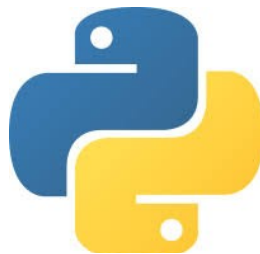




ESPOL

LENGUAJES DE PROGRAMACION

PROYECTO:
HERRAMIENTA EDUCATIVA PYTHON



INTEGRANTES:
DELGADO CAMPOVERDE PABLO
CELLY ASANZA JIMMY
SANCHEZ NARANJO EILEEN

Introducción

Un compilador es un programa que transforma las líneas de código que normalmente son instrucciones de alto nivel, en instrucciones legibles para la computadora. Para que un compilador pueda transformar dicho código a código máquina, se debe seguir una serie de pasos los cuales son: análisis léxico, análisis sintáctico, generación de código intermedio, optimización de código y generación del código ensamblador. El presente trabajo tiene como objetivo de estudio, implementar los dos primeros pasos antes mencionados.

Para el análisis léxico de las sentencias escritas en un código fuente, se utilizará la herramienta **lex**. **Lex** es un programa que genera analizadores léxicos y ha sido muy usada a lo largo de los años con el auge de los sistemas Unix. A su vez, se utilizará el programa **yacc**, que por sus siglas en inglés significa Yet Another Compiler-Compiler, es usado junto con lex ya que es el encargado de generar el análisis sintáctico de las expresiones.

Esto quiere decir que **yacc** comprueba que la estructura organizada en el código fuente concuerde con las especificaciones sintácticas del lenguaje mediante **BNF**. Para el desarrollo de este proyecto se usará el lenguaje Python ya que es multiparadigma, lo cual permite varios estilos al momento de programar como la programación estructural, funcional, orientada a objetos, etc.

Python posee de una librería muy útil la cual permite hacer análisis léxico y sintáctico, el nombre de esta librería es **PLY** (Python Lex-Yacc) el cual implementa las herramientas de parseo Lex y Yacc.

Objetivo

Construir una interfaz gráfica que pueda visualizar el análisis y validación de la parte léxica (Lexer) de los tokens usados y visualizar la construcción de la parte sintáctica (Parser) con la librería PLY de python.

Alcance

El alcance del compilador se definirá para los condicionales **if – else** y funciones (**def**) definidas dentro de Python con paradigma estructural y funcional. Los casos utilizados serán los 10 ejercicios que se presentaran a continuación:

Ejercicio 1:

- Escriba un programa que pida la nota de un estudiante y diga que si es mayor a 6 está aprobado y menor a 6 está reprobado.

Ejercicio 2:

- Escribir un programa que pregunte al usuario su edad y muestre por pantalla si es mayor de edad o no.

Ejercicio 3:

- Escribir un programa que determine el precio a pagar en una entrada del cine, el valor de los tickets es 5,50, sabiendo que:
 - Los niños de 0-5 no pagan
 - De 6 a 12 pagan la mitad

Ejercicio 4:

- Escriba un programa que determine si un número es par o impar.

Ejercicio 5:

- Los empleados de una fábrica trabajan en turno diurno. Se desea calcular el jornal diario de acuerdo con los siguientes puntos: "
 - La tarifa de horas diurnas es de \$40. "
 - Caso de ser Domingo, la tarifa se incrementará en \$100 en el turno diurno

Ejercicio 6:

- Determinar la cantidad de dinero que recibirá un trabajador por concepto de las horas extras trabajadas en una empresa, sabiendo que cuando las horas de trabajo exceden de 40, el resto se consideran horas extras y que estas se pagan al doble de una hora normal.

Ejercicio 7:

- Escriba una función que dado un radio calcule el area de un circulo, dado su radio.

Ejercicio 8:

- Escriba una función que dado un dividendo y un divisor devuelva su división pero si el divisor es cero retorne 'No se puede'

Ejercicio 9:

- Escriba un programa que determine si el 19 es número primo.

