Anagrams

return resultados

```
def find anagrams(letters, previous word, shortest):
     results=set()
     for w in find_words(letters):
          if len(w)>= shortest and w > previous word:
              remainder = removed(letters,w)
              if remainder:
                   for rest in find_anagrams(remainder,w,shortest):
                       results.add(w+' '+rest)
              else:
                   results.add(w)
     return results
Blank Tiles
 # UPDATED: Removed '_' =0 and instead added keys for each lowerca
 PUNTOS = dict(A=1, B=3, C=3, D=2, E=1, F=4, G=2, H=4, I=1, J=8, I
 puntos = defaultdict(int)
 for 1 in ascii_lowercase: puntos[1]
 PUNTOS.update(puntos)
def bonus template(quadrant):
      return mirror(map(mirror, quadrant.split()))
 def mirror(secuencia): return secuencia + secuencia[-2::-1]
 SCRABBLE = bonus_template("""
Boggle
- def boggle_words(tablero, longitud_minima=3):
     resultados = set()
     filas = size(tablero)
     for (i,sq) in enumerate(tablero):
        if sq != BORDE:
            find_boggle_words(tablero, filas, i, sq, longitud_minima, resultados)
     return resultados
def find_boggle_words(tablero, filas, i, pre, longitud_minima, resultados=None, visitados=set()):
     if resultados is None: resultados = set()
     if pre in PALABRAS and len(pre) >= longitud_minima:
        resultados.add(pre)
     if pre in PREFIJOS:
        ultimo = (i, tablero[i])
        visitados.add(ultimo)
        for nbr in vecinos(i, filas):
            if tablero[nbr] != BORDE and (nbr, tablero[nbr]) not in visitados:
                find_boggle_words(tablero, filas, nbr,
                                pre + tablero[nbr], longitud_minima, resultados, visitados)
        visitados.remove(ultimo)
```