main

var possible = allPossibleAnswers

var outcome = ""

var attempt = ""

while outcome is not "\*\*\*\*\*"

attempt = bestAttempt(possible, validWords)

print(attempt)

print("Please mark the attempt")

outcome = input

possible = possibleAnswersAfterAttempt(possible, attempt, outcome).asList()

end while

end main

function isGreen(attempt String, target String, n Int) as Bool ->

target[n] is attempt[n]

function setChar(word String, n Int, newChar Char) as String ->

word[..n] + newChar + word[n+1..]

function setAttemptIfGreen(attempt String, target String, n Int) as String ->

if attempt.isGreen(target, n) then attempt.setChar(n, '\*') els`e attempt

function setTargetIfGreen(attempt String, target String, n Int) as String ->

if attempt.isGreen(target, n) then target.setChar(n, '.') else target

function isYellow(attempt String, target String, n Int) as Bool ->   
 target.contains(attempt[n])

function isAlreadyMarkedGreen(attempt String, n Int) as Bool -> attempt[n] is '\*'

function setAttemptIfYellow(attempt String, target String, n Int) as String ->

if isAlreadyMarkedGreen(attempt, n) then attempt

else if attempt.isYellow(target, n) then attempt.setChar(n, '+')

else attempt.setChar(n, '\_')

function setTargetIfYellow(attempt String, target String, n Int) as String ->

if attempt.isAlreadyMarkedGreen(n) then target

else if attempt.isYellow(target, n) then target.setChar(target.indexOf(attempt[n]), '.')

else target

function evaluateGreens(attempt String, target String) as (String, String) ->

range(5).reduce((attempt, target), \_

lambda a, x-> (setAttemptIfGreen(a.attempt, a.target, x), setTargetIfGreen(a.attempt, a.target, x)))

function evaluateYellows(attempt String, target String) as (String, String) ->

range(5).reduce((attempt, target), \_

lambda(a, t), x -> (setAttemptIfYellow(a, t, x), setTargetIfYellow(a, t, x)))

function markAttempt(attempt String, target String) as String ->

let (attemptAfterGreens, x) = evaluateGreens(attempt, target) in

attemptAfterGreens.evaluateYellows(targetAfterGreens)[0]

function possibleAnswersAfterAttempt(prior Iterable<String>, attempt String, mark String) as Iterable<String> ->

prior.filter(lambda w -> markAttempt(attempt, w) is mark)

function wordCountRemainingAfterAttempt(possibleAnswers Iterable<String>, attempt String) as Int ->

let groups = possibleAnswers.groupBy(lambda w -> markAttempt(attempt, w)) in

groups.max(lambda g -> g.count())

function allRemainingWordCounts(possAnswers List<String>, possAttempts Iterable<String>) as Iterable<(String, Int)> ->

possAttempts.map(lambda w -> (w, wordCountRemainingAfterAttempt(possAnswers, w)))

function betterOf(word1 (String, Int), word2 (String, Int), possAnswers Iterable< String >) as (String, Int) ->

let (w1, w1Count) = word1,

(w2, w2Count) = word2,

isBetter = w2Count < w1count,

isEqualAndPossAnswer = w2count is w1count and possAnswers.contains(w2) in

if isBetter or isEqualAndPossAnswer then word2 else word1

function bestAttempt(possAnswers Iterable<String>, possAttempts List<String>) as String ->

let wordCounts = allRemainingWordCounts(possAnswers, possAttempts) in

wordCounts.reduce(lambda bestSoFar, newWord -> betterOf(bestSoFar, newWord, possAnswers))[0]

constant allPossibleAnswers = {"ABACK","...","ZONAL" } // 2,309 words total

constant validWords = { "ABACK", "...", "ZYMIC" } // 12,947 words total