Ejercicio 10:

- Escriba una función que agregue un elemento a una lista.

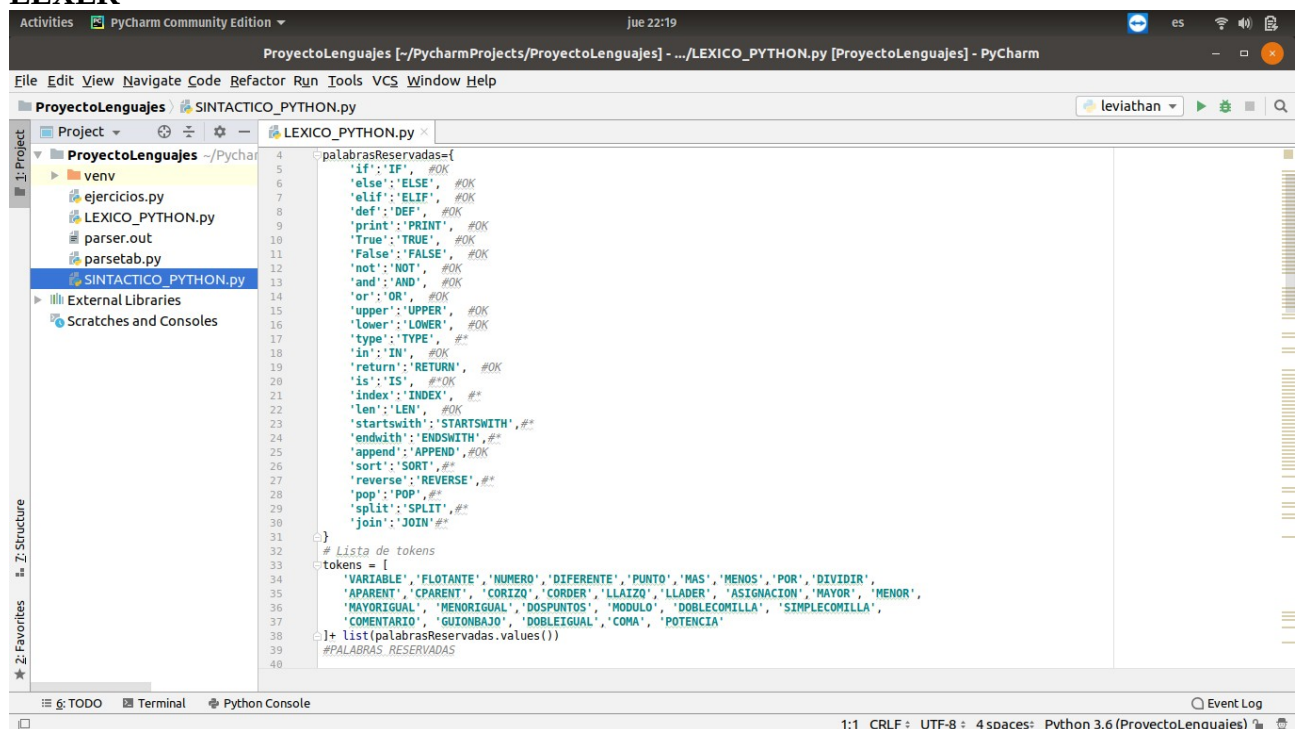
Metodología

La construcción del lexer validará los lexemas presentes en las instrucciones a evaluar por el mismo, estos también serán analizados a través de los patrones previamente definidos para los tokens. Estos patrones se definen a través de expresiones regulares, las cuales se encargan de limitar el número de lexemas posibles.

El objeto lex se encarga de realizar el análisis léxico, validando todos los lexemas en una cadena enviada de manera arbitraria, dicha cadena será válida si no existe algún lexema no válido en ella. Una vez que el conjunto de cadena es validado dentro del lenguaje, se procede a la construcción del árbol sintáctico a través del objeto yacc el cual se encarga de emular la parte sintáctica del lenguaje, esta parte se define a través de reglas BNF previamente creadas por los autores de este proyecto.

