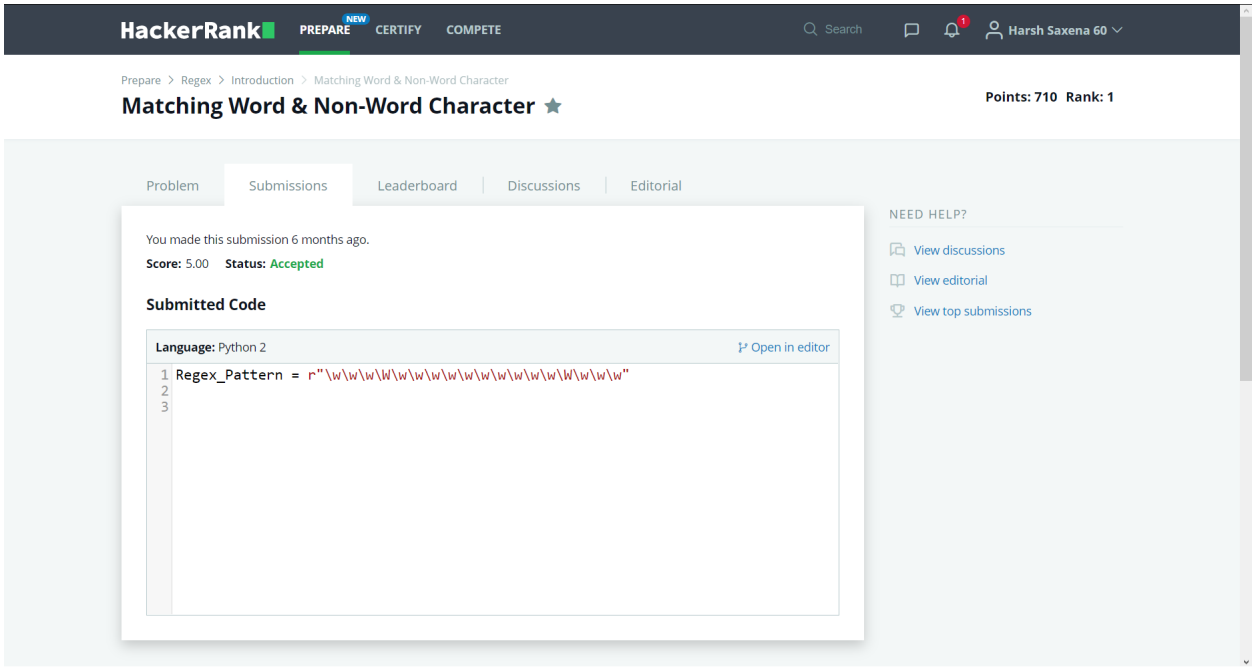
36.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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 6 months ago.
Score: 5.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 Regex_Pattern = r"\d\d(.)\d\d(.)\d\d\d\d"
2
NEED HELP?
View discussions
View editorial
View top submissions

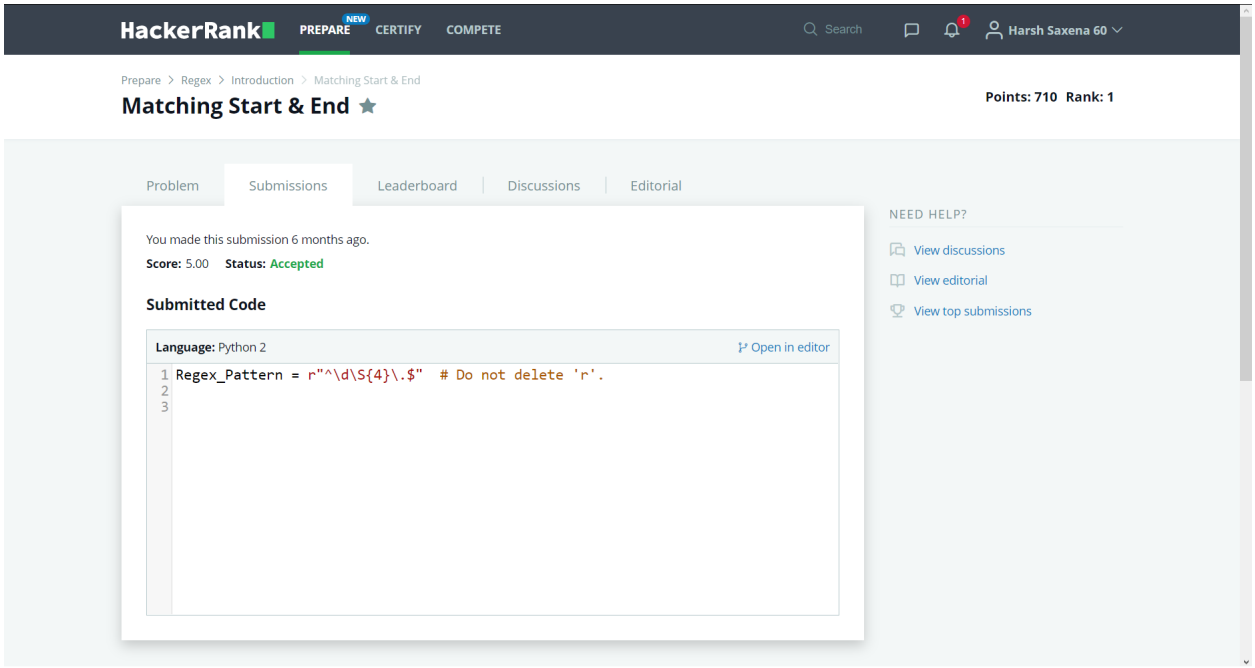
37.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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 6 months ago.
Score: 5.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 Regex_Pattern = r"\S\S\s\S\S\s\S\S"
2
3
NEED HELP?
View discussions
View editorial
View top submissions

