STUDENT PORTFOLIO

Name: Yuvraj Singh Chauhan
Register Number: RA1911027010058
Mail ID: yc4823@srmist.edu.in
Department: CSE
Specialization: Bigdata
Semester: VI

Subject Title: 18CSC304J Compiler Designs

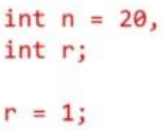
Handled By: Sharanya S

Assignments

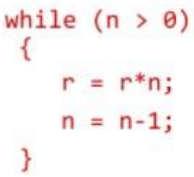
Assignments were given to students as a part of course for many topics of chapters. Total 4 assignments were given which the students are supposed to solve. These assignments are considered very important and our faculty provided us the necessary practice by providing these assignments.

Assignment 1

1. Tokenize the code snippet attached below.
2. Illustrate all the compilation phase for the instruction: z=x/5+y/3



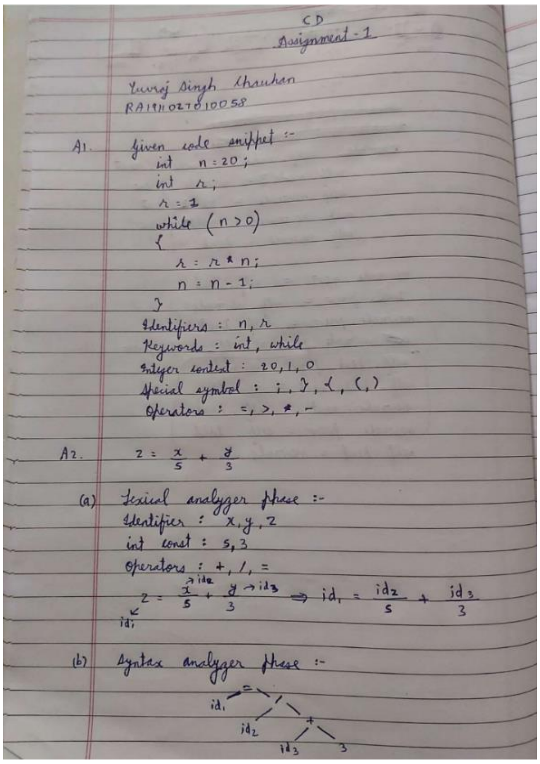
```
int n = 20,
int r;

r = 1;
```



```
while (n > 0)
 {
    r = r*n;
    n = n-1;
 }
```

This assignment focused on lexical analyser where tokens are generated and also the instruction is written for compilation phase which are:

Lexical analyser
Semantic analyser
Intermediate Code generator
Code Optimizer
Final Code Generator



CD
Assignment - 1

Yuvraj Singh Chauhan
RA1911027010058

A1. Given code snippet :-

```
int  n = 20;
int  r;
r = 1
while (n > 0)
{
   r = r * n;
   n = n - 1;
}
```

Identifiers : n, r
Keywords : int, while
Integer content : 20, 1, 0
Special symbol : ;, }, {, (,)
Operators : =, >, *, -

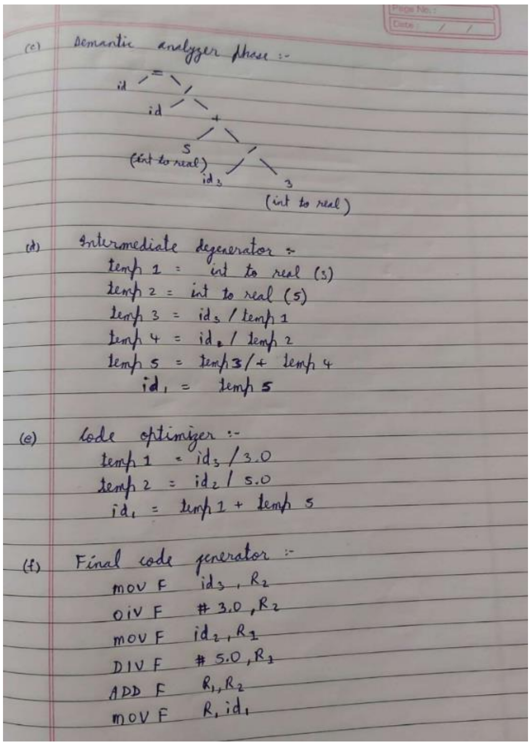
A2. $z = \frac{x}{5} + \frac{y}{3}$

(a) Lexical analyzer phase :-
Identifier : x, y, z
int const : 5, 3
Operators : +, /, =

$\underset{id_1}{z} = \frac{\overset{\rightarrow id_2}{x}}{5} + \frac{y}{3} \rightarrow id_3 \Rightarrow id_1 = \frac{id_2}{5} + \frac{id_3}{3}$

(b) Syntax analyzer phase :-

=
id₁
/
id₂
+
id₃
3



(c) Semantic analyzer phase :-

=
id
/
id
+
5
(int to real)
/
id₃
3
(int to real)

(d) Intermediate degenerator :-
temp 1 = int to real (3)
temp 2 = int to real (5)
temp 3 = id_3 / temp 1
temp 4 = id_2 / temp 2
temp 5 = temp3/+ temp 4
id_1 = temp 5

(e) Code optimizer :-
temp 1 = id_3 / 3.0
temp 2 = id_2 / 5.0
id_1 = temp 1 + temp 5

(f) Final code generator :-

```
mov F   id3 , R2
OIV F   # 3.0 , R2
mov F   id2 , R1
DIV F   # 5.0 , R1
ADD F   R1, R2
mov F   R, id1
```

(1)
(2)

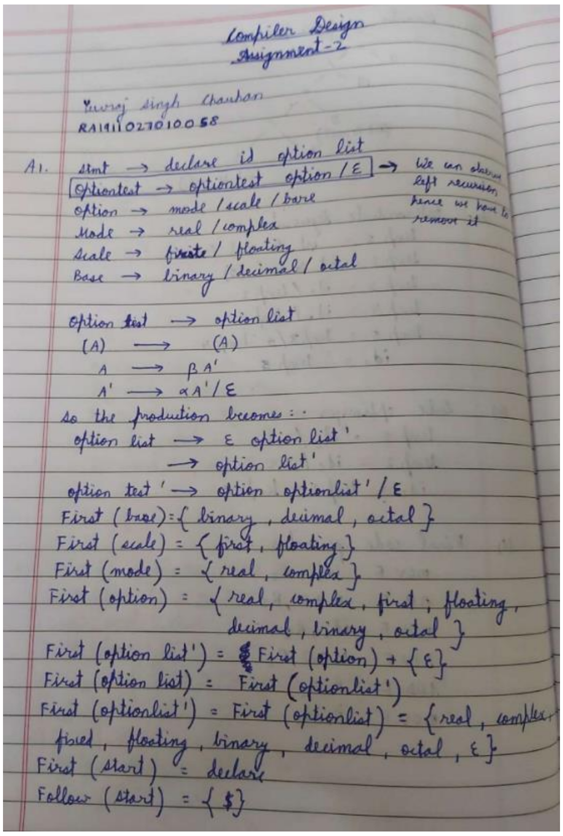
ASSIGNMENT 2
PROBLEM STATEMENT

1. Perform predictive parsing.
   Stmt→ **declare id** *optionList*
   optionList→ *optionList option* |ε
   option→*mode*|*scale*|*base*
   mode→**real|complex**
   scale→**fixed|floating**
   base→**binary|decimal|octal**

2. Derive the sentence " A lion saw the deer under the tree" from the following grammar using top down and bottom-up parse trees.

   S→ NP VP
   NP→DT N| NP PP
   PP→PRP NP
   VP→V NP| VP PP
   DT→'a'| 'the'
   N→'Lion'|'deer'| 'tree'
   PRP→'under' |'with' |'above'
   V→'ate'|'saw'|'ran'

IN THIS ASSIGNMENT WE WERE ASKED TO PERFORM PREDICTIVE PARSING FOR A GIVEN gRAMMAR. ALSO TOP DOWN AND BOTTOM-UP PARSERS WERE REVISED BY ASKING US TO DERIVER A SENTENCE USING GIVEN GRAMMAR IN BOTH PARSER TREES.



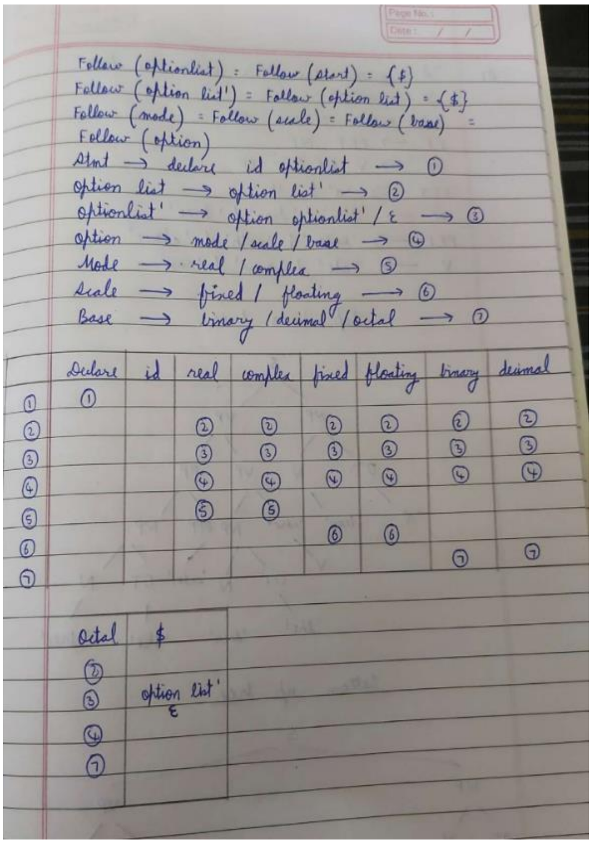
Compiler Design
Assignment-2

Yuvraj singh Chauhan
RA1911027010058

A1. stmt → declare id option list
optiontest → optiontest option / ε → We can observe left recursion, hence we have to remove it
option → mode / scale / base
Mode → real / complex
Scale → finite / floating
Base → binary / decimal / octal

option test → option list
(A) ⟶ (A)
A ⟶ β A'
A' ⟶ α A' / ε
so the production becomes : ·
option list ⟶ ε option list'
⟶ option list'
option test ' ⟶ option optionlist' / ε
First (base) = { binary, decimal, octal }
First (scale) = { first, floating }
First (mode) = { real, complex }
First (option) = { real, complex, first ; floating, decimal, binary, octal }
First (option list') = First (option) + { ε }
First (option list) = First (optionlist')
First (optionlist') = First (optionlist) = { real, complex, fixed, floating, binary, decimal, octal, ε }
First (start) = declare
Follow (start) = { $ }

(1)

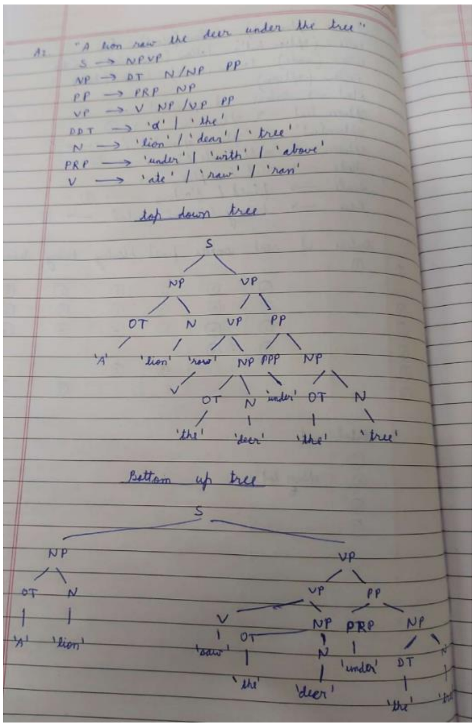


Follow (optionlist) = Follow (start) = { $ }
Follow (option list') = Follow (option list) = { $ }
Follow (mode) = Follow (scale) = Follow (base) =
Follow (option)
Stmt ⟶ declare id optionlist ⟶ ①
option list ⟶ option list' ⟶ ②
optionlist' ⟶ option optionlist' / ε ⟶ ③
option ⟶ mode / scale / base ⟶ ④
Mode ⟶ real / complex ⟶ ⑤
Scale ⟶ fixed / floating ⟶ ⑥
Base ⟶ binary / decimal / octal ⟶ ⑦

	Declare	id	real	complex	fixed	floating	binary	decimal
①	①							
②			②	②	②	②	②	②
③			③	③	③	③	③	③
④			④	④	④	④	④	④
⑤			⑤	⑤				
⑥					⑥	⑥		
⑦							⑦	⑦

Octal	$
②	
③	option list' ε
④	
⑦	

(2)



A2. "A lion saw the deer under the tree"
S ⟶ NP VP
NP ⟶ DT N / NP PP
PP ⟶ PRP NP
VP ⟶ V NP / VP PP
DDT ⟶ 'a' | 'the'
N ⟶ 'lion' / 'dear' / 'tree'
PRP ⟶ 'under' | 'with' | 'above'
V ⟶ 'ate' / 'saw' / 'ran'

top down tree

Bottom up tree

(3)

Assignment 3

Problem Statement

1. Compare your observations and inferences on SLR, CLR and LALR parsing on the grammar:
   Course_1→Course_2Course_3| discrete_ mathematics

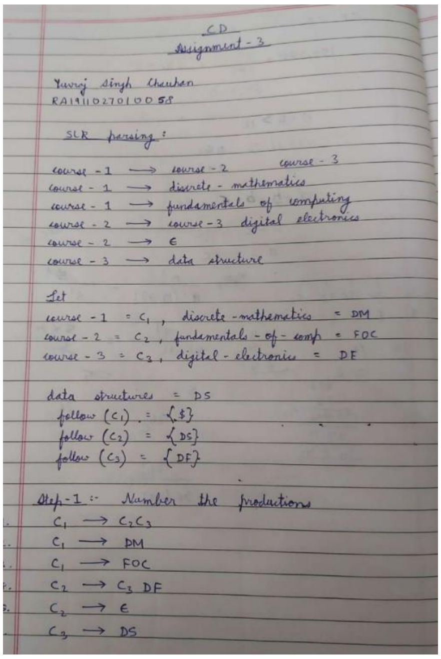
   |fundamenrals_of_computing

   Course_2→Course_3 digiral_electronics | ε

   Course_3→ data_structures

   Give your inferences in tabular form after working out the grammar.