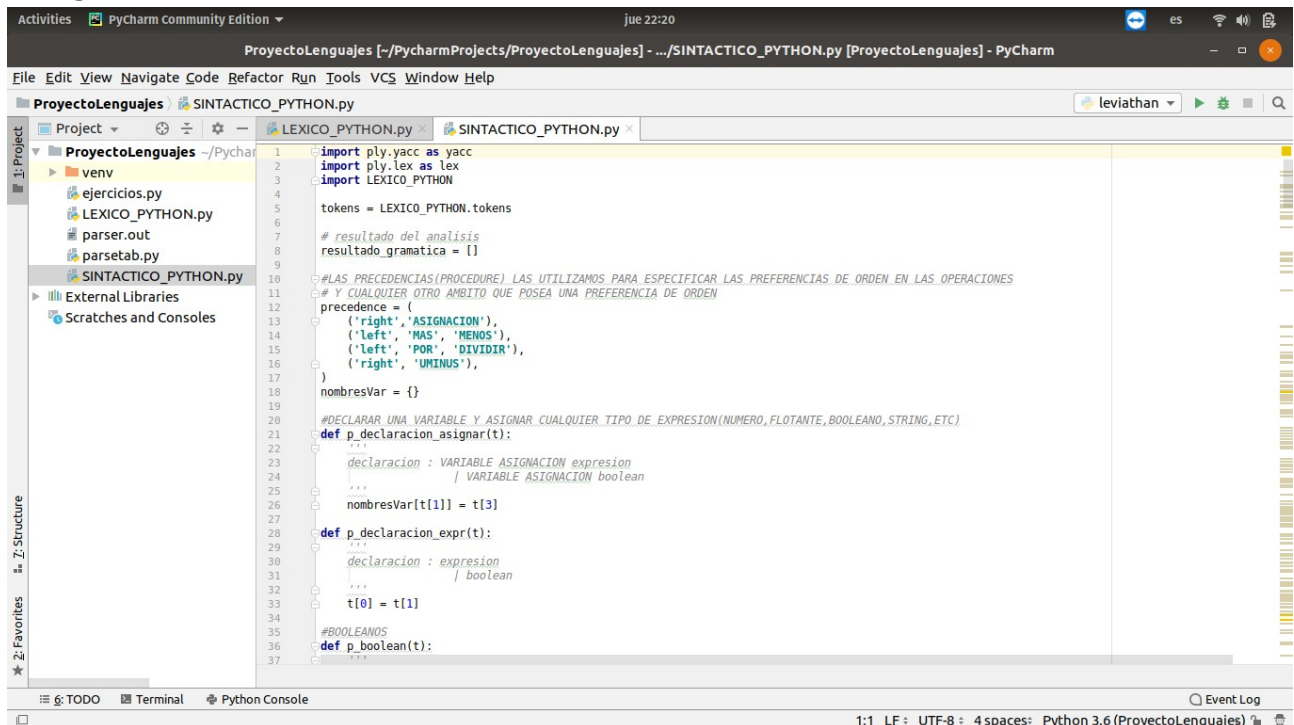
Anexos

LEXER



```
4 palabrasReservadas={
5     'if':'IF', #OK
6     'else':'ELSE', #OK
7     'elif':'ELIF', #OK
8     'def':'DEF', #OK
9     'print':'PRINT', #OK
10    'True':'TRUE', #OK
11    'False':'FALSE', #OK
12    'not':'NOT', #OK
13    'and':'AND', #OK
14    'or':'OR', #OK
15    'upper':'UPPER', #OK
16    'lower':'LOWER', #OK
17    'type':'TYPE', #*
18    'in':'IN', #OK
19    'return':'RETURN', #OK
20    'is':'IS', #*OK
21    'index':'INDEX', #*
22    'len':'LEN', #OK
23    'startswith':'STARTSWITH', #*
24    'endwith':'ENDSWITH', #*
25    'append':'APPEND', #OK
26    'sort':'SORT', #*
27    'reverse':'REVERSE', #*
28    'pop':'POP', #*
29    'split':'SPLIT', #*
30    'join':'JOIN', #*