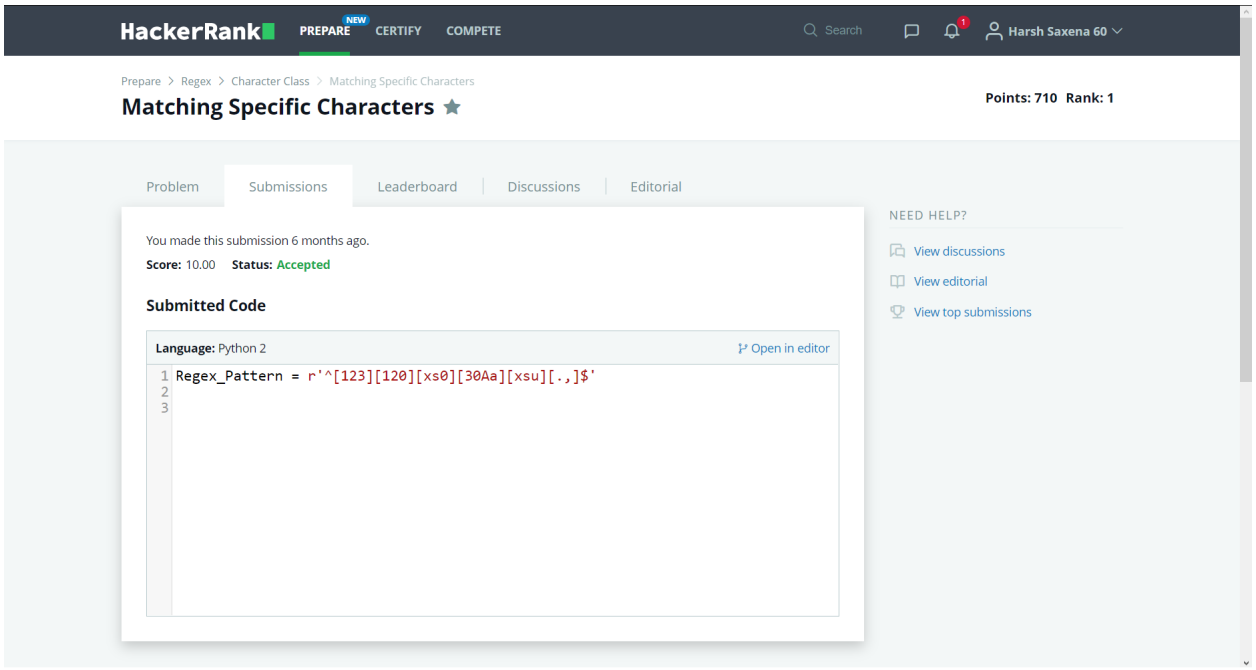
38.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Introduction > Matching Word & Non-Word Character
Matching Word & Non-Word Character
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 6 months ago.
Score: 5.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 Regex_Pattern = r"\w\w\w\W\w\w\w\w\w\w\w\w\w\W\w\w\w"
2
3
NEED HELP?
View discussions
View editorial
View top submissions