This assignment was given to have practice on SLR, CLR and LALR. We were asked to compare the observations we got after these parsing grammars.



CD
Assignment - 3

Yuvraj Singh Chauhan
RA1911027010058

SLR parsing :

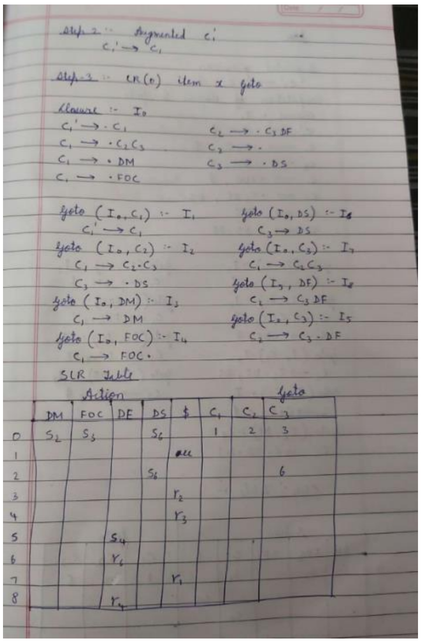
course - 1 ⟶ course - 2 course - 3
course - 1 ⟶ discrete - mathematics
course - 1 ⟶ fundamentals of computing
course - 2 ⟶ course - 3 digital electronics
course - 2 ⟶ ϵ
course - 3 ⟶ data structure

Let
course - 1 = C_1 , discrete - mathematics = DM
course - 2 = C_2 , fundamentals - of - comp = FOC
course - 3 = C_3 , digital - electronics = DE

data structures = DS
follow (C_1) = {$}
follow (C_2) = {DS}
follow (C_3) = {DF}

Step-1 :- Number the productions
1. $C_1 \longrightarrow C_2 C_3$
2. $C_1 \longrightarrow DM$
3. $C_1 \longrightarrow FOC$
4. $C_2 \longrightarrow C_3\ DF$
5. $C_2 \longrightarrow \epsilon$
6. $C_3 \longrightarrow DS$

(1)



Step 2 :- Augmented C_1
$C_1' \rightarrow C_1$

Step-3 :- LR(0) item & goto

Closure :- I_0
$C_1' \rightarrow \cdot C_1$
$C_1 \rightarrow \cdot C_2 C_3$
$C_1 \rightarrow \cdot DM$
$C_1 \rightarrow \cdot FOC$
$C_2 \rightarrow \cdot C_3 DF$
$C_2 \rightarrow \cdot$
$C_3 \rightarrow \cdot DS$

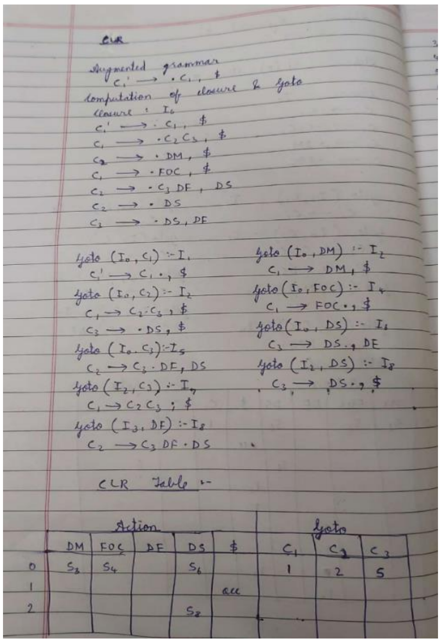
goto (I_0, C_1) :- I_1
$C_1' \rightarrow C_1$
goto (I_0, C_2) :- I_2
$C_1 \rightarrow C_2 \cdot C_3$
$C_3 \rightarrow \cdot DS$
goto (I_0, DM) :- I_3
$C_1 \rightarrow DM$
goto (I_0, FOC) :- I_4
$C_1 \rightarrow FOC \cdot$

goto (I_0, DS) :- I_6
$C_3 \rightarrow DS$
goto (I_0, C_3) :- I_7
$C_1 \rightarrow C_2 C_3$
goto (I_5, DF) :- I_8
$C_2 \rightarrow C_3 DF$
goto (I_2, C_3) :- I_5
$C_2 \rightarrow C_3 \cdot DF$

SLR Table

	Action					goto		
	DM	FOC	DE	DS	$	C_1	C_2	C_3
0	S_2	S_3		S_6		1	2	3
1					acc			
2				S_6				6
3					r_2			
4					r_3			
5			S_4					
6			r_6					
7					r_1			
8			r_4					

(2)



CLR

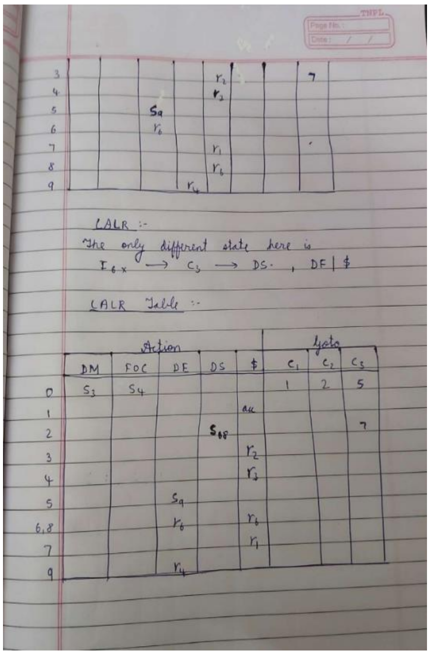
Augmented grammar
$C_1' \longrightarrow \cdot C_1 , \$$
Computation of closure & goto
Closure : I_0
$C_1' \longrightarrow \cdot C_1 , \$$
$C_1 \longrightarrow \cdot C_2 C_3 , \$$
$C_2 \longrightarrow \cdot DM , \$$
$C_1 \longrightarrow \cdot FOC , \$$
$C_2 \longrightarrow \cdot C_3 DF , DS$
$C_2 \longrightarrow \cdot DS$
$C_3 \longrightarrow \cdot DS , DF$

goto (I_0, C_1) :- I_1
$C_1' \longrightarrow C_1 \cdot , \$$
goto (I_0, C_2) :- I_2
$C_1 \rightarrow C_2 \cdot C_3 , \$$
$C_3 \rightarrow \cdot DS , \$$
goto (I_0. C_3) :- I_5
$C_2 \rightarrow C_3 \cdot DF , DS$
goto (I_2, C_3) :- I_7
$C_1 \rightarrow C_2 C_3 ; \$$
goto (I_3, DF) :- I_8
$C_2 \longrightarrow C_3 DF \cdot DS$

goto (I_0, DM) :- I_2
$C_1 \longrightarrow DM , \$$
goto (I_0, FOC) :- I_4
$C_1 \rightarrow FOC \cdot , \$$
goto (I_0, DS) :- I_6
$C_3 \longrightarrow DS \cdot , DE$
goto (I_2, DS) :- I_8
$C_3 \longrightarrow DS \cdot , \$$

CLR Table :-

(3)



	Action					goto		
	DM	FOC	DE	DS	$	C_1	C_2	C_3
0	S_3	S_4		S_6		1	2	5
1					acc			
2				S_8				
3					r_2			7
4					r_3			
5			S_9					
6			r_6					
7					r_1			
8					r_6			
9				r_4				

LALR :-
The only different state here is
$I_{6x} \longrightarrow C_3 \longrightarrow DS \cdot , DF \mid \$$

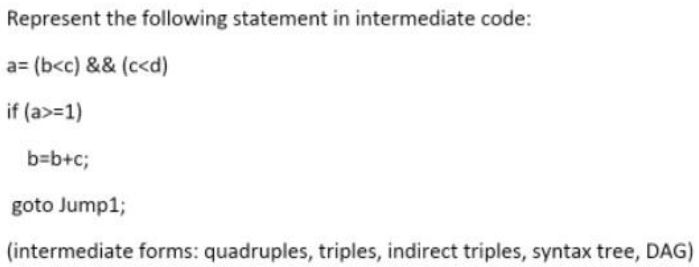
LALR Table :-

	Action					goto		
	DM	FOC	DE	DS	$	C_1	C_2	C_3
0	S_3	S_4				1	2	5
1					acc			
2				S_{68}				7
3					r_2			
4					r_3			
5			S_9					
6,8			r_6		r_6			
7					r_1			
9			r_4					

(4)

Assignment 4

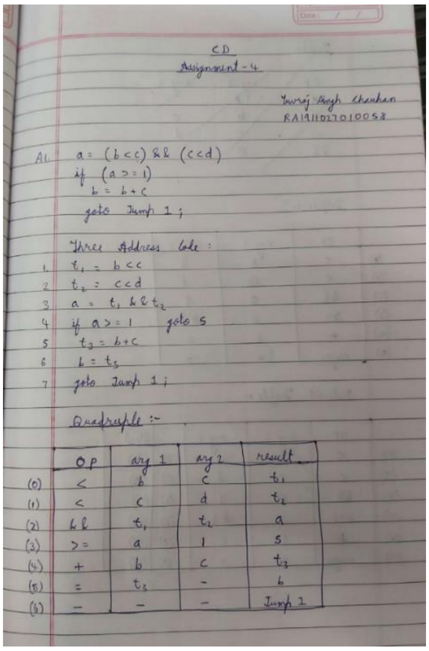
This assignment was given so that intermediate code for a particular code can be written. The question above has a code written and the intermediate code was asked. Intermediate forms are as follows: Quadruples, triples, indirect triples, Syntax tree, DAG.



Represent the following statement in intermediate code:

```
a= (b<c) && (c<d)
if (a>=1)
   b=b+c;
 goto Jump1;
```

(intermediate forms: quadruples, triples, indirect triples, syntax tree, DAG)



CD

Assignment - 4

Yuvraj Singh Chauhan
RA1911027010058

A1.
```
a = (b<c) && (c<d)
if (a>=1)
  b = b+c
goto Jump 1;
```

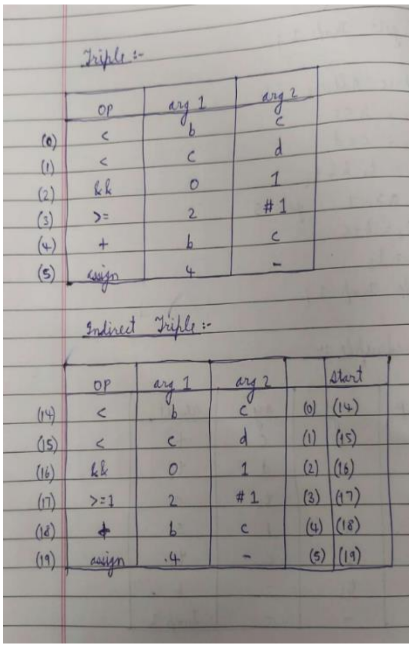
Three Address Code :

```
1  t1 = b<c
2  t2 = c<d
3  a = t1 && t2
4  if a>=1    goto 5
5   t3 = b+c
6   b = t3
7  goto Jump 1;
```

Quadruple :-

	op	arg 1	arg 2	result
(0)	<	b	c	t_1
(1)	<	c	d	t_2
(2)	&&	t_1	t_2	a
(3)	>=	a	1	5
(4)	+	b	c	t_3
(5)	=	t_3	-	b
(6)	-	-	-	Jump 1

(1)



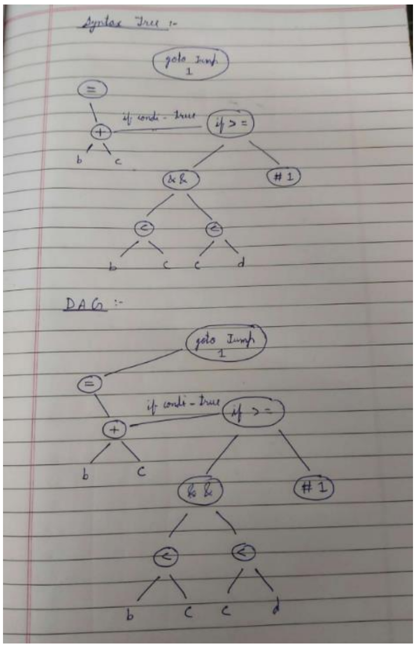
Triple :-

	op	arg 1	arg 2
(0)	<	b	c
(1)	<	c	d
(2)	&&	0	1
(3)	>=	2	#1
(4)	+	b	c
(5)	assign	4	-

Indirect Triple :-

	op	arg 1	arg 2		Start
(14)	<	b	c	(0)	(14)
(15)	<	c	d	(1)	(15)
(16)	&&	0	1	(2)	(16)
(17)	>=1	2	#1	(3)	(17)
(18)	+	b	c	(4)	(18)
(19)	assign	.4	-	(5)	(19)

(2)



Syntax Tree :-

DAG :-

(3)

Lab Record

This course has lab component where we made use of our theory knowledGe, we learnt in the class to write proGrammes to implement those theory concepts. StartinG from lexical analyser to Intermediate Code. Below u can find Title, Aim, code, output for all the lab done.

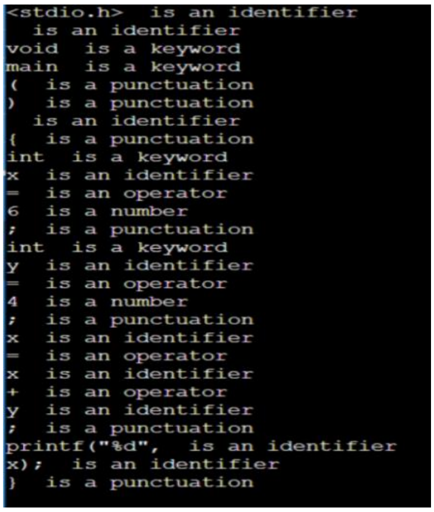
1. Implementation of Lexical analyser

Aim: To implement a lexical analyser based on the given problem.