31 }
32 # Lista de tokens
33 tokens = [
34     'VARIABLE', 'FLOTANTE', 'NUMERO', 'DIFERENTE', 'PUNTO', 'MAS', 'MENOS', 'POR', 'DIVIDIR',
35     'APARENT', 'CPARENT', 'CORIZO', 'CORDER', 'LLAIZO', 'LLADER', 'ASIGNACION', 'MAYOR', 'MENOR',
36     'MAYORIGUAL', 'MENORIGUAL', 'DOSPUNTOS', 'MODULO', 'DOBLECOMILLA', 'SIMPLECOMILLA',
37     'COMENTARIO', 'GUIONBAJO', 'DOBLEIGUAL', 'COMA', 'POTENCIA'
38 ]
39 list(palabrasReservadas.values())
40 #PALABRAS RESERVADAS
```

PARSER



PyCharm Community Edition - jue 22:20

ProjectoLenguajes [-/PycharmProjects/ProjectoLenguajes] - .../SINTACTICO_PYTHON.py [ProjectoLenguajes] - PyCharm

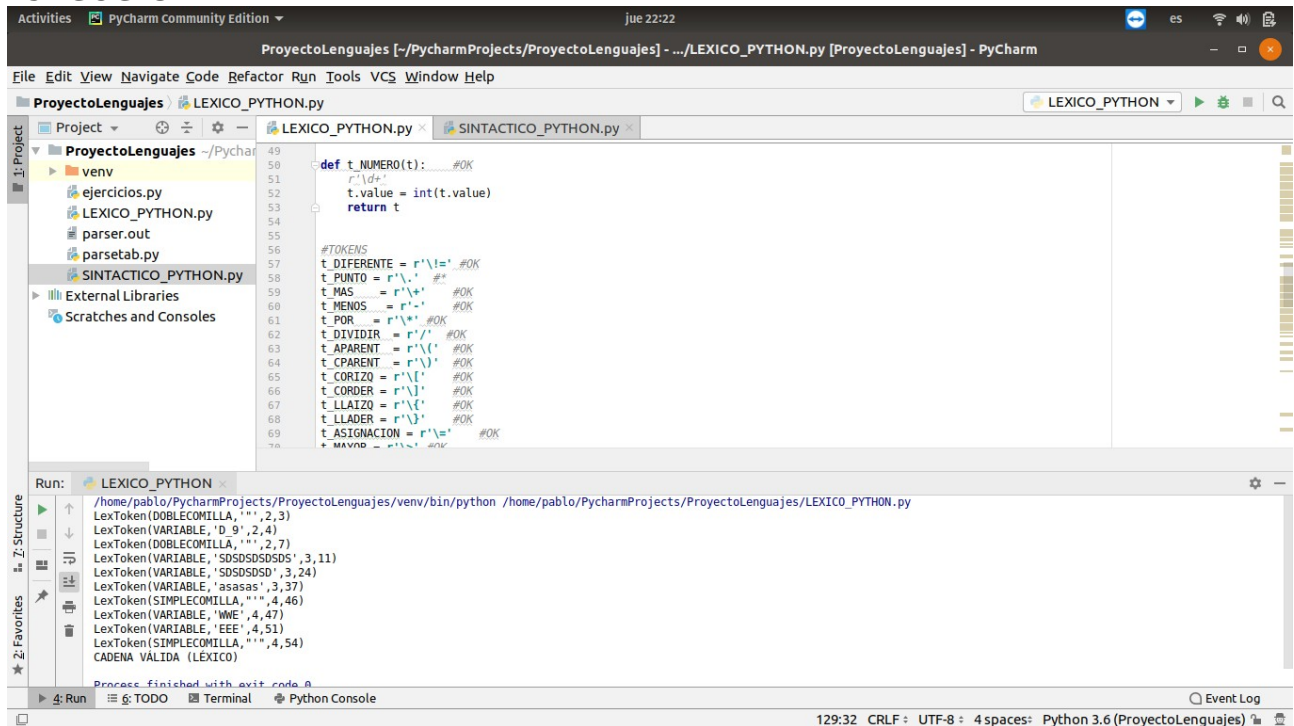
File Edit View Navigate Code Refactor Run Tools VCS Window Help