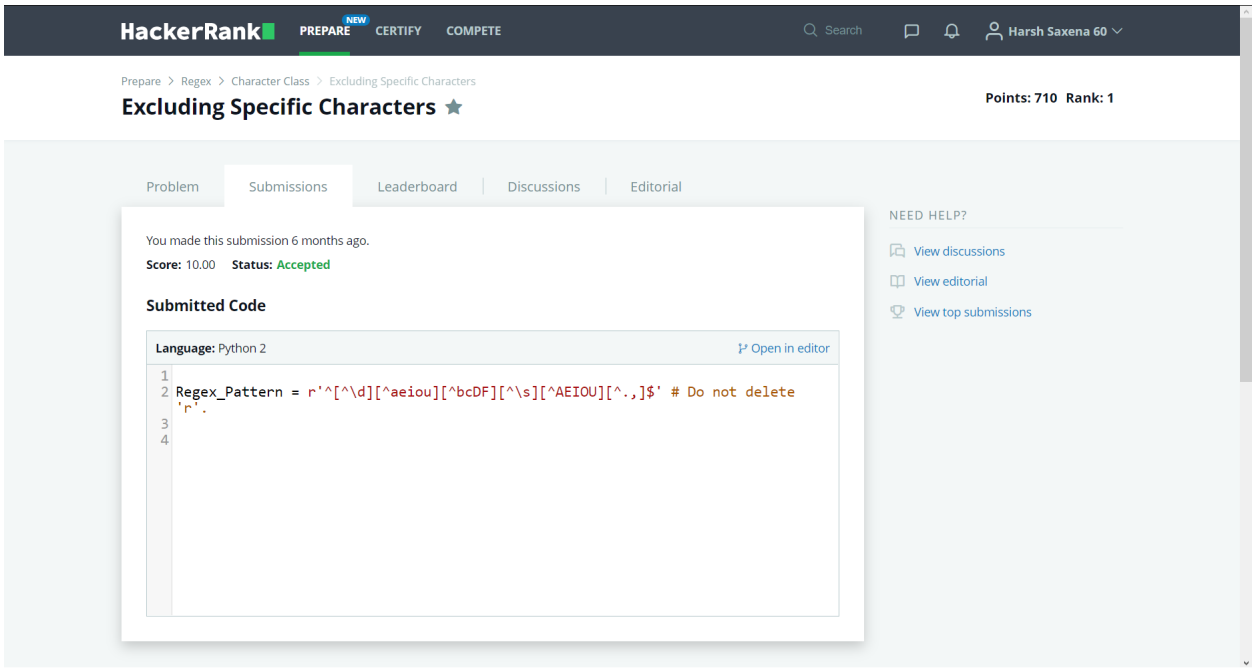
39.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Introduction > Matching Start & End
Matching Start & End
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 6 months ago.
Score: 5.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 Regex_Pattern = r"^\d\S{4}\.$" # Do not delete 'r'.
2
3
NEED HELP?
View discussions
View editorial
View top submissions

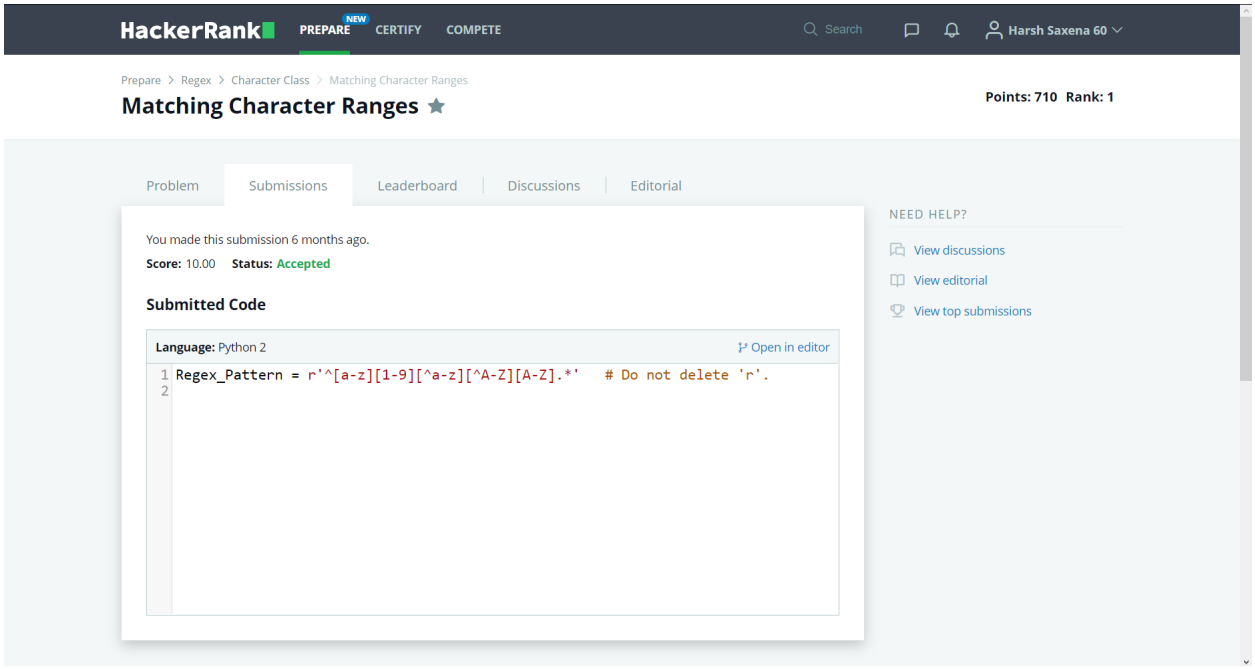
40.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Character Class > Matching Specific Characters
Matching Specific Characters
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 6 months ago.
Score: 10.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 Regex_Pattern = r'^[123][120][xs0][30Aa][xsu][.,]$'
2
3
NEED HELP?
View discussions
View editorial
View top submissions