Source Code:

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

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

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

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

Output:



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

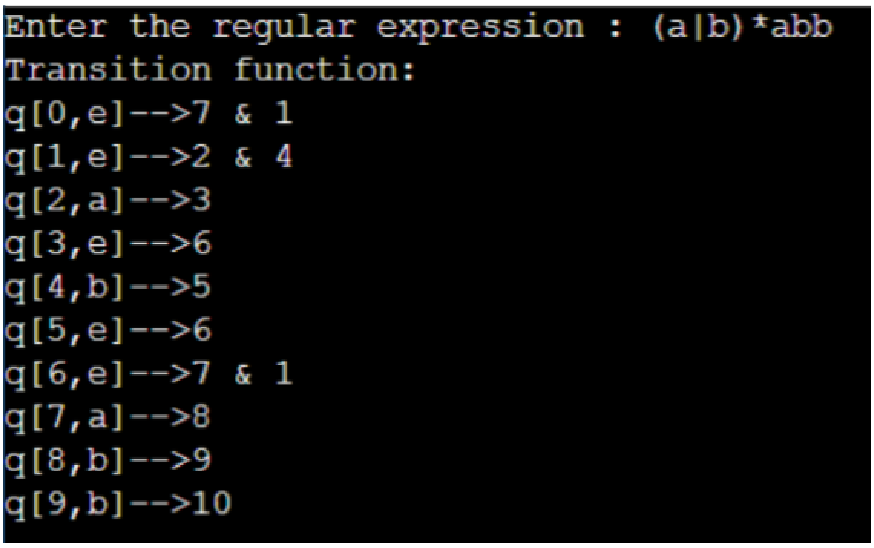
2.Conversion of reGular Expression To NFA

Aim: To convert a regular expression to NFA

Source Code:

```
transition_table = [ [0] *3 for _ in range (20)]

re = input ("Enter the regular expression: ")
re += " "

i = 0
j = 1
while(i<len(re)):
    if re[i] == 'a':
        try:
            if re[i+1] != '|' and re[i+1] !='*':
                transition_table[j][0] = j+1
                j += 1
            elif re[i+1] == '|' and re[i+2] =='b':
                transition_table[j][2]=((j+1)*10)+(j+3)
                j+=1
                transition_table[j][0]=j+1
                j+=1
                transition_table[j][2]=j+3
                j+=1
                transition_table[j][1]=j+1
                j+=1
                transition_table[j][2]=j+1
                j+=1
                i=i+2
            elif re[i+1]=='*':
                transition_table[j][2]=((j+1)*10)+(j+3)
                j+=1
                transition_table[j][0]=j+1
                j+=1
                transition_table[j][2]=((j+1)*10)+(j-1)
                j+=1
        except:
            transition_table[j][0] = j+1

    elif re[i] == 'b':
        try:
            if re[i+1] != '|' and re[i+1] !='*':
                transition_table[j][1] = j+1
                j += 1
            elif re[i+1]=='|' and re[i+2]=='a':
                transition_table[j][2]=((j+1)*10)+(j+3)
                j+=1
                transition_table[j][1]=j+1
                j+=1
                transition_table[j][2]=j+3
                j+=1
                transition_table[j][0]=j+1
                j+=1
                transition_table[j][2]=j+1
                j+=1
                i=i+2
            elif re[i+1]=='*':
                transition_table[j][2]=((j+1)*10)+(j+3)
                j+=1
                transition_table[j][1]=j+1
                j+=1
                transition_table[j][2]=((j+1)*10)+(j-1)
                j+=1
        except:
            transition_table[j][1] = j+1

    elif re[i]=='e' and re[i+1]!='|'and re[i+1]!='*':
        transition_table[j][2]=j+1
        j+=1

    elif re[i]==')' and re[i+1]=='*':

        transition_table[0][2]=((j+1)*10)+1
        transition_table[j][2]=((j+1)*10)+1
        j+=1

    i +=1

print ("Transition function:")
for i in range(j):
    if(transition_table[i][0]!=0):
        print("q[{0},a]-->{1}".format(i,transition_table[i][0]))
    if(transition_table[i][1]!=0):
        print("q[{0},b]-->{1}".format(i,transition_table[i][1]))
    if(transition_table[i][2]!=0):
        if(transition_table[i][2]<10):
            print("q[{0},e]-->{1}".format(i,transition_table[i][2]))
        else:
            print("q[{0},e]-->{1} & {2}".format(i,int(transition_table[i][2]/10),transition_table[i][2]%10))
```

Output:



```
Enter the regular expression : (a|b)*abb
Transition function:
q[0,e]-->7 & 1
q[1,e]-->2 & 4
q[2,a]-->3
q[3,e]-->6
q[4,b]-->5
q[5,e]-->6
q[6,e]-->7 & 1
q[7,a]-->8
q[8,b]-->9
q[9,b]-->10
```

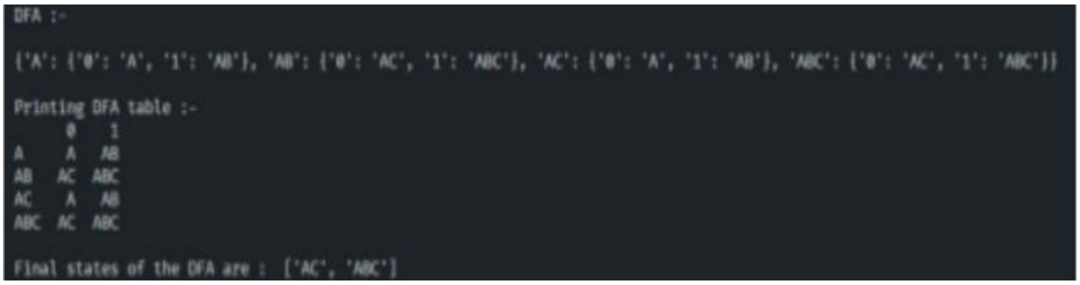
3.CONVERSION OF NFA TO DFA

Aim: To convert a NFA to DFA based on the given problem.

Source Code:

```
import pandas as pd

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

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

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

new_states_list = []

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

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

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

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

    new_states_list.remove(new_states_list[0])

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

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

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

Ouput:



```
DFA :-

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

Printing DFA table :-
      0    1
A     A   AB
AB   AC  ABC
AC    A   AB
ABC  AC  ABC

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

4. ELIMINATION OF LEFT RECURSION AND LEFT FACTORING

Aim:

1. To remove Left Recursion for given production
2. To remove Left Factoring for given production

Source Code:

Left Recursion:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
int n=1,i=0,j=0,k=0;
char a[10][10],f[10];
```

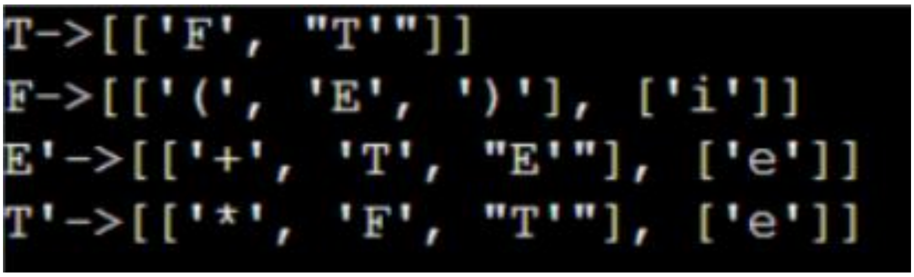
```
 int main(){
   int i=0,z;
   char c,ch;

     printf("Enter the production:\n");
     for(i=0;i<n;i++)
        scanf("%s%c",a[i],&ch);

     c=a[0][0];
     if(a[0][2] == c)
     {
       printf("Left recursion found: \n");
       printf("%c' -> ",c);
       for(k=3;k<strlen(a[0]) && a[0][k] != '|';k++)
       {
                  printf("%c",a[0][k]);
                }
                printf("%c' | e",c);
                n=k; i=0;
                printf("\n%c ->",c);
                for(k=n+1; k<strlen(a[0]) && a[0][k]!='\0'; k++)
                {
                          printf("%c",a[0][k]);
                }
                printf("%c'",c);
     }
     else{
             printf("No left recursion!!");
               }
      return 0;
   }
```

Left Factoring

```
#include<bits/stdc++.h>
using  namespace  std;
int main()
{
    char a[10],a1[10],a2[10],a3[10],a4[10],a5[10];
    int i ,j=0,k,l;
    cout<<"Enter any productions A->";
    cin>>a;
    for(i=0;a[i]!='|';i++,j++)
    a1[j]=a[i];
    a1[j]='\0';
    for(j=++i,i=0;a[j]!='\0';j++,i++)
    a2[i]=a[j];
    a2[i]='\0';
    k=0;l=0;
    for(i=0;i<strlen(a1)||i<strlen(a2);i++)
    {
        if(a1[i]==a2[i])
        a3[k++]=a1[i];
        else
        {
            a4[l]=a1[i];
            a5[l]=a2[i];
            l++;
        }
    }
        a3[k]='X'; a3[++k]='\0';
        a4[l]='|'; a5[l]='\0';
        a4[++l]='\0';
        strcat(a4,a5);
        cout<<"\n A->"<<a3;
        cout<<"\n X->"<<a4;
        return 0;
}
```

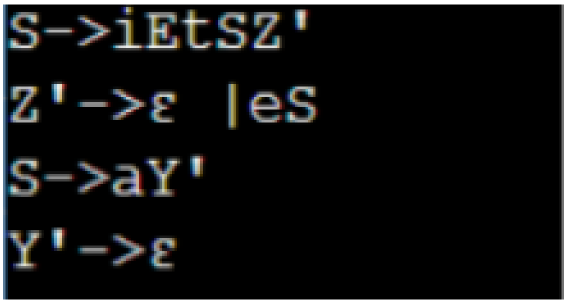
Output:

Left Recursion



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

Left Factoring



```
S->iEtSZ'
Z'->ε |eS
S->aY'
Y'->ε
```

5. Computation of FIRST and FOLLOW

Aim: To Compute First () and Follow () based on the given problem.

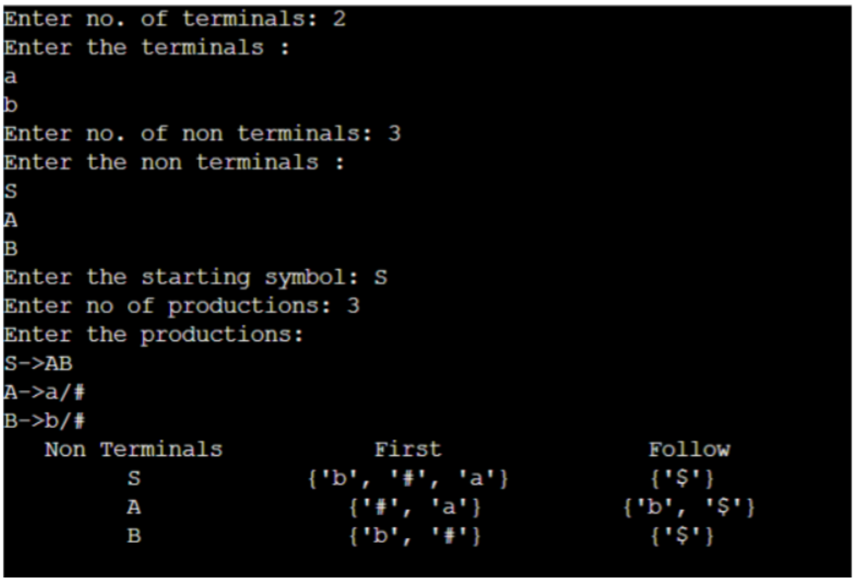
Source Code:

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
int n,m=0,p,i=0,j=0;
```

```
char a[10][10],f[10];
void follow(char c);
void first(char  c);
int main(){
   int i,x;
   char c,ch;
   printf("No of productions:\n");
   scanf("%d",&n);
   printf("Enter the productions:\n");
   for(i=0;i<n;i++){
      scanf("%s%c",a[i],&ch);
   }
   do{
      m=0;
      printf("Enter the elements whose first & foloow is to be found:");
      scanf("%c",&c);
      first(c);
      printf("First(%c)={",c);
      for(i=0;i<m;i++){
         printf("%c",f[i]);
      }
      printf("}\n");
      strcpy(f," ");
      m=0;
      follow(c);
      printf("Follow(%c)={",c);
      for(i=0;i<m;i++){
         printf("%c",f[i]);
      }
      printf("}\n");
      printf("Continue(0/1)?");
      scanf("%d%c",&x,&ch);
   }
   while(x==1);
   return(0);
}

void first(char c){
   int k;
   if(!isupper(c))
   f[m++]=c;
   for(k=0;k<n;k++){
      if(a[k][0]==c){
         if(a[k][2]=='$'){
            follow(a[k][0]);
         }
         else if(islower(a[k][2])){
            f[m++]=a[k][2];
         }
         else first(a[k][2]);
```

```
        }
    }
}
void follow(char c){
    if(a[0][0]==c){
        f[m++]='$';
    }
    for(i=0;i<n;i++){
        for(j=2;j<strlen(a[i]);j++){
            if(a[i][j]==c){
                if(a[i][j+1]!='\0'){
                    first(a[i][j+1]);
                }
                if(a[i][j+1]=='\0' && c!=a[i][0]){
                    follow(a[i][0]);
                }
            }
        }
    }
}
```

Output:



```
Enter no. of terminals: 2
Enter the terminals :
a
b
Enter no. of non terminals: 3
Enter the non terminals :
S
A
B
Enter the starting symbol: S
Enter no of productions: 3
Enter the productions:
S->AB
A->a/#
B->b/#
   Non Terminals           First               Follow
         S           {'b', '#', 'a'}          {'$'}
         A              {'#', 'a'}          {'b', '$'}
         B              {'b', '#'}            {'$'}
```

6. Construction of Predictive ParsinG table

Aim: To construct a predictive parsing Table for an inputted grammar.