ProjectoLenguajes SINTACTICO_PYTHON.py

```
1 import ply.yacc as yacc
2 import ply.lex as lex
3 import LEXICO_PYTHON
4
5 tokens = LEXICO_PYTHON.tokens
6
7 # resultado del analisis
8 resultado_gramatica = []
9
10 #LAS PRECEDENCIAS(PROCEDURE) LAS UTILIZAMOS PARA ESPECIFICAR LAS PREFERENCIAS DE ORDEN EN LAS OPERACIONES
11 # Y CUALQUIER OTRO AMBITO QUE POSEA UNA PREFERENCIA DE ORDEN
12 precedence = (
13     ('right', 'ASIGNACION'),
14     ('left', 'MAS', 'MENOS'),
15     ('left', 'POR', 'DIVIDIR'),
16     ('right', 'UNIMINUS'),
17 )
18 nombresVar = {}
19
20 #DECLARAR UNA VARIABLE Y ASIGNAR CUALQUIER TIPO DE EXPRESION(NUMERO, FLOTANTE, BOOLEANO, STRING, ETC)
21 def p_declaracion_asignar(t):
22     """
23     declaracion : VARIABLE ASIGNACION expresion
24                 | VARIABLE ASIGNACION boolean
25     """
26     nombresVar[t[1]] = t[3]
27
28 def p_declaracion_expr(t):
29     """
30     declaracion : expresion
31                 | boolean
32     """
33     t[0] = t[1]
34
35 #BOOLEANOS
36 def p_boolean(t):
37     """
```

1:1 LF : UTF-8 : 4 spaces: Python 3.6 (ProjectoLenguajes)

EJECUCIÓN LEXER



PyCharm Community Edition - jue 22:22

ProjectoLenguajes [-/PycharmProjects/ProjectoLenguajes] - .../LEXICO_PYTHON.py [ProjectoLenguajes] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

ProjectoLenguajes LEXICO_PYTHON.py

```
49 def t_NUMERO(t):
50     """
51     t.value = int(t.value)
52     return t
53
54 #TOKENS
55 t_DIFERENTE = r'\!=' #OK
56 t_PUNTO = r'\.' #OK
57 t_MAS = r'\+' #OK
58 t_MENOS = r'\-' #OK
59 t_POR = r'\*' #OK
60 t_DIVIDIR = r'\/' #OK
61 t_APARENT = r'\(' #OK
62 t_CPARENT = r'\)' #OK
63 t_CORIZO = r'\[' #OK
64 t_CORDER = r'\]' #OK
65 t_LLAIZO = r'\{' #OK
66 t_LLADER = r'\}' #OK
67 t_ASIGNACION = r'\=' #OK
68 t_MAYORD = r'\>' #OK
```