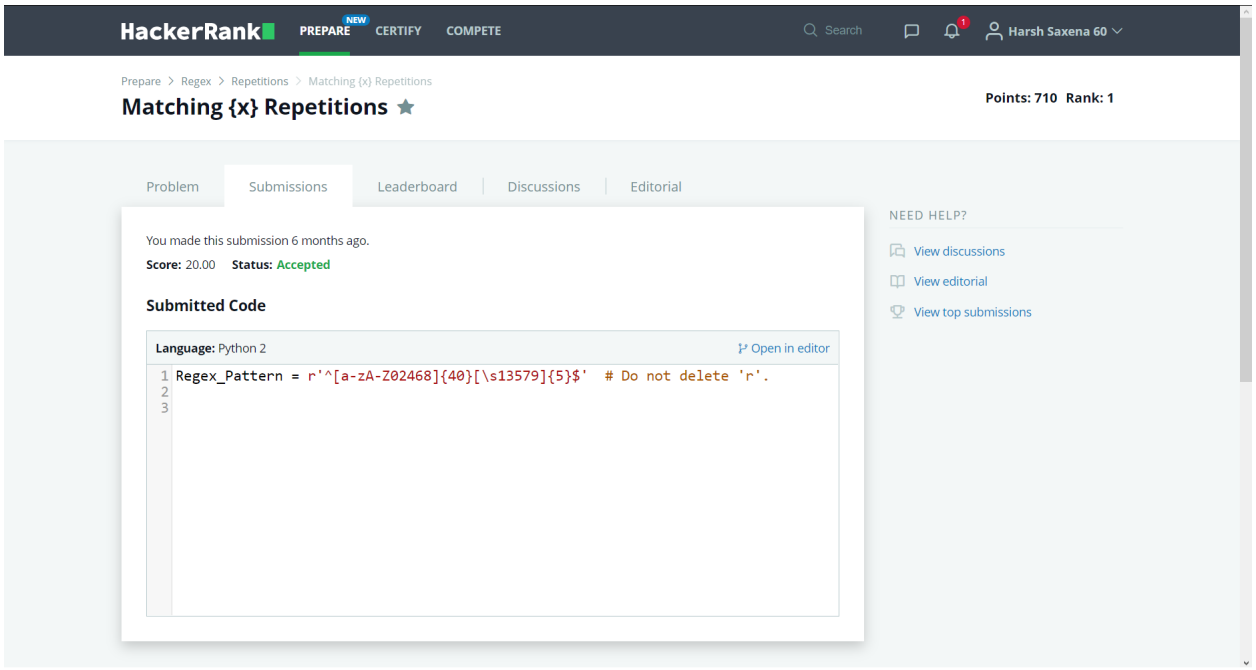
41.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Character Class > Excluding Specific Characters
Excluding Specific Characters
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 6 months ago.
Score: 10.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1
2
Regex_Pattern = r'^[^\d][^aeiou][^bcDF][^\s][^AEIOU][^.,]$' # Do not delete
'r'.
3
4
NEED HELP?
View discussions
View editorial
View top submissions

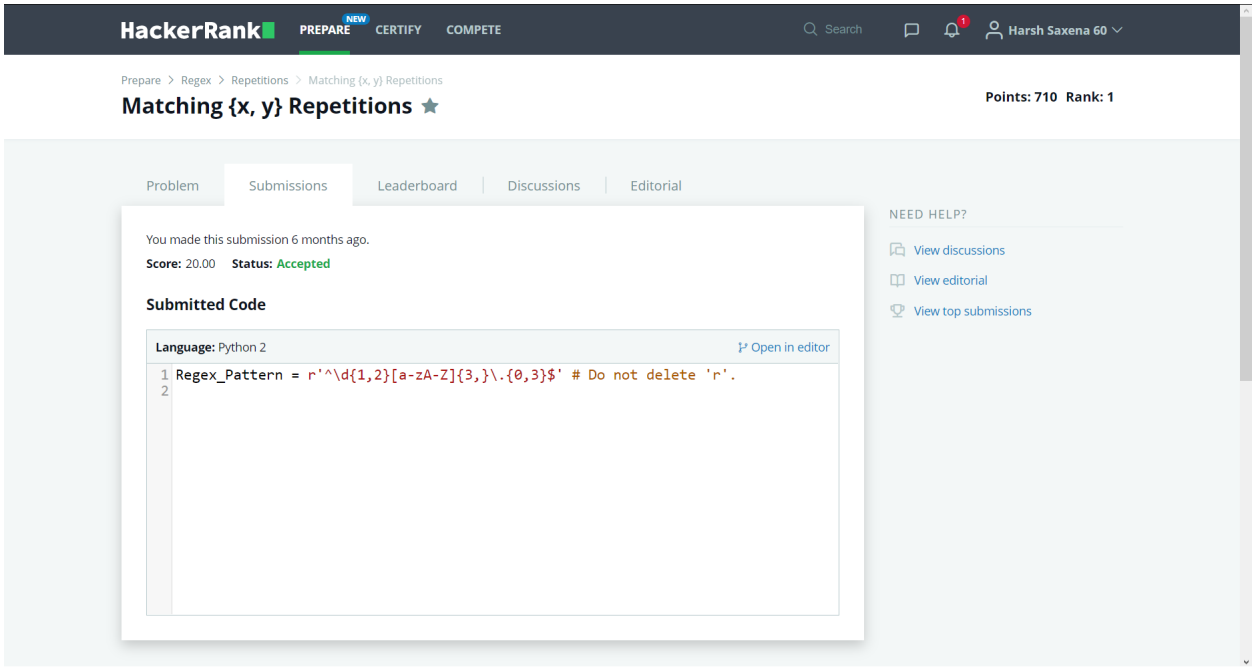
42.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Character Class > Matching Character Ranges
Matching Character Ranges
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 6 months ago.
Score: 10.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 Regex_Pattern = r'^[a-z][1-9][^a-z][^A-Z][A-Z].*' # Do not delete 'r'.
2
NEED HELP?
View discussions
View editorial
View top submissions