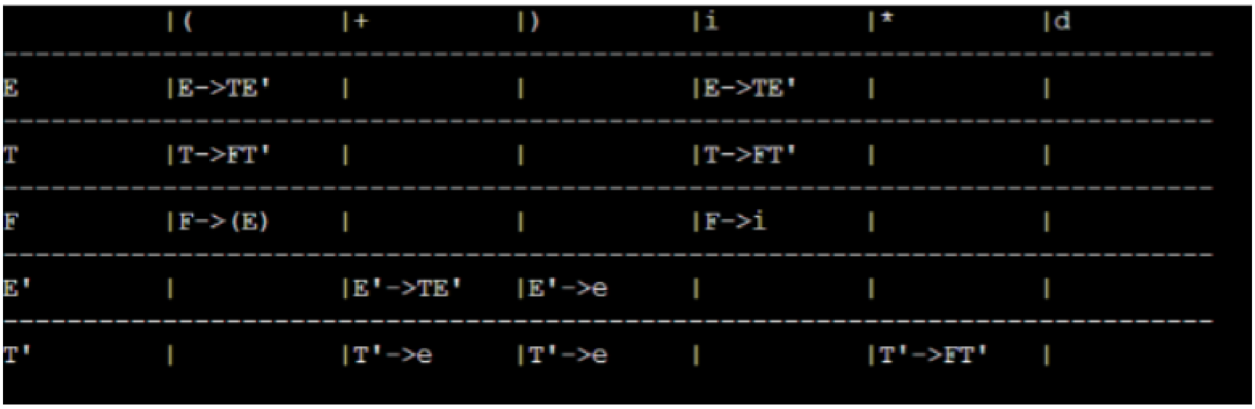
Source Code:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    char fin[10][20], st[10][20], ft[20][20], fol[20][20];
    int a, i, t, b, n, j, s = 0, p;
    cout << "Number of productions: ";
```

```
cin >> n;
cout << "Productions of the grammar:\n";
for (i = 0; i < n; i++)
    cin >> st[i];
cout << "\nEnter the FIRST and FOLLOW of each non-terminal:";
for (i = 0; i < n; i++)
{
    cout << "\nFIRST[" << st[i][0] << "] : ";
    cin >> ft[i];
    cout << "FOLLOW[" << st[i][0] << "] : ";
    cin >> fol[i];
}
cout << "\nThe contents of the predictive parser table are:\n";
for (i = 0; i < n; i++)
{
    j = 3;
    while (st[i][j] != '\0')
    {
        if (st[i][j - 1] == '|' || j == 3)
        {
            for (p = 0; p <= 2; p++)
                fin[s][p] = st[i][p];
            t = j;
            for (p = 3; st[i][j] != '|' && st[i][j] != '\0'; p++, j++)
                fin[s][p] = st[i][j];
            fin[s][p] = '\0';
            if (st[i][t] == 'e')
            {
                a = b = 0;
                while (st[a++][0] != st[i][0])
                    ;
                while (fol[a][b] != '\0')
                {
                    cout << "M[" << st[i][0] << "," << fol[a][b]
                         << "] = " << fin[s] << "\n";
                    b++;
                }
            }
            else if (!(st[i][t] > 64 && st[i][t] < 91))
                cout << "M[" << st[i][0] << "," << st[i][t]
                     << "] = " << fin[s] << "\n";
            else
            {
                a = b = 0;
                while (st[a++][0] != st[i][3])
                    ;
                while (ft[a][b] != '\0')
                {
                    cout << "M[" << st[i][0] << "," << ft[a][b]
                         << "] = " << fin[s] << "\n";
                    b++;
                }
            }
            s++;
        }
```

```
            if (st[i][j] == '|')
                j++;
        }
    }
    return 0;
}
```

Output:



```
         |(        |+        |)        |i        |*        |d
---------------------------------------------------------------------
E        |E->TE'   |         |         |E->TE'   |         |
---------------------------------------------------------------------
T        |T->FT'   |         |         |T->FT'   |         |
---------------------------------------------------------------------
F        |F->(E)   |         |         |F->i     |         |
---------------------------------------------------------------------
E'       |         |E'->TE'  |E'->e    |         |         |
---------------------------------------------------------------------
T'       |         |T'->e    |T'->e    |         |T'->FT'  |
```

7. Implementation of Shift reduce Parser

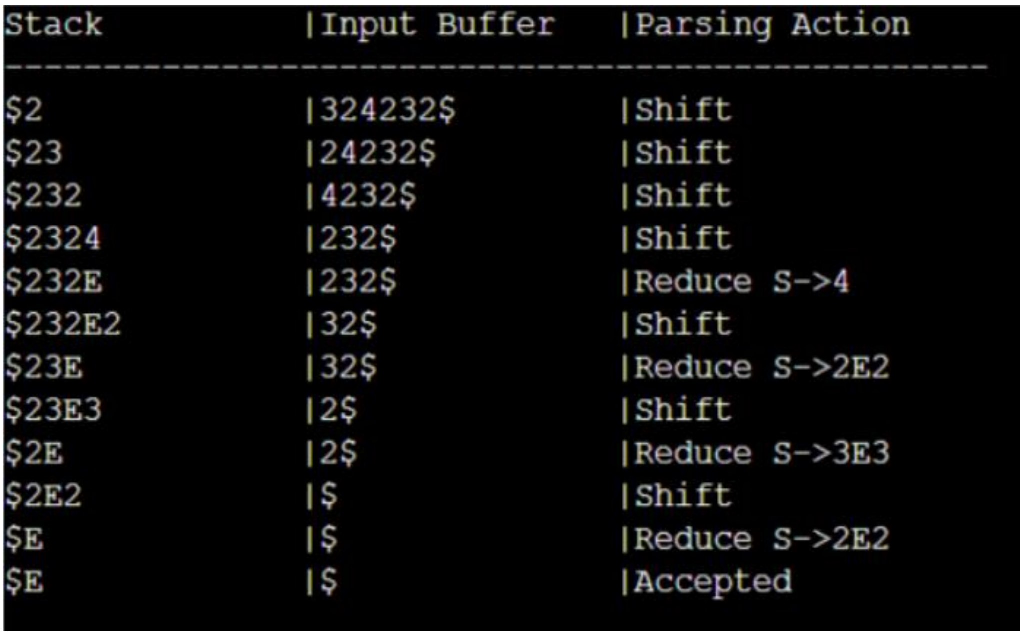
Aim: To write a code that can take grammar and produce shift reduce parser table

Source code:

```
gram = {
	"E":["2E2","3E3","4"]
}

starting_terminal = "E"
inp = "324230"
stack = "$"
print(f'{"Stack": <15}'+"|"+f'{"Input Buffer": <15}'+"|"+f'Parsing Action')
print(f'{"-":-<50}')

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

		i+=1
	if len(inp)>1:
		stack+=inp[0]
		inp=inp[1:]
		print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Shift')
		action = False
```

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

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

Output:



```
Stack          |Input Buffer   |Parsing Action
---------------------------------------------------
$2             |324232$        |Shift
$23            |24232$         |Shift
$232           |4232$          |Shift
$2324          |232$           |Shift
$232E          |232$           |Reduce S->4
$232E2         |32$            |Shift
$23E           |32$            |Reduce S->2E2
$23E3          |2$             |Shift
$2E            |2$             |Reduce S->3E3
$2E2           |$              |Shift
$E             |$              |Reduce S->2E2
$E             |$              |Accepted
```

8. LEADING AND TRAILING

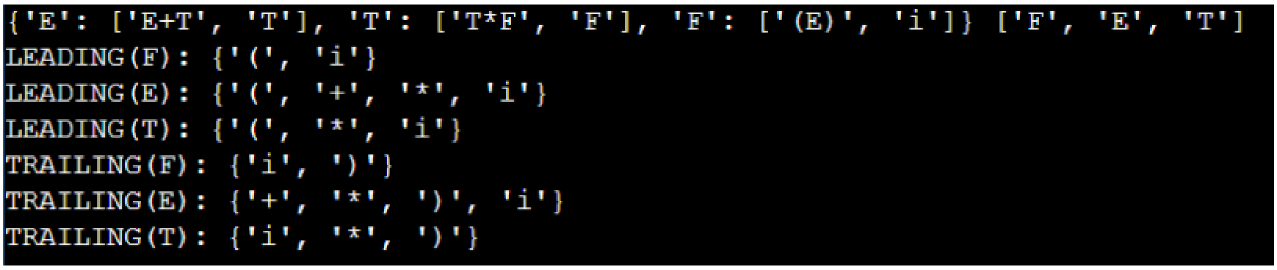
Aim: To write a program that implements leading and trailing

Source Code:

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

```
int vars,terms,i,j,k,m,rep,count,temp=-1;
char var[10],term[10],lead[10][10],trail[10][10];
struct grammar
{
        int prodno;
        char lhs,rhs[20][20];
}gram[50];
void get()
{
        cout<<"\nLEADING AND TRAILING\n";
        cout<<"\nEnter the no. of variables : ";
        cin>>vars;
        cout<<"\nEnter the variables : \n";
        for(i=0;i<vars;i++)
        {
                cin>>gram[i].lhs;
                var[i]=gram[i].lhs;
        }
        cout<<"\nEnter the no. of terminals : ";
        cin>>terms;
        cout<<"\nEnter the terminals : ";
        for(j=0;j<terms;j++)
                cin>>term[j];
        cout<<"\nPRODUCTION DETAILS\n";
        for(i=0;i<vars;i++)
        {
                cout<<"\nEnter the no. of production of "<<gram[i].lhs<<":";
                cin>>gram[i].prodno;
                for(j=0;j<gram[i].prodno;j++)
                {
                        cout<<gram[i].lhs<<"->";
                        cin>>gram[i].rhs[j];
                }
        }
}
void leading()
{
        for(i=0;i<vars;i++)
        {
                for(j=0;j<gram[i].prodno;j++)
                {
                        for(k=0;k<terms;k++)
                        {
                                if(gram[i].rhs[j][0]==term[k])
                                        lead[i][k]=1;
                                else
                                {
                                        if(gram[i].rhs[j][1]==term[k])
                                                lead[i][k]=1;
                                }
                        }
        }
    }
    for(rep=0;rep<vars;rep++)
    {
        for(i=0;i<vars;i++)
        {
            for(j=0;j<gram[i].prodno;j++)
            {
                for(m=1;m<vars;m++)
                {
                    if(gram[i].rhs[j][0]==var[m])
                    {
                        temp=m;
                        goto out;
                    }
                }
                out:

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

    for(rep=0;rep<vars;rep++)
    {
        for(i=0;i<vars;i++)
        {
            for(j=0;j<gram[i].prodno;j++)
            {
                count=0;
                while(gram[i].rhs[j][count]!='\x0')
                    count++;
                for(m=1;m<vars;m++)
                {
                    if(gram[i].rhs[j][count-1]==var[m])
                        temp=m;
                }
                for(k=0;k<terms;k++)
                {
                    if(trail[temp][k]==1)
                        trail[i][k]=1;
                }
            }
        }
    }
}
void display()
{
    for(i=0;i<vars;i++)
    {
        cout<<"\nLEADING("<<gram[i].lhs<<") = ";
        for(j=0;j<terms;j++)
        {
            if(lead[i][j]==1)
                cout<<term[j]<<",";
        }
    }
    cout<<endl;
    for(i=0;i<vars;i++)
    {
        cout<<"\nTRAILING("<<gram[i].lhs<<") = ";
        for(j=0;j<terms;j++)
        {
            if(trail[i][j]==1)
                cout<<term[j]<<",";
        }
    }
}
int main()
{
    get();
    leading();
    trailing();
    display();
}
```

Output:



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

9. LR (0)

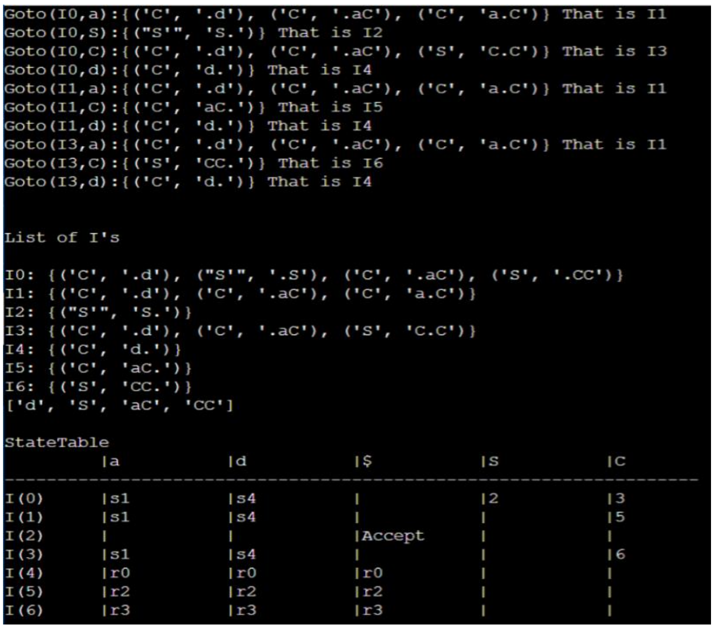
Aim: To write a program that can implement LR (0) parser and give out I traction.

Source Code:

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

using namespace std;

char prod[20][20],listofvar[26]="ABCDEFGHIJKLMNOPQR";
int novar=1,i=0,j=0,k=0,n=0,m=0,arr[30];
int noitem=0;

struct Grammar
{
        char lhs;
        char rhs[8];
}g[20],item[20],clos[20][10];
int isvariable(char variable)
{
        for(int i=0;i<novar;i++)
                if(g[i].lhs==variable)
                        return i+1;
        return 0;
}
void findclosure(int z, char a)
{
        int n=0,i=0,j=0,k=0,l=0;
        for(i=0;i<arr[z];i++)
        {
```

```
        for(j=0;j<strlen(clos[z][i].rhs);j++)
        {
            if(clos[z][i].rhs[j]=='.' && clos[z][i].rhs[j+1]==a)
            {
                clos[noitem][n].lhs=clos[z][i].lhs;
                strcpy(clos[noitem][n].rhs,clos[z][i].rhs);
                char temp=clos[noitem][n].rhs[j];
                clos[noitem][n].rhs[j]=clos[noitem][n].rhs[j+1];
                clos[noitem][n].rhs[j+1]=temp;
                n=n+1;
            }
        }
    }

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

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

 int main()
{
    cout<<"ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :\n";
    do
    {
        cin>>prod[i++];
    }while(strcmp(prod[i-1],"0")!=0);
    for(n=0;n<i-1;n++)
    {
        m=0;
        j=novar;
        g[novar++].lhs=prod[n][0];
        for(k=3;k<strlen(prod[n]);k++)
        {
            if(prod[n][k] != '|')
            g[j].rhs[m++]=prod[n][k];
            if(prod[n][k]=='|')
            {
                g[j].rhs[m]='\0';
                m=0;
                j=novar;
                g[novar++].lhs=prod[n][0];
            }
        }
    }
    for(i=0;i<26;i++)
        if(!isvariable(listofvar[i]))
            break;
    g[0].lhs=listofvar[i];
    char temp[2]={g[1].lhs,'\0'};
    strcat(g[0].rhs,temp);
    cout<<"\n\n augumented grammar \n";
    for(i=0;i<novar;i++)
        cout<<endl<<g[i].lhs<<"->"<<g[i].rhs<<" ";

    for(i=0;i<novar;i++)
    {
        clos[noitem][i].lhs=g[i].lhs;
        strcpy(clos[noitem][i].rhs,g[i].rhs);
        if(strcmp(clos[noitem][i].rhs,"ε")==0)
            strcpy(clos[noitem][i].rhs,".");
        else
        {
                for(int j=strlen(clos[noitem][i].rhs)+1;j>=0;j--)
                    clos[noitem][i].rhs[j]=clos[noitem][i].rhs[j-1];
                clos[noitem][i].rhs[0]='.';
            }
    }
    arr[noitem++]=novar;
    for(int z=0;z<noitem;z++)
    {
        char list[10];
        int l=0;
        for(j=0;j<arr[z];j++)
        {
            for(k=0;k<strlen(clos[z][j].rhs)-1;k++)
            {
                if(clos[z][j].rhs[k]=='.')
                {
                    for(m=0;m<l;m++)
                        if(list[m]==clos[z][j].rhs[k+1])
                            break;
                    if(m==l)
                        list[l++]=clos[z][j].rhs[k+1];
                }
            }
        }

        for(int x=0;x<l;x++)
            findclosure(z,list[x]);
    }
    cout<<"\n THE SET OF ITEMS ARE \n\n";
    for(int z=0; z<noitem; z++)
    {
        cout<<"\n I"<<z<<"\n\n";
        for(j=0;j<arr[z];j++)
            cout<<clos[z][j].lhs<<"->"<<clos[z][j].rhs<<"\n";

    }

}
```

