417 - Word Index

```
"""
UVA 417 - Word Index
Searching strategy: DFS
"""

from queue import Queue

M = {}
true, false = True, False


def inc_char(value):
    return chr(ord(value) + 1)


def generate_positions():
    q = Queue()
    c = 'a'
    while c <= 'z':
        q.put(c)
        c = inc_char(c)

    cnt = 1

    while not q.empty():
        s = q.get()
        M[s] = cnt
        cnt += 1

        if len(s) is 5:
            continue

        c = inc_char(s[len(s) - 1])
        while c <= 'z':
            q.put(s + c)
            c = inc_char(c)


generate_positions()

more_entry = true
while more_entry:
    try:
        s = input()
        if s in M:
            print(M[s])
        else:
            print(0)
    except EOFError:
        more_entry = false
```

441 - Lotto

```
"""
UVA 441 - Lotto
Searching strategy: DFS
"""

true, false = True, False
A, ans, n = [], [0] * 6, 0


# region Method Descriptions


def scan(t=int):
    scanned = input().split()
    len_scan = len(scanned)

    for i in range(len_scan):
        scanned[i] = t(scanned[i])

    return scanned


def dfs(idx, i):
    if idx == 6:
        print(ans[0], end='')
        i = 1
        while i < 6:
            print('', ans[i], end='')
            i += 1
        print('')
        return
    while i < n:
        ans[idx] = A[i]
        dfs(idx + 1, i + 1)
        i += 1


# endregion


first = 1
while true:
    line = scan()
    n = line[0]
    if n is 0:
        break
    if first is not 1:
        print('')
    first = 0
    A = line[1:]
    dfs(0, 0)
```

10004 - Bicoloring

```
"""
UVA 10004 - Bicoloring
Searching strategy: DFS
"""

true, false = True, False


# region Method Definitions
def init_array(n1, n2=None, value=false):
    if n2 is None:
        n2 = n1
    return [[value] * n2 for i in range(n1)]


def scan(t=int):
    scanned = input().split()
    len_scan = len(scanned)
    if len_scan is 1:
        return t(scanned[0])
    for i in range(len_scan):
        scanned[i] = t(scanned[i])

    return scanned


# endregion

while true:
    n = scan()
    if n is 0:
        break

    a = init_array(n)
    color = [-1] * n
    color[0] = 0

    num_edge = scan()
    for i in range(num_edge):
        x, y = scan()
        a[x][y] = true
        a[y][x] = true

    stack = [0]
    colorable = true
    while colorable and stack:
        i = stack.pop()
        for j in range(n):
            if a[i][j]:
                if color[j] is -1:
                    color[j] = color[i] ^ 1
                    stack.append(j)
                elif color[j] == color[i]:
                    colorable = false
                    break

    if colorable:
        print("BICOLORABLE.")
    else:
        print("NOT BICOLORABLE.")
```

11396 - Claw Decomposition

```
"""
UVA 11396 - Claw Decomposition
Searching strategy: BFS
"""

from queue import Queue

true, false = True, False
black, white = 1, 0


# region Method Descriptions

def init_array(n):
  return [[] for i in range(n)]


def scan(t=int):
  scanned = input().split()
  len_scan = len(scanned)
  if len_scan is 1:
    return t(scanned[0])

  for i in range(len_scan):
    scanned[i] = t(scanned[i])

  return scanned


# endregion

while true:
  V = scan()
  if V is 0:
    break

  graph = init_array(V + 1)
  while true:
    u, v = scan()
    if u is v and v is 0:
      break
    graph[u].append(v)
    graph[v].append(u)
```

```
q = Queue()
q.put(1)
colors = [-1] * (V + 1)
colors[1] = 1

yes = true
while not q.empty() and yes:
    u = q.get()
    for v in graph[u]:
        if colors[v] is -1:
            colors[v] = 1 - colors[u]
            q.put(v)
        elif colors[v] is colors[u]:
            yes = false
            break

print("YES" if yes else "NO")
```