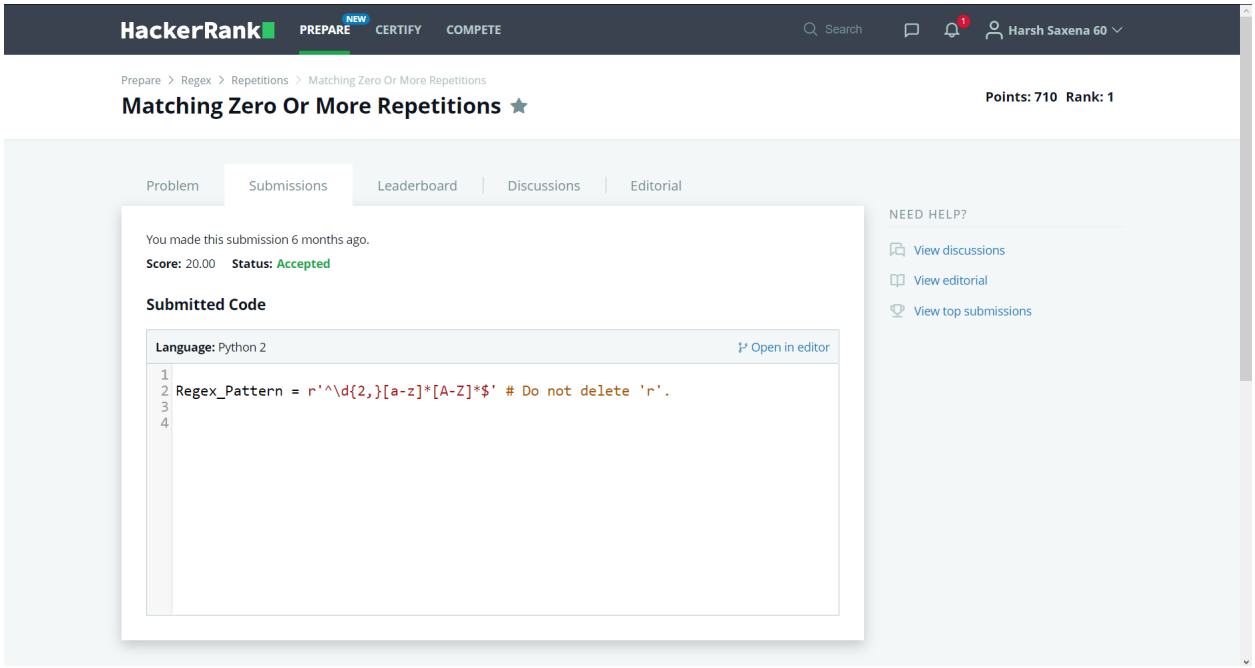
43.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Repetitions > Matching {x} Repetitions
Matching {x} Repetitions
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 6 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 Regex_Pattern = r'^[a-zA-Z02468]{40}[\s13579]{5}$' # Do not delete 'r'.
2
3
NEED HELP?
View discussions
View editorial
View top submissions