Output:



```
Goto(I0,a):{('C', '.d'), ('C', '.aC'), ('C', 'a.C')} That is I1
Goto(I0,S):{("S'", 'S.')} That is I2
Goto(I0,C):{('C', '.d'), ('C', '.aC'), ('S', 'C.C')} That is I3
Goto(I0,d):{('C', 'd.')} That is I4
Goto(I1,a):{('C', '.d'), ('C', '.aC'), ('C', 'a.C')} That is I1
Goto(I1,C):{('C', 'aC.')} That is I5
Goto(I1,d):{('C', 'd.')} That is I4
Goto(I3,a):{('C', '.d'), ('C', '.aC'), ('C', 'a.C')} That is I1
Goto(I3,C):{('S', 'CC.')} That is I6
Goto(I3,d):{('C', 'd.')} That is I4


List of I's

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

StateTable
         |a          |d          |$          |S          |C
-------------------------------------------------------------------
I(0)     |s1         |s4         |           |2          |3
I(1)     |s1         |s4         |           |           |5
I(2)     |           |           |Accept     |           |
I(3)     |s1         |s4         |           |           |6
I(4)     |r0         |r0         |r0         |           |
I(5)     |r2         |r2         |r2         |           |
I(6)     |r3         |r3         |r3         |           |
```

10 . INTERMEDIATE CODE GENERATOR

INFIX TO POSTFIX AND PREFIX

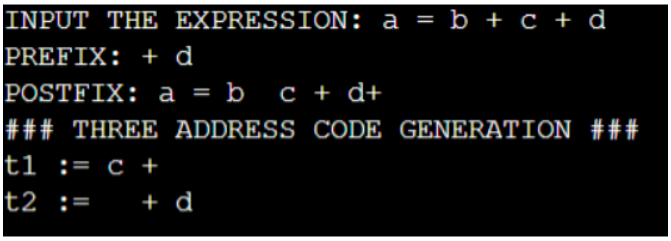
Aim: A program that implement intermediate code generation for Post and Prefix

Source Code:

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

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


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

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

    output = ''

    for ch in formula:

        if ch not in OPERATORS:

            output += ch

        elif ch == '(':

            stack.append('(')

        elif ch == ')':

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

            stack.pop()  # pop '('

        else:

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

            stack.append(ch)

            # leftover

    while stack:
        output += stack.pop()

    print(f'POSTFIX: {output}')

    return output


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

def infix_to_prefix(formula):
    op_stack = []
    exp_stack = []

    for ch in formula:

        if not ch in OPERATORS:

            exp_stack.append(ch)

        elif ch == '(':

            op_stack.append(ch)

        elif ch == ')':

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

                a  = exp_stack.pop()

                b = exp_stack.pop()

                exp_stack.append(op + b + a)

            op_stack.pop()  # pop '('

        else:

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

                a = exp_stack.pop()

                b = exp_stack.pop()

                exp_stack.append(op + b + a)

            op_stack.append(ch)

            # leftover

    while op_stack:
        op = op_stack.pop()

        a  = exp_stack.pop()

        b = exp_stack.pop()

        exp_stack.append(op + b + a)

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

    return exp_stack[-1]

expres = input("INPUT THE EXPRESSION: ")

pre = infix_to_prefix(expres)
```

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

Output:



```
INPUT THE EXPRESSION: a = b + c + d
PREFIX: + d
POSTFIX: a = b  c + d+
### THREE ADDRESS CODE GENERATION ###
t1 := c +
t2 :=   + d
```

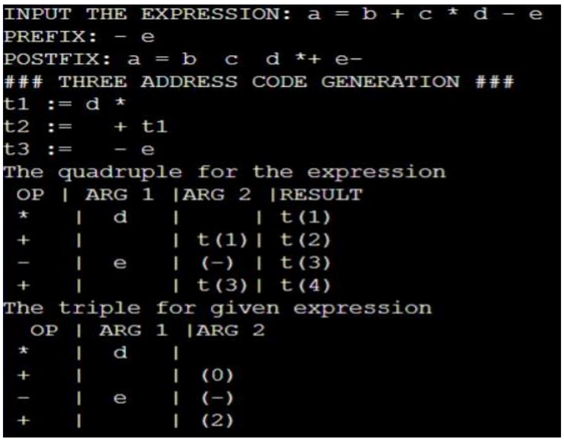
11. REPRESENTATION OF INTERMEDIATE CODE QUADRUPLES, TRIPLES, THREE ADDRESS CODE

Aim: Write a code to represent intermediate code for Quadruples, triples, three address code

Source code:

```
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
#include<string.h>
void small();
void dove(int i);
int p[5]={0,1,2,3,4},c=1,i,k,l,m,pi;
char sw[5]={'=','-','+','/','*'},j[20],a[5],b[5],ch[2];
void main()
{
 printf("Enter the expression:");
 scanf("%s",j);
 printf("\tThe Intermediate code is:\n");
 small();
}
void dove(int i)
{
 a[0]=b[0]='\0';
 if(!isdigit(j[i+2])&&!isdigit(j[i-2]))
 {
   a[0]=j[i-1];
   b[0]=j[i+1];
 }
 if(isdigit(j[i+2])){
   a[0]=j[i-1];
   b[0]='t';
   b[1]=j[i+2];
 }
 if(isdigit(j[i-2]))
 {
   b[0]=j[i+1];
   a[0]='t';
   a[1]=j[i-2];
   b[1]='\0';
 }
 if(isdigit(j[i+2]) &&isdigit(j[i-2]))
 {
   a[0]='t';
   b[0]='t';
   a[1]=j[i-2];
   b[1]=j[i+2];
   sprintf(ch,"%d",c);
   j[i+2]=j[i-2]=ch[0];
  }
  if(j[i]=='*')
  printf("\tt%d=%s*%s\n",c,a,b);
  if(j[i]=='/')
  printf("\tt%d=%s/%s\n",c,a,b);
  if(j[i]=='+')
   printf("\tt%d=%s+%s\n",c,a,b);if(j[i]=='-')
   printf("\tt%d=%s-%s\n",c,a,b);
  if(j[i]=='=')
  printf("\t%c=t%d",j[i-1],--c);
  sprintf(ch,"%d",c);
  j[i]=ch[0];
  c++;
  small();
}
void small()
{
  pi=0;l=0;
  for(i=0;i<strlen(j);i++)
  {
   for(m=0;m<5;m++)
     if(j[i]==sw[m])
     {
      if(pi<=p[m])
      {
        pi=p[m];
        l=1;
      }
     }
  }
  if(l==1)
  dove(k);
  else
   exit(0);
}
```

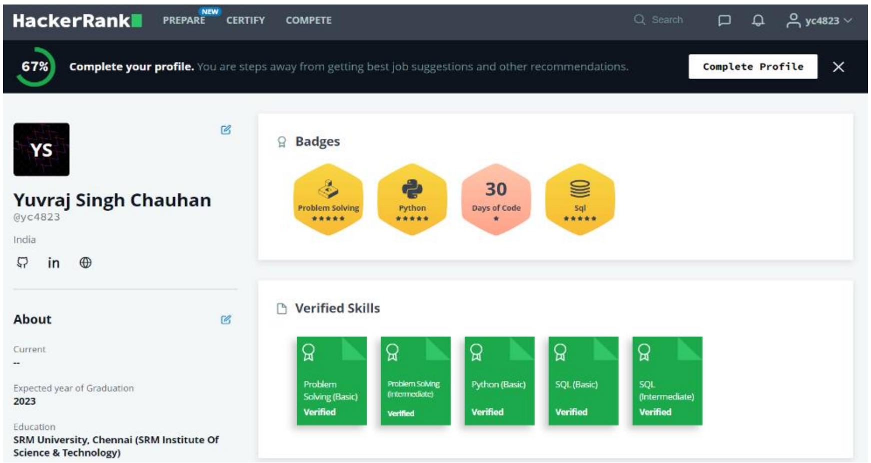
Output:



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

Hacker rank Achievements

Profile



67% **Complete your profile.** You are steps away from getting best job suggestions and other recommendations. Complete Profile

YS

Yuvraj Singh Chauhan
@yc4823
India

About

Current
--

Expected year of Graduation
2023

Education
SRM University, Chennai (SRM Institute Of Science & Technology)

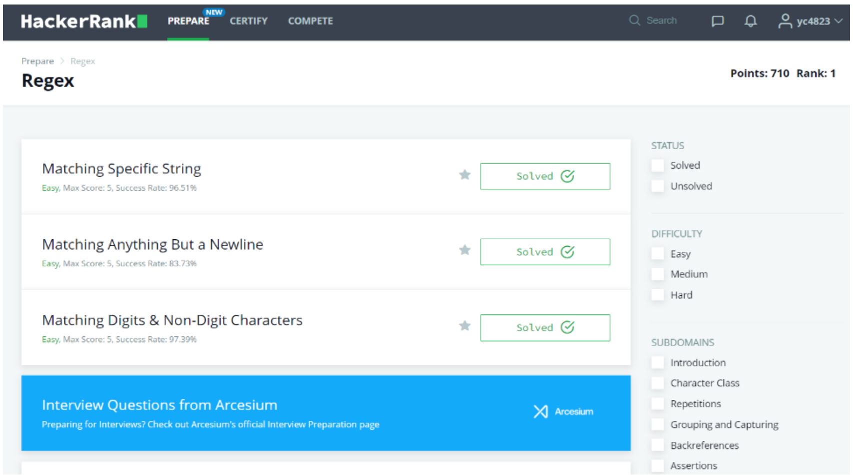
Badges

Problem Solving ★★★★★
Python ★★★★★
30 Days of Code ★
Sql ★★★★★

Verified Skills

Problem Solving (Basic) Verified
Problem Solving (Intermediate) Verified
Python (Basic) Verified
SQL (Basic) Verified
SQL (Intermediate) Verified

Regex Page



Prepare > Regex

Regex

Points: 710 Rank: 1

Matching Specific String ★ Solved ✓
Easy, Max Score: 5, Success Rate: 96.51%

Matching Anything But a Newline ★ Solved ✓
Easy, Max Score: 5, Success Rate: 83.73%

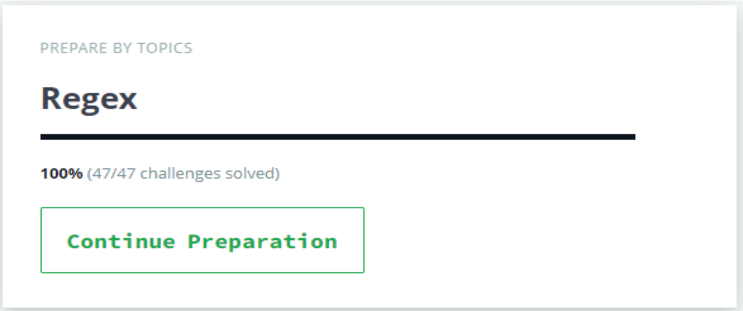
Matching Digits & Non-Digit Characters ★ Solved ✓
Easy, Max Score: 5, Success Rate: 97.39%

Interview Questions from Arcesium
Preparing for Interviews? Check out Arcesium's official Interview Preparation page
Arcesium

STATUS
□ Solved
□ Unsolved

DIFFICULTY
□ Easy
□ Medium
□ Hard

SUBDOMAINS
□ Introduction
□ Character Class
□ Repetitions
□ Grouping and Capturing
□ Backreferences
□ Assertions

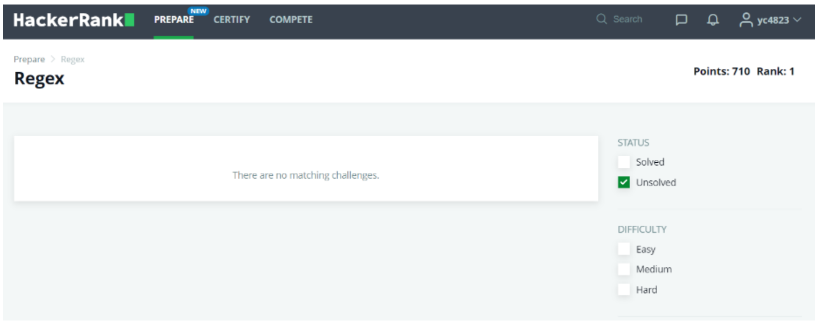


PREPARE BY TOPICS

Regex

100% (47/47 challenges solved)

Continue Preparation



Prepare > Regex

Regex

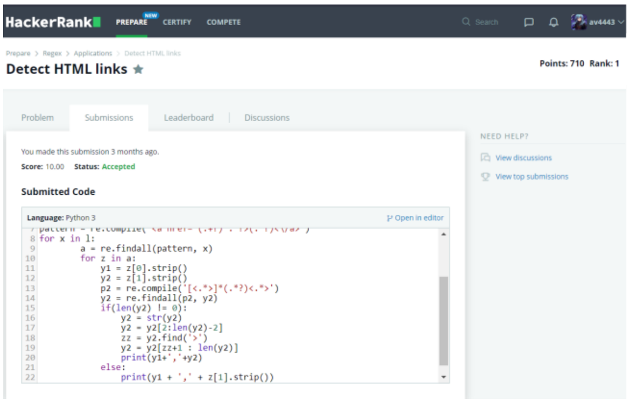
Points: 710 Rank: 1

There are no matching challenges.

STATUS
□ Solved
☑ Unsolved

DIFFICULTY
□ Easy
□ Medium
□ Hard

<u>Code Snippets:</u>



Prepare > Regex > Applications > Detect HTML links

Detect HTML links ★

Points: 710 Rank: 1

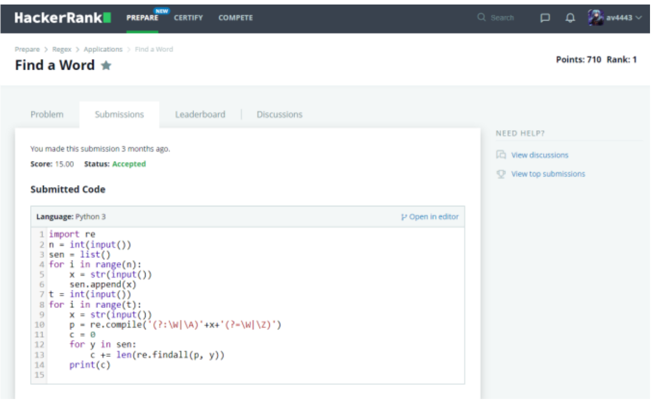
You made this submission 3 months ago.

Score: 10.00 **Status:** Accepted

Submitted Code

Language: Python 3