Run: LEXICO_PYTHON

/home/pablo/PycharmProjects/ProjectoLenguajes/venv/bin/python /home/pablo/PycharmProjects/ProjectoLenguajes/LEXICO_PYTHON.py

```
LexToken(DOUBLECOMILLA, '"', 2, 3)
LexToken(VARIABLE, 'D 9', 2, 4)
LexToken(DOUBLECOMILLA, '"', 2, 7)
LexToken(VARIABLE, 'SDSDSDSDSDS', 3, 11)
LexToken(VARIABLE, 'SDSDSDSD', 3, 24)
LexToken(VARIABLE, 'asasas', 3, 37)
LexToken(SIMPLECOMILLA, "'", 4, 46)
LexToken(VARIABLE, 'wwe', 4, 47)
LexToken(VARIABLE, 'EEE', 4, 51)
LexToken(SIMPLECOMILLA, "'", 4, 54)
CADENA VALIDA (LEXICO)
```

Process finished with `exit` code 0

129:32 CRLF : UTF-8 : 4 spaces: Python 3.6 (ProjectoLenguajes)

PARSER.OUT

```
Grammar
Rule 0      S' -> declaracion
Rule 1      declaracion -> VARIABLE ASIGNACION expresion
Rule 2      declaracion -> VARIABLE ASIGNACION declaracion
Rule 3      declaracion -> expresion
Rule 4      declaracion -> VARIABLE
Rule 5      declaracion -> FLOTANTE
Rule 6      declaracion -> expr_funcion
Rule 7      declaracion -> lista
Rule 8      declaracion -> expr_def_funcion
Rule 9      declaracion -> expr_return
Rule 10     declaracion -> if
Rule 11     declaracion -> expr_print
Rule 12     declaracion -> append
Rule 13     declaracion -> len
Rule 14     declaracion -> input
Rule 15     declaracion -> in
Rule 16     declaracion -> is
Rule 17     declaracion -> join
Rule 18     declaracion -> int
Rule 19     declaracion -> upper
Rule 20     declaracion -> lower
Rule 21     expresion -> TRUE
Rule 22     expresion -> FALSE
Rule 23     expresion -> expresion MAS expresion
Rule 24     expresion -> expresion MENOS expresion
Rule 25     expresion -> expresion POR expresion
Rule 26     expresion -> expresion DIVIDIR expresion
Rule 27     expresion -> expresion POTENCIA expresion
Rule 28     expresion -> expresion MODULO expresion
Rule 29     expresion -> VARIABLE MAS VARIABLE
Rule 30     expresion -> VARIABLE MENOS VARIABLE
Rule 31     expresion -> VARIABLE POR VARIABLE
Rule 32     expresion -> VARIABLE DIVIDIR VARIABLE
Rule 33     expresion -> VARIABLE POTENCIA VARIABLE
Rule 34     expresion -> VARIABLE MODULO VARIABLE
```

PARSETAB.PY

```
# pylint: disable=W,C,R
tabversion = '3.10'

_lr_method = 'LALR'
_lr_signature = 'rightASIGNACIONleftMASMENOSleftPORDIVIDIRrightUMINUSAND APARENT APPEND ASIGNACION CADENACOMILLADOBLE CADENACOMILLASIN'
_lr_action_items = {'VARIABLE': ([0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 36, 37, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1399, 1400, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1440, 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1500, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1567, 1568, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1587, 1588, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1600, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1610, 1611, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1620, 1621, 1622, 1623, 1624, 1625, 1626, 1627, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1660, 1661, 1662, 1663, 1664, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1690, 1691, 1692, 1693, 1694, 1695, 1696, 1697, 1698, 1699, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1720, 1721, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1729, 1730, 1731, 1732, 1733, 1734, 1735, 1736, 1737, 1738, 1739, 1740, 1741, 1742, 1743, 1744, 1745, 1746, 1747, 1748, 1749, 1750, 1751, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788, 1789, 1790, 1791, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1810, 1811, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1820, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1847, 1848, 1849, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1860, 1861, 1862, 1863, 1864, 1865, 1866, 1867, 1868, 1869, 1870, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879, 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1890, 1891, 1892, 1893, 1894, 1895, 1896, 1897, 1898, 1899, 1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911, 1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933, 1934, 193
```