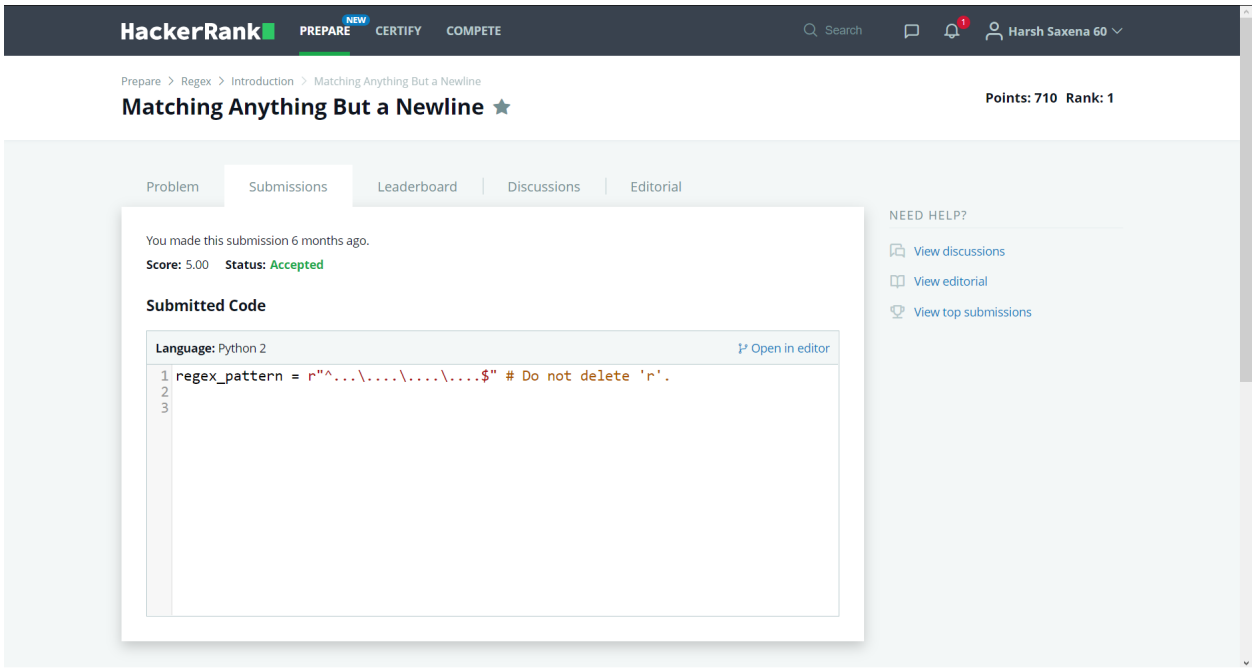
44.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
Prepare > Regex > Repetitions > Matching {x, y} Repetitions
Matching {x, y} Repetitions
Points: 710 Rank: 1
Problem
Submissions
Leaderboard
Discussions
Editorial
You made this submission 6 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 Regex_Pattern = r'^\d{1,2}[a-zA-Z]{3,}\.{0,3}$' # Do not delete 'r'.
2
NEED HELP?
View discussions
View editorial
View top submissions

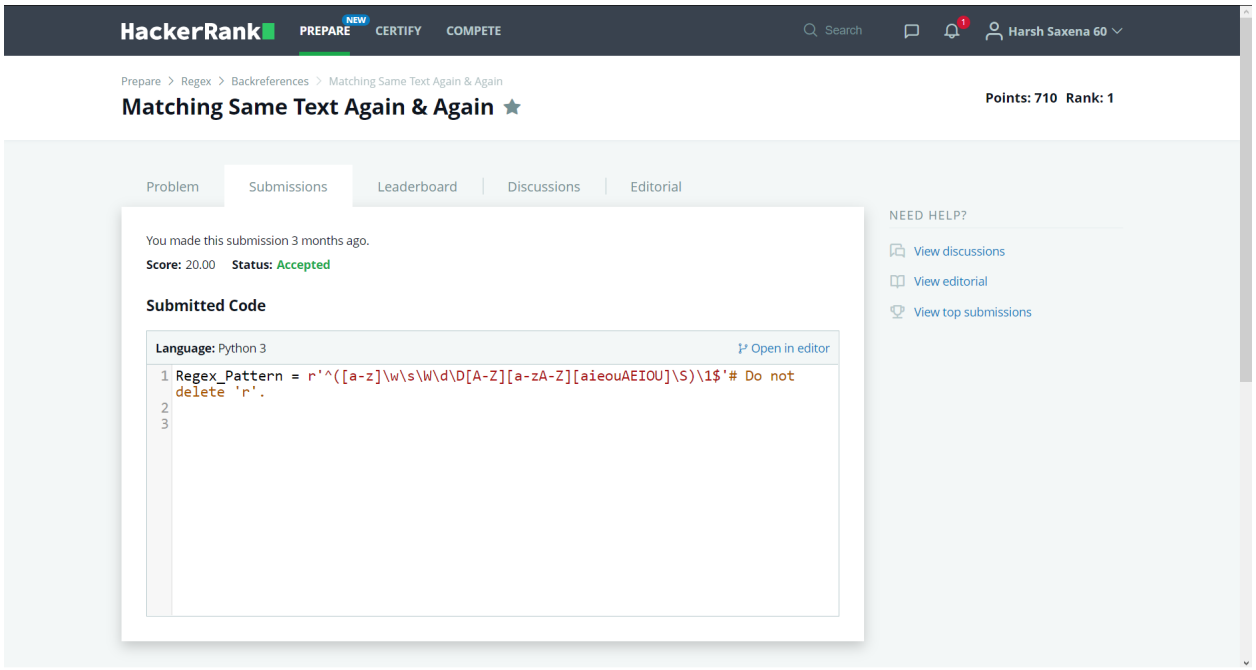
45.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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 6 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1
2
3
4
Regex_Pattern = r'^\d{2,}[a-z]*[A-Z]*$' # Do not delete 'r'.
NEED HELP?
View discussions
View editorial
View top submissions

46.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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 6 months ago.
Score: 5.00 Status: Accepted
Submitted Code
Language: Python 2
Open in editor
1 regex_pattern = r"^...\....\....\....$" # Do not delete 'r'.
2
3
NEED HELP?
View discussions
View editorial
View top submissions

47.