```python
pattern = re.compile('<a href="(.+?)".*?>(.*?)</a>')
for x in l:
        a = re.findall(pattern, x)
        for z in a:
            y1 = z[0].strip()
            y2 = z[1].strip()
            p2 = re.compile('[<.*>]*(.*?)<.*>')
            y2 = re.findall(p2, y2)
            if(len(y2) != 0):
                y2 = str(y2)
                y2 = y2[2:len(y2)-2]
                zz = y2.find('>')
                y2 = y2[zz+1 : len(y2)]
                print(y1+','+y2)
            else:
                print(y1 + ',' + z[1].strip())
```



Prepare > Regex > Applications > Find a Word

Find a Word ★

Points: 710 Rank: 1

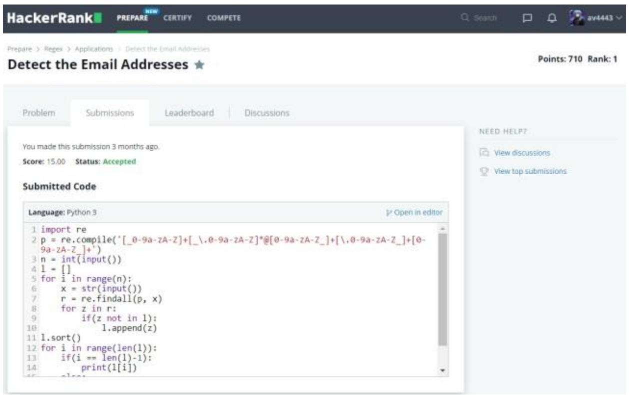
You made this submission 3 months ago.

Score: 15.00 **Status:** Accepted

Submitted Code

Language: Python 3

```python
import re
n = int(input())
sen = list()
for i in range(n):
    x = str(input())
    sen.append(x)
t = int(input())
for i in range(t):
    x = str(input())
    p = re.compile('(?:\W|\A)'+x+'(?=\W|\Z)')
    c = 0
    for y in sen:
        c += len(re.findall(p, y))
    print(c)

```



Prepare > Regex > Applications > Detect the Email Addresses

Detect the Email Addresses ★

Points: 710 Rank: 1

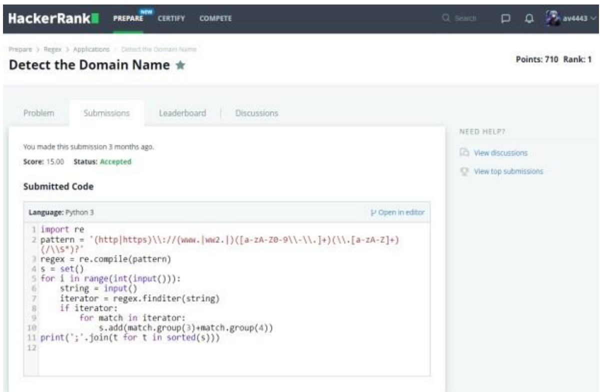
You made this submission 3 months ago.

Score: 15.00 **Status:** Accepted

Submitted Code

Language: Python 3

```python
import re
p = re.compile('[_0-9a-zA-Z]+[_\.0-9a-zA-Z]*@[0-9a-zA-Z_]+[\.0-9a-zA-Z_]+[0-
9a-zA-Z_]+')
n = int(input())
l = []
for i in range(n):
    x = str(input())
    r = re.findall(p, x)
    for z in r:
        if(z not in l):
            l.append(z)
l.sort()
for i in range(len(l)):
    if(i == len(l)-1):
        print(l[i])
    else:
```



Prepare > Regex > Applications > Detect the Domain Name

Detect the Domain Name ★

Points: 710 Rank: 1

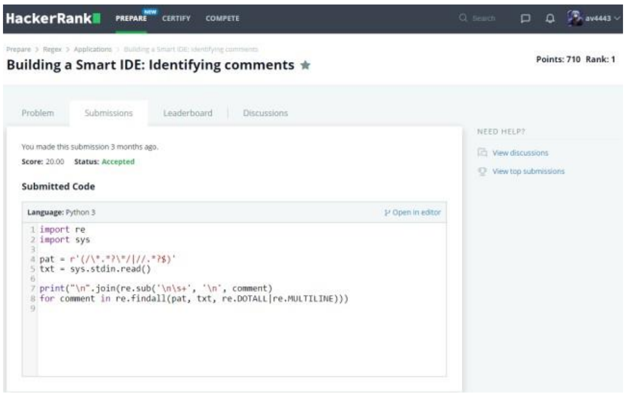
You made this submission 3 months ago.

Score: 15.00 **Status:** Accepted

Submitted Code

Language: Python 3

```python
import re
pattern = '(http|https)\\://(www.|ww2.|)([a-zA-Z0-9\\-\\.]+)(\\.[a-zA-Z]+)
(/\\S*)?'
regex = re.compile(pattern)
s = set()
for i in range(int(input())):
    string = input()
    iterator = regex.finditer(string)
    if iterator:
        for match in iterator:
            s.add(match.group(3)+match.group(4))
print(';'.join(t for t in sorted(s)))

```



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

Building a Smart IDE: Identifying comments ★

Points: 710 Rank: 1

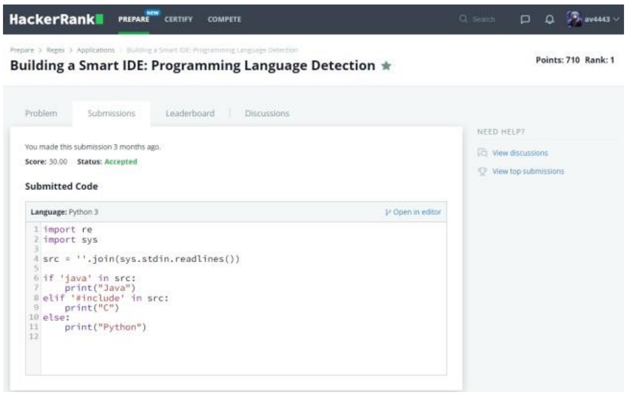
You made this submission 3 months ago.

Score: 20.00 **Status:** Accepted

Submitted Code

Language: Python 3

```python
import re
import sys

pat = r'(/\*.*?\*/|//.*?$)'
txt = sys.stdin.read()

print("\n".join(re.sub('\n\s+', '\n', comment)
for comment in re.findall(pat, txt, re.DOTALL|re.MULTILINE)))

```



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

Building a Smart IDE: Programming Language Detection ★

Points: 710 Rank: 1

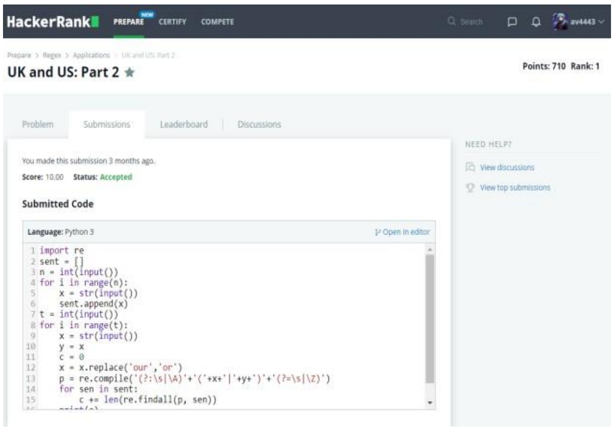
You made this submission 3 months ago.

Score: 30.00 **Status:** Accepted

Submitted Code

Language: Python 3

```python
import re
import sys

src = ''.join(sys.stdin.readlines())

if 'java' in src:
    print("Java")
elif '#include' in src:
    print("C")
else:
    print("Python")

```



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

UK and US: Part 2 ★

Points: 710 Rank: 1

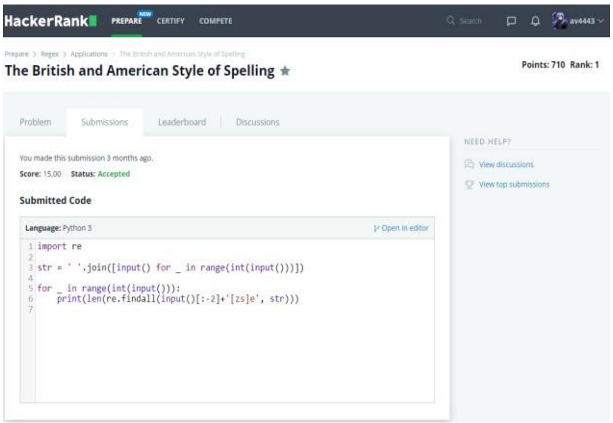
You made this submission 3 months ago.

Score: 10.00 **Status:** Accepted

Submitted Code

Language: Python 3

```python
import re
sent = []
n = int(input())
for i in range(n):
    x = str(input())
    sent.append(x)
t = int(input())
for i in range(t):
    x = str(input())
    y = x
    c = 0
    x = x.replace('our','or')
    p = re.compile('(?:\s|\A)'+'('+x+'|'+y+')'+'(?=\s|\Z)')
    for sen in sent:
        c += len(re.findall(p, sen))
    print(c)
```



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

The British and American Style of Spelling ★

Points: 710 Rank: 1

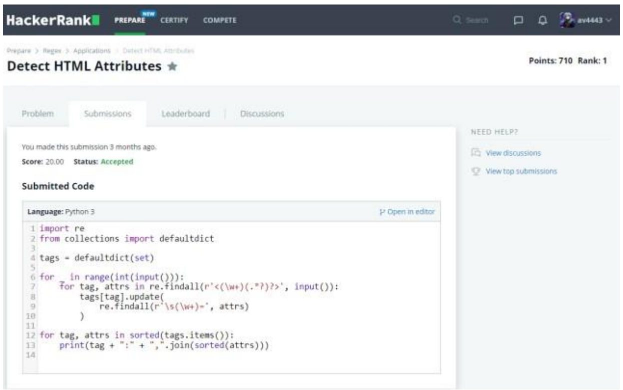
You made this submission 3 months ago.

Score: 15.00 **Status:** Accepted

Submitted Code

Language: Python 3

```python
import re

str = ' '.join([input() for _ in range(int(input()))])

for _ in range(int(input())):
    print(len(re.findall(input()[:-2]+'[zs]e', str)))

```



Detect HTML Attributes ★

Points: 710 Rank: 1

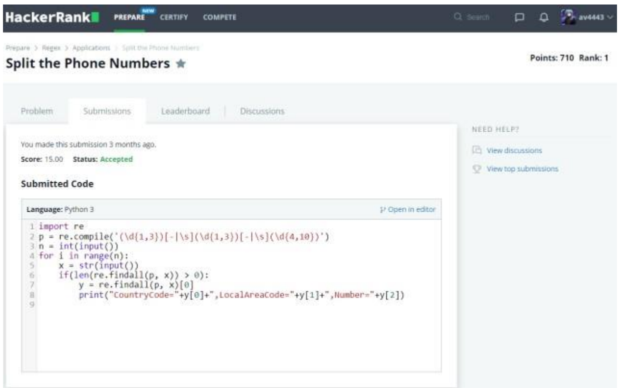
You made this submission 3 months ago.

Score: 20.00 Status: Accepted

Submitted Code

Language: Python 3