HackerRank
PREPARE
NEW
CERTIFY
COMPETE
Search
Harsh Saxena 60
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 3 months ago.
Score: 20.00 Status: Accepted
Submitted Code
Language: Python 3
Open in editor
1 Regex_Pattern = r'^([a-z]\w\s\W\d\D[A-Z][a-zA-Z][aieouAEIOU]\S)\1$'# Do not delete 'r'.
2
3
NEED HELP?
View discussions
View editorial
View top submissions

ASSIGNMENTS

Compiler Design Assignment - 1

ANTLR compiler TOOL

Harsh Saxena RA1911033010060

→ In computer based language recognition, ANTLR (another TOOL for Language Recognition) is a parser generator that uses LL for parsing.

→ It is the succesor of PCLTS

→ ANTLR takes as input a grammer that specifies a language and generates an output source code for recognizer of that language.

→ While version 3 supported generating code in the programming languages Ada 95, ActionScript, C, C#, Java, JavaScript, Objective-C, Perl, Python, Ruby and standard ML, version 4 at present targets C#, C++ and Dart, Java, JavaScript, GO, PHP, Python (2 & 3) and Swift.

→ A language is specified using a CFG (context Free Grammer) expressed using Extended Backus-Naur form

→ ANTLR can generate Lexers, Parsers, Tree Parsers, and combined Lexer-Parsers.

→ Parsers can automatically generate parser trees or abstract syntax trees which can be further processed with tree parsers.

→ ANTLR provides a single consistent notation for ~~sp~~ specifying lexers parsers and tree parsers.

→ By default, ANTLR reads a grammer and generates a recognizer for the language defined by the the grammer (i.e; a program that reads an input stream and generates an error if the input stream does not conform to the syntax specified by the grammer).

→ If there is no syntax errors, the default action is to ~~sy~~ simply exit without printing any message.

→ In order to do something useful with the language, actions can be attached to grammer elements. These actions are written in the programming language in which the recognizer is generated, the actions are embedded in the source code of the recognizer at the appropriate points.

→ Actions can be used to build and check symbol tables and to emit instructions in a target language, in the case of a compiler.

→ Other than lexers and parsers, ANTLR can be used to generate tree parsers. these are recognizers that process abstract syntax trees, which can be automatically generated by parsers.

→ these tree parser are unique to ANTLR and help processing abstract syntax trees.

Compiler Design Assignment - 2

Predictive Parsing

Harsh Saxena RA1911033010006

Q.] $S \rightarrow AaAb / BbBa$

$A \rightarrow \varepsilon \; ; B \rightarrow \varepsilon$

→ The grammar has got not left recurssion and left factoring

Calculation of FIRST & FOLLOW.

Symbol	FIRST	FOLLOW
S	{a, b}	{$}
A	ε	{a, b}
B	ε	{b, a}
a	a	
b	b.	

Predictive parsing table.

	a	b	$
S	S → AaAb	S → BbBa	
A	A → ε	A → ε	
B	B → ε	B → ε	

compiler design Assignment-3.

SLR.

Harsh Saxena RA1911033010060

$E \rightarrow E+T / T$

$T \rightarrow T*F / F$

$F(E) / id$.

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

$E \rightarrow E+T$

$E \rightarrow T$

$T \rightarrow T*F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow id$.

Augmented Grammar

$E' \rightarrow E$

$E \rightarrow E+T$

$E \rightarrow T$

$T \rightarrow T*F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow id$.

Closure of $[E' \rightarrow E]$

$E' \rightarrow \cdot E$
$E \rightarrow \cdot E+T$
$T \rightarrow \cdot T*F$
$T \rightarrow \cdot T*F$
$T \rightarrow \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot id$. I_0

Goto (I_0, E)
$E' \rightarrow E \cdot$
$E \rightarrow E \cdot +T$ I_1

Goto (I_0, T)
$T \rightarrow T \cdot *F$
$E \rightarrow T \cdot$ I_2

Goto (I_0, F)
$T \rightarrow F \cdot$ I_3

Goto I_0, (
$F \rightarrow (\cdot E)$
$E \rightarrow \cdot E+T$
$E \rightarrow \cdot T$
$T \rightarrow \cdot T*F$
$T \rightarrow \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot id$. I_4

Goto $(I_2, *)$
$T \rightarrow T* \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot id$. I_7

Goto I_0, id
$F \rightarrow id \cdot$ I_5

Goto $(I_1, +)$
$E \rightarrow E+ \cdot T$
$T \rightarrow \cdot T*F$
$T \rightarrow \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot id$ I_6

Goto (I_4, E)
$F \rightarrow (E \cdot)$
$E \rightarrow E \cdot +T$ I_8

Goto (I_7, F)
$T \rightarrow T*F \cdot$ I_{10}

Goto I_6, T
$E \rightarrow E+T \cdot$
$T \rightarrow T \cdot *F$ I_9

Goto $(I_8,)$
$F \rightarrow (E) \cdot$ I_{11}

SLR Parsing Table.

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

IP : id * id + id.

Stack	I/P	Action
0	id + id + id $	Shift
0 id 5	+ id + id $	Reduce F → id
0 F 3	* id + id $	Reduce by (4)
0 T_2	id + id $	Shift
0 T_2 * 7	id + id $	Shift
0 T2 * 7 id 5	+ id $	Reduce by (6)
0 T2 * 7 F_{10}	+ id $	Reduce by (3)
0 T_2	+ id $	Reduce by (2)
0 E_1	+ id $	Shift
0 E_1 + 6	id $	Shift
0 E_1 + 6 id 5	$	Reduce by (6)
0 E_1 + 6 F_3	$	Reduce by (2)
0 E_1 + 6 T 9	$	Reduce by (1)
0 E_1	$	Accept

Compiler Design Assignment - 4

Cross Compilers

Harsh Saxena RA1911033010106D

→ A cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is running.

→ For exq: A compiler that runs on a PC but generates code that runs on an Android Smartphone is a guest compiler.

→ A cross compiler is necessary to compile code for multiple platforms from one development host. Direct compilation on the target platform might be infeasible for example on embedded systems with limited computing resources.

→ Cross compilers are distinct for source-to-source compilers. A cross compilers is for cross-platform software generation of machine code. While a source to source compiler translates from one programming language to another in text code. Both are programming tools.

→ The fundamental use of a cross compiler is to seperate the built environment from target environment. This is useful is several situations:-

→ Embedded computer where a computer has extremely limited resources

→ compiling for multiple machines. For example, a company may wish to support different versions of a operating systems and for different operating syscams
→ compiling on a server farm
→ Bootstrapping to a new platform
→ compiling native code for emulators for older new-obsolete platforms like the commodate-64 or Apple II.

Compiler Design Assignment-5
Storage Allocation.

Harsh Saxena RA1110330100060

The various storage allocation techniques are as follows.

Static Allocation :-

→ Simplest allocation scheme in which allocation of data objects is done at compile time because the size of every data item can be determined by compiler.

:→ Recursive subprogram and arrays of adjustable length and not permitted in a language. In static allocation, the compiler can decide the amount of storage needed by each data object. Thus it becomes ~~each data object~~ easy for a compiler to identify the address of these data in activation record.

Advantage :-

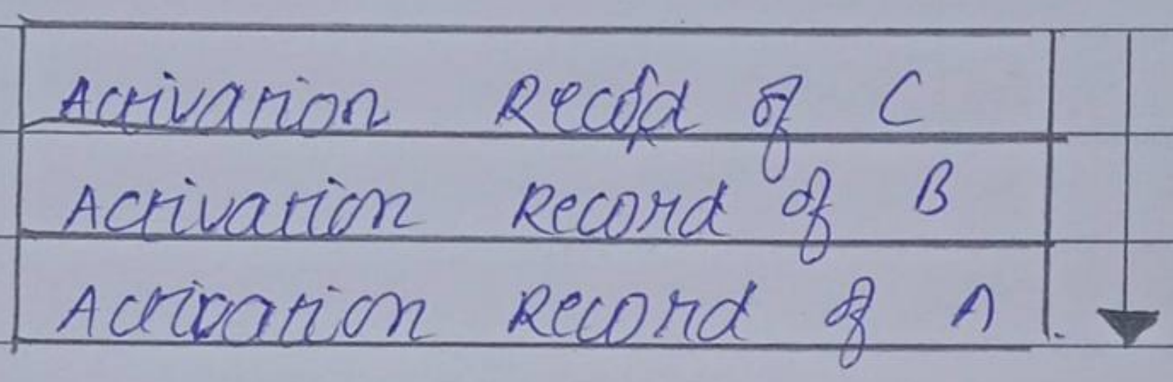
→ Easy to implement

→ Allows type checking

→ Dynamic Allocation

→ The stack allocation is a runtime storage management technique. The activation records are

pushed and ~~pf~~ popped as activation record is pushed onto the stack when the call is made.
→ It can be used to determined the size of the variables at a runtime and hence local variables can have different storage locations and different values during various activations allows recursive subprograms



Stack Allocation.

Heap storage Allocation
→ Enables the allocation of memory in a non-nested design - storage can be allocated and freed arbitrarily from an area.
→ Heap is maintained as a list of free space called free space list.