```
import re
from collections import defaultdict

tags = defaultdict(set)

for _ in range(int(input())):
    for tag, attrs in re.findall(r'<(\w+)(.*?)?>', input()):
        tags[tag].update(
            re.findall(r'\s(\w+)=', attrs)
        )

for tag, attrs in sorted(tags.items()):
    print(tag + ":" + ",".join(sorted(attrs)))

```



Split the Phone Numbers ★

Points: 710 Rank: 1

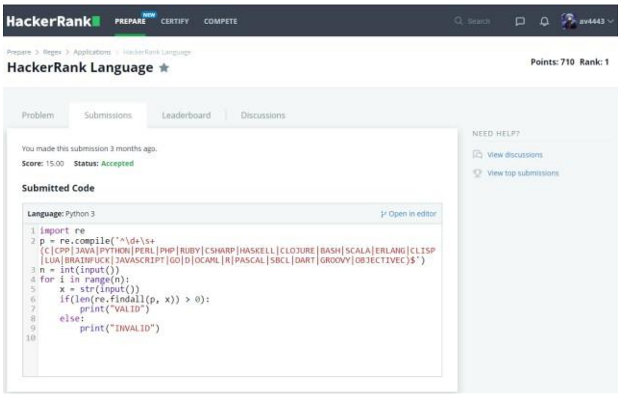
You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3

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

```



HackerRank Language ★

Points: 710 Rank: 1

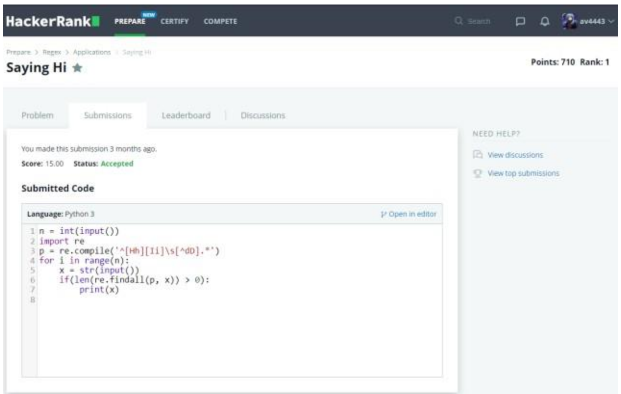
You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3

```
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")

```



Saying Hi ★

Points: 710 Rank: 1

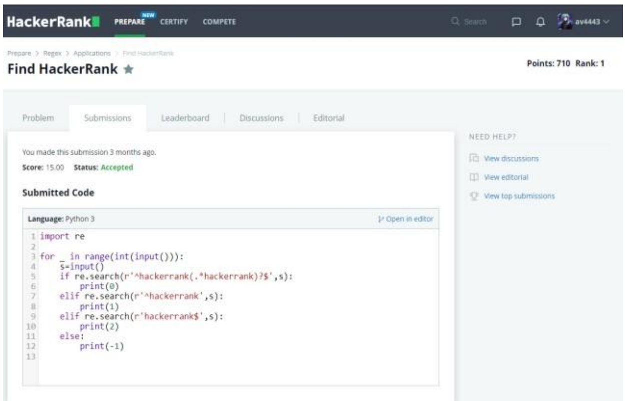
You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3

```
n = int(input())
import re
p = re.compile('^[Hh][Ii]\s[^dD].*')
for i in range(n):
    x = str(input())
    if(len(re.findall(p, x)) > 0):
        print(x)

```



Find HackerRank ★

Points: 710 Rank: 1

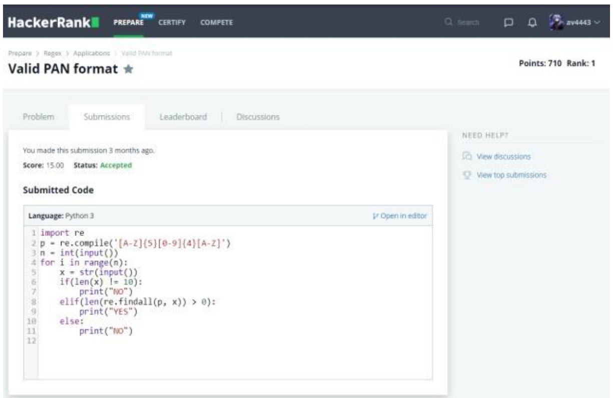
You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3

```
import re

for _ in range(int(input())):
    s=input()
    if re.search(r'^hackerrank(.*hackerrank)?$',s):
        print(0)
    elif re.search(r'^hackerrank',s):
        print(1)
    elif re.search(r'hackerrank$',s):
        print(2)
    else:
        print(-1)

```



Valid PAN format ★

Points: 710 Rank: 1

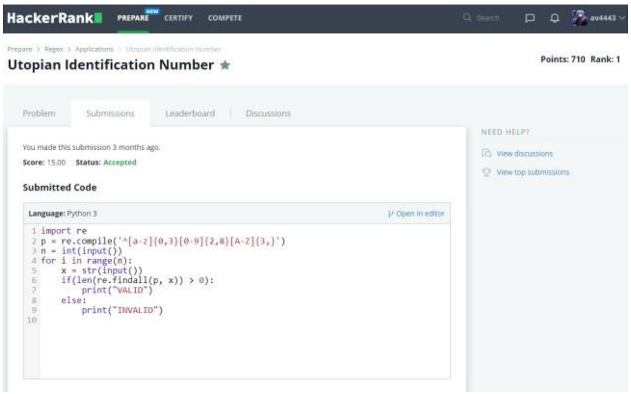
You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3

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

```



Utopian Identification Number ★

Points: 710 Rank: 1

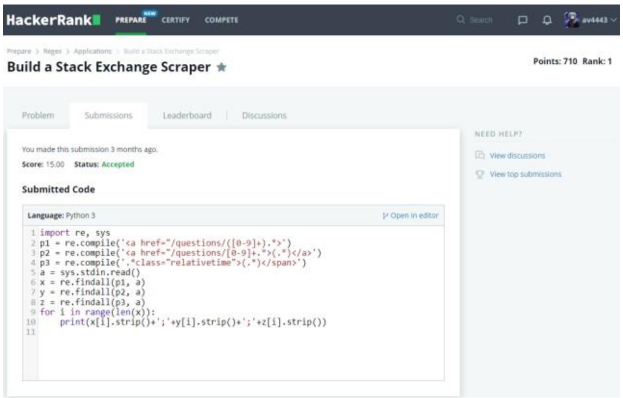
You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3

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

```



Build a Stack Exchange Scraper ★

Points: 710 Rank: 1

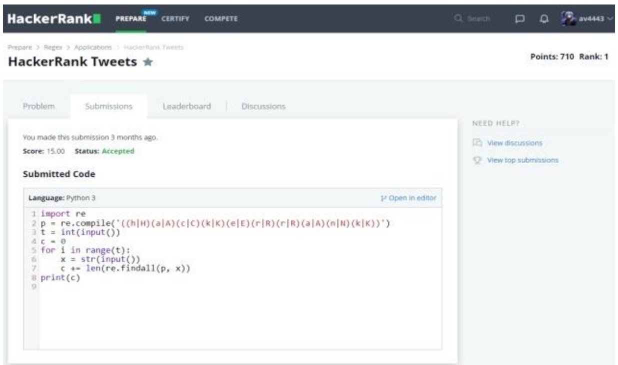
You made this submission 3 months ago.

Score: 15.00 Status: Accepted

Submitted Code

Language: Python 3

```
import re, sys
p1 = re.compile('<a href="/questions/([0-9]+).*">')
p2 = re.compile('<a href="/questions/[0-9]+.*">(.*)</a>')
p3 = re.compile('.*class="relativetime">(.*)</span>')
a = sys.stdin.read()
x = re.findall(p1, a)
y = re.findall(p2, a)
z = re.findall(p3, a)
for i in range(len(x)):
    print(x[i].strip()+';'+y[i].strip()+';'+z[i].strip())

```



HackerRank Tweets ★

Points: 710 Rank: 1

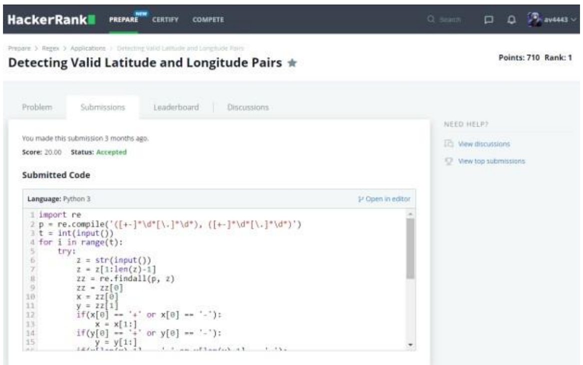
You made this submission 3 months ago.

Score: 15.00 **Status:** Accepted

Submitted Code

Language: Python 3

```
import re
p = re.compile('((h|H)(a|A)(c|C)(k|K)(e|E)(r|R)(r|R)(a|A)(n|N)(k|K))')
t = int(input())
c = 0
for i in range(t):
    x = str(input())
    c += len(re.findall(p, x))
print(c)
```



Detecting Valid Latitude and Longitude Pairs ★

Points: 710 Rank: 1

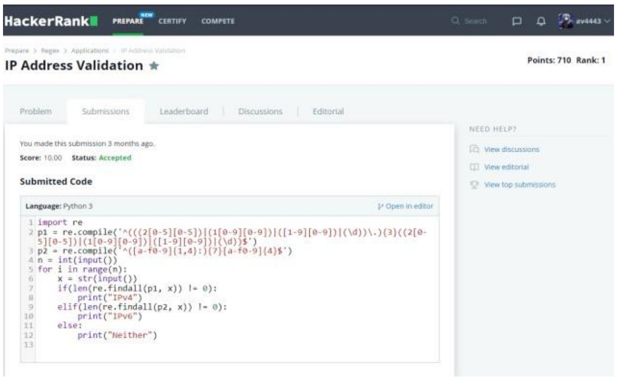
You made this submission 3 months ago.

Score: 20.00 **Status:** Accepted

Submitted Code

Language: Python 3

```
import re
p = re.compile('([+-]*\d*[\.]*\d*), ([+-]*\d*[\.]*\d*)')
t = int(input())
for i in range(t):
    try:
        z = str(input())
        z = z[1:len(z)-1]
        zz = re.findall(p, z)
        zz = zz[0]
        x = zz[0]
        y = zz[1]
        if(x[0] == '+' or x[0] == '-'):
            x = x[1:]
        if(y[0] == '+' or y[0] == '-'):
            y = y[1:]
```



IP Address Validation ★

Points: 710 Rank: 1

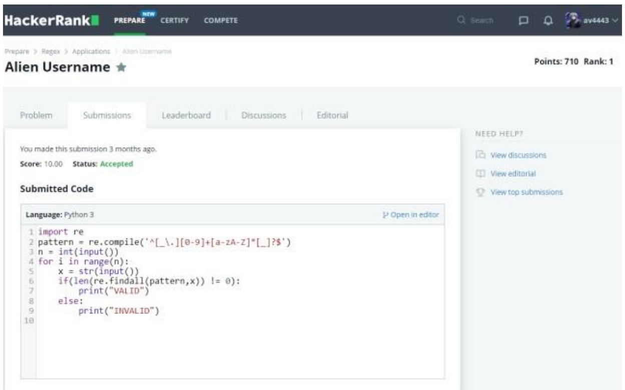
You made this submission 3 months ago.

Score: 10.00 **Status:** Accepted

Submitted Code

Language: Python 3

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



Alien Username ★

Points: 710 Rank: 1

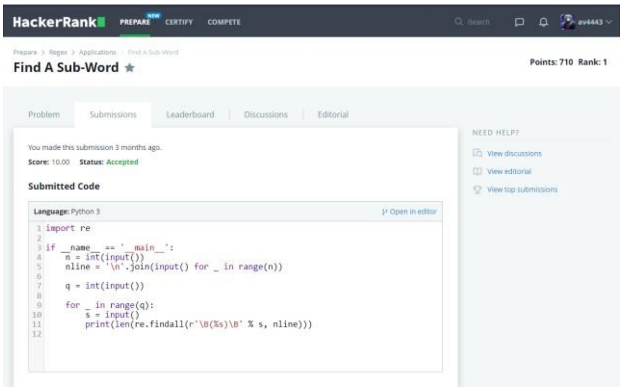
You made this submission 3 months ago.

Score: 10.00 **Status:** Accepted

Submitted Code

Language: Python 3

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



Find A Sub-Word ★

Points: 710 Rank: 1

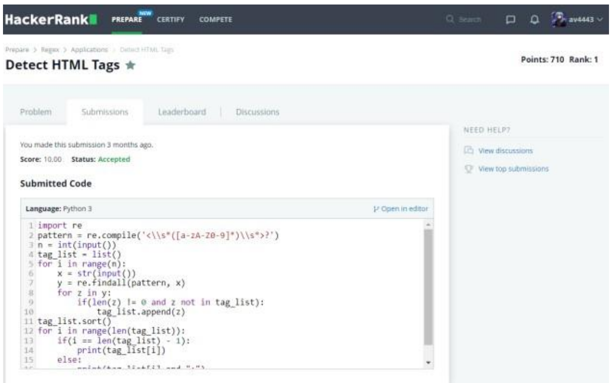
You made this submission 3 months ago.

Score: 10.00 **Status:** Accepted

Submitted Code

Language: Python 3

```
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)))
```



Detect HTML Tags ★

Points: 710 Rank: 1

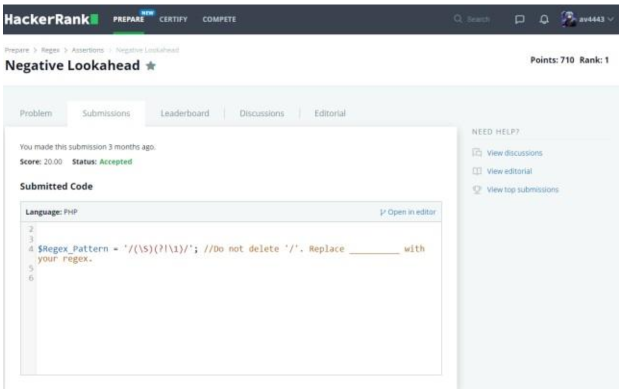
You made this submission 3 months ago.

Score: 10.00 **Status:** Accepted

Submitted Code

Language: Python 3

```
import re
pattern = re.compile('<\\s*([a-zA-Z0-9]*)\\s*>?')
n = int(input())
tag_list = list()
for i in range(n):
    x = str(input())
    y = re.findall(pattern, x)
    for z in y:
        if(len(z) != 0 and z not in tag_list):
            tag_list.append(z)
tag_list.sort()
for i in range(len(tag_list)):
    if(i == len(tag_list) - 1):
        print(tag_list[i])
    else:
```



Negative Lookahead ★

Points: 710 Rank: 1

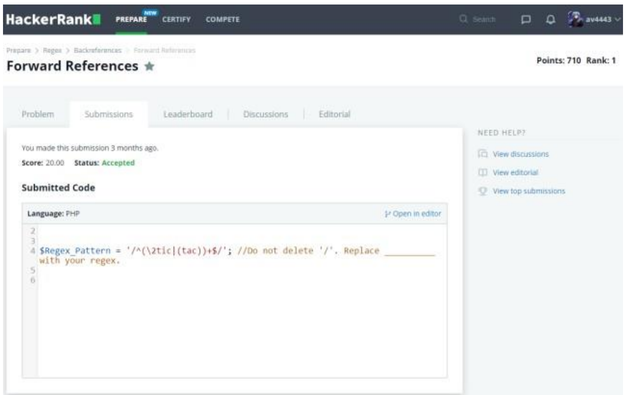
You made this submission 3 months ago.

Score: 20.00 **Status:** Accepted

Submitted Code

Language: PHP

```


$Regex_Pattern = '/(\S)(?!\1)/'; //Do not delete '/'. Replace __________ with
your regex.


```



Forward References ★

Points: 710 Rank: 1

You made this submission 3 months ago.

Score: 20.00 **Status:** Accepted

Submitted Code

Language: PHP

```


$Regex_Pattern = '/^(\2tic|(tac))+$/'; //Do not delete '/'. Replace __________
with your regex.


```

As a part of Student online assiGnment every student was asked to complete 20 ReGex problems from any codinG platform available and I tookhack rank and completed the module as shown in above photos.

<u>Mini Project</u>

SRM INSTITUTE OF SCIENCE AND

TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF DATASCIENCE AND

BUSINESS SYSTEMS

18CSC304J COMPILER DESIGN

MINI PROJECT REPORT

<u>Title</u> – Random Password Generator

Name: Yuvraj Singh Chauhan, Arnav Kumar

Register Number: RA1911027010058,

RA1911027010040

Department: B.Tech

Specialization: CSE BIG DATA Analytics

Semester: VI

CONTENT PAGE

INTRODUCTION

Text based username-password is the most commonly employed authentication mechanism in many multiuser environments. These multiuser applications, while registering users to their application, some applications allow users to create password their own and others generate random password and supply to users. Various surveys have shown users created passwords are less secure than system generated passwords. Most user created passwords can be found in common password lists on internet. The user created passwords can be guessed easily, with a bit of social engineering like user'spersonal information or type of application. System generated passwords cannot be guessed easily and have no relevance with the user’s personal information and typeof application but are hard to remember.

Text based password authentication systems involve a tradeoff between security and memorability of passwords. Some passwords are easy to remember but also easy to guess for an adversary. Random passwords are hard to remember and hard to crack because they are made up ofarbitrary sequence of characters. Several studies have examined how password composition policies affect users.In some studies, it is revealed that password composition policies influence the predictability of passwords and how well they affect the user behaviour and sentiments. Their results demonstrate that successfully creating a password is significantly more difficult under stricter password composition policies. They measured how many people failed at least once to create an acceptable password and further observed how users deal with it.

SYNOPSIS

Passwords are a real security threat. Over 80% of hacking-related breaches are due to weak or stolen passwords, a recent report shows . So if you want to safeguard your personal info and assets, creating secure passwords is a big first step. And that’s where Random Password Generator can help. Impossible-to-crack passwords are complex with multiple types of characters (numbers, letters, and symbols). Making your passwords different foreach website or app also helps defend against hacking.

A random password generator is software program or hardware device that takes input from a random or pseudo-random number generator and automatically generates a password. Random passwords can be generated manually, using simple sources of randomnesssuch as dice or coins, or they can be generated using a computer.

This random password generator is built around lexical analysis using python where we are converting charactersinto tokens and concatenating them to make a random password for the user

LANGUAGE AND MODULES

- Python
- String Module
- Random Module
- Tkinter Module

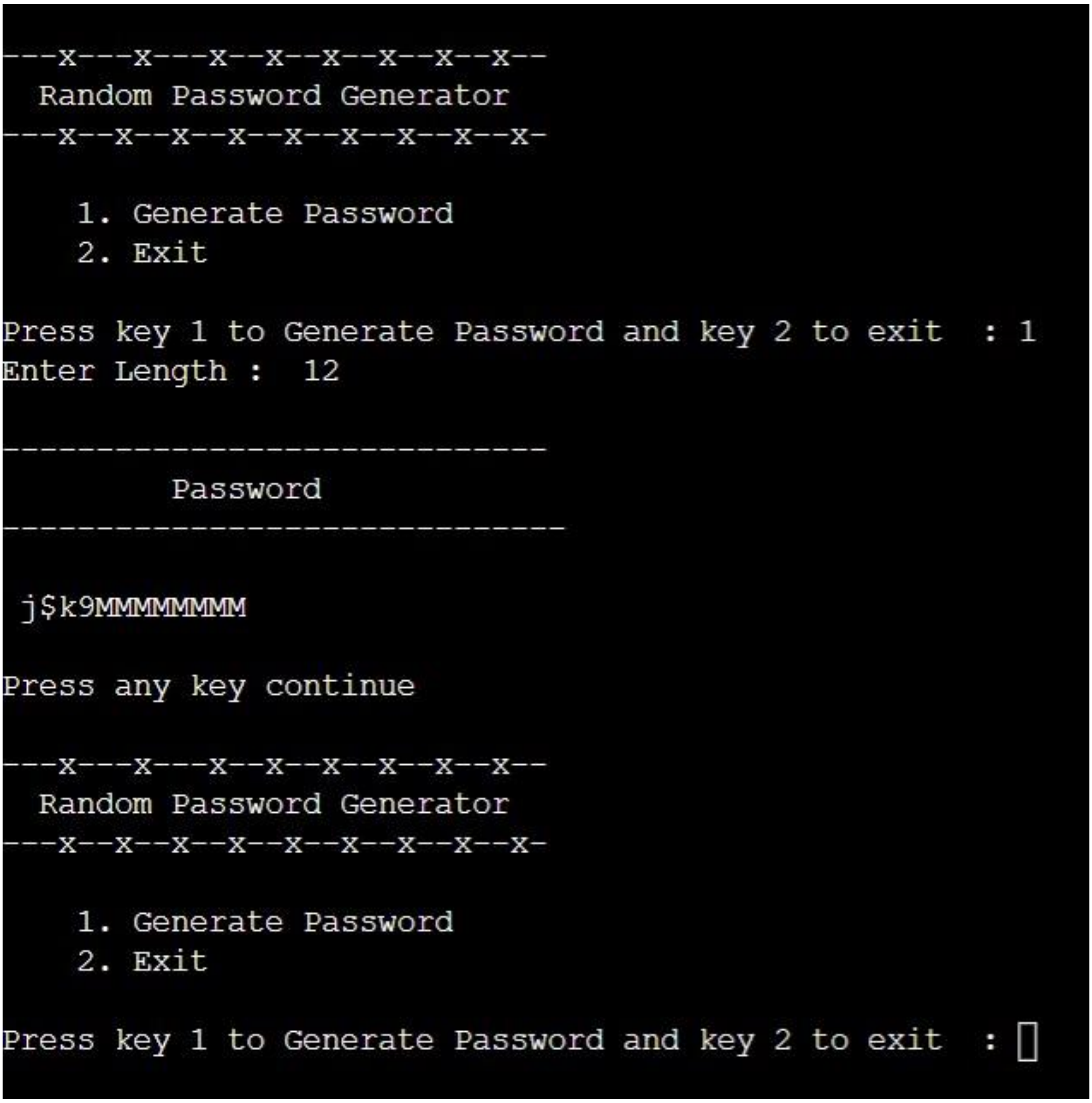
SOURCE CODE

```
#include <bits/stdc++.h>
using namespace std;

int selectArray()
```

```
{
    srand(time(NULL));
    int i = rand() % 5;
    if (i == 0)
        i++;
    return i;
}


int getKey()
{
    srand(time(NULL));

    int key = rand() % 26;
    return key;
}
void generate_password(int length)
{

    string password = "";
    string alphabet = "abcdefghijklmnopqrstuvwxyz";
    string ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    string   s_symbol   =   "!@#$%&";
    string number = "0123456789";
    int  key,  count_alphabet  =  0,  count_ALPHABET  =  0,  count_number  =  0,
count_s_symbol = 0;
    int count = 0;
    while  (count  <  length)  {
        int k = selectArray();if
        (count == 0) {
            k = k % 3;
            if (k == 0)
                k++;
        }
        switch (k) {
        case 1:

            if ((count_alphabet == 2) && (count_number == 0 || count_ALPHABET == 0
|| count_ALPHABET == 1 || count_s_symbol == 0))break;

            key = getKey();
            password = password + alphabet[key];
            count_alphabet++;
            count++;
            break;
        case 2:

            if ((count_ALPHABET == 2) && (count_number == 0 || count_alphabet == 0
|| count_alphabet == 1 || count_s_symbol == 0))
                break;
            key = getKey();
            password = password + ALPHABET[key];
            count_ALPHABET++;
            count++;
            break;

        case 3:

            if ((count_number == 1) && (count_alphabet == 0 || count_alphabet == 1 ||
count_ALPHABET == 1 || count_ALPHABET == 0 || count_s_symbol == 0))
                break;

            key = getKey();
            key = key % 10;
            password = password + number[key];
            count_number++;
            count++;
            break;

        case 4:

            if ((count_s_symbol == 1) && (count_alphabet == 0 || count_alphabet == 1 ||
count_ALPHABET == 0 || count_ALPHABET == 1 || count_number == 0))
                break;

            key = getKey();
            key = key % 6;
            password = password + s_symbol[key];
            count_s_symbol++;
            count++;
            break;
        }
    }

    cout << "\n-______________________\n";
    cout << "          Password              \n";
    cout << "________________________\n\n";
    cout << " " << password;
    cout << "\n\nPress any key continue \n";
    getchar();
```

```
}
int main()
{
    int opt, length;
    do {
        cout << "\n---x---x---x--x--x--x--x--x--\n"; cout
        << " Random Password Generator\n";cout <<
        "---x--x--x--x--x--x--x--x--x-\n\n"; cout << " 1.
        Generate Password"
             << "\n";
        cout << "      2. Exit"
             << "\n\n";
        cout << "Press key 1 to Generate Password and key 2 to exit  : ";cin >>
        opt;

        switch (opt) {
        case 1:
            cout << "Enter Length : ";cin
            >> length;

            if (length < 7) {
                cout << "\nError : Password Length Should be atleast 7\n";cout
                << "Press any key to try again \n";
                getchar();
            }

            else if (length > 100) {
                cout << "\nError : Maximum length of password should be 100\n";cout
                << "Press any key to try again \n";
                getchar();
            }

            else
                generate_password(length);
            break;

        default:

            if (opt != 2) {
                printf("\nInvalid choice\n");
                printf("Please Press ( 1 ) to generate password and ( 2 ) to exit.\n");cout
                << "Press any key to try again \n";
                getchar();
            }
            break;
        }
    } while (opt != 2);

    return 0;
}
```

OUTPUT



```
---x---x---x--x--x--x--x--x--
  Random Password Generator
---x--x--x--x--x--x--x--x--x-

    1. Generate Password
    2. Exit

Press key 1 to Generate Password and key 2 to exit  : 1
Enter Length :  12

-----------------------------
         Password
------------------------------

 j$k9MMMMMMMM

Press any key continue

---x---x---x--x--x--x--x--x--
  Random Password Generator
---x--x--x--x--x--x--x--x--x-

    1. Generate Password
    2. Exit

Press key 1 to Generate Password and key 2 to exit  : 
```

CONCLUSION

Complex password composition policies and policies that require password must be changed after a period of time happens to be major obstacle for users. The proposed technique can assist system administrators in creating secure and memorable passwords for users with desired complex password composition policies. The generated password along with helping information (random word, random position string and random character string) will besent to users.

This technique gives several benefits to users such assecurity, and confidentiality. The password generated using proposed technique is more secure because it ischosen from a large distribution of passwords and is stronger than user created passwords. The proposed technique causes more Confidentiality because in thistechnique, distinct passwords are given to users on different applications.

If an application is compromised then rest of all are protected. Future work includes determining the memorability of the generated password. Intuitively, it canbe said that the passwords generated using proposed technique are more memorable than pure random passwords.

APPENDIX B - GITHUB PROFILE AND LINK FOR THE PROJECT

GitHub Profile - https://github.com/yuvrajsinghchauhan
Project link - https://github.com/yuvrajsinghchauhan/Random-Password-Generator

SIGNATURE

NOTE: ENCLOSE THE ASSIGNMENT AND RELEVANT CERTIFICATES ALONG WITH THE PROFILE