# Contents

**1 Basic Test Results** 2

**2 wordsearch.py** 3

# 1 Basic Test Results

```
Starting tests...
Sun Aug  8 21:50:11 IDT 2021
9b76209d9ed2f5f85955d38d8a3dbdd405ed5b5e  -


Archive:  /tmp/bodek.LeoVbi/intro2cse/ex5/yaniv.hayoun/presubmission/submission
  inflating: src/wordsearch.py


Running presubmit code tests...
5 passed tests out of 5 in test set named 'ex5'.
result_code    ex5    5    1
Done running presubmit code tests

Finished running the presubmit tests

Additional notes:

Make sure to thoroughly test your code.

```

# 2 wordsearch.py

```python
import os.path
import sys

def read_wordlist(filename):
    """A function that receives a name of file of a wordlist (each word in a line)
    and returns the words in a list."""
    if os.path.isfile(filename) is False:
        return False
    with open(filename, 'r') as f:
        return [word.strip('\n') for word in f.readlines()]


def read_matrix(filename):
    """A function that receives a name of a file of matrix (each matrix's line in a seperate line,
     the line's characters seperated by commas) and returns a two-dimensional list of the matrix."""
    if os.path.isfile(filename) is False:
        return False
    with open(filename, 'r') as f:
        return [line.strip('\n').split(',') for line in f]


def find_directions(directions):
    """A function that receives a string of letters which represts directions, and returns it as a set.
    (Directions represented by: u for up, d for down, r for right, l for left, w for right up,
    x for left up, y for right down, z for left down. For uncertified direction character in the string,
     the function returns False)."""
    if set(directions) <= {'u','d','r','l','w','x','y','z'}:
        return set(directions)
    else:
        return False


def direction_strings(mat, direction):
    """A function that recevies a matrix in a two-dimensional list, and a a letter string reoresenting a direction,
    and returns a list of all the combination of strings in that direction from the matrix."""
    if direction == 'r':  # Direction: from left to right.
        return [''.join(mat_line) for mat_line in mat]
    elif direction == 'l':  # Direction: from right to left.
        return [''.join(mat_line[::-1]) for mat_line in mat]
    elif direction == 'd':  # Direction: from up to down.
        return [''.join([mat_line[line_index] for mat_line in mat]) for line_index in range(len(mat[0]))]
    elif direction == 'u':  # Direction: from down to up.
        return [''.join([mat_line[line_index] for mat_line in mat][::-1]) for line_index in range(len(mat[0]))]
    elif direction == 'w':  # Direction: diagnol line to the right and up.
        return [''.join([mat[diagnol_line-line_index][line_index] for line_index in range(len(mat[0])) if
                         0 <= (diagnol_line-line_index) < len(mat)]) for diagnol_line in range(len(mat)+len(mat[0])-1)]
    elif direction == 'z':  # Direction: diagnol line to left and down.
        return [''.join([mat[diagnol_line-line_index][line_index] for line_index in range(len(mat[0]))
                         if 0 <= (diagnol_line-line_index) < len(mat)][::-1])
                for diagnol_line in range(len(mat)+len(mat[0])-1)]
    elif direction == 'x':  # Direction: diagnol line to left and up.
        return [''.join([mat[diagnol_line-line_index][-line_index-1] for line_index in range(len(mat[0])) if
                         0 <= (diagnol_line-line_index) < len(mat)]) for diagnol_line in range(len(mat)+len(mat[0])-1)]
    elif direction == 'y':  # Direction: diagnol line to right and down.
        return [''.join([mat[diagnol_line-line_index][-line_index-1] for line_index in range(len(mat[0]))
                         if 0 <= (diagnol_line-line_index) < len(mat)][::-1])
                for diagnol_line in range(len(mat)+len(mat[0])-1)]
```

```python
def word_in_strings(word, mat, directions):
    """A function that receives a word, a matrix in a two-dimensional list, and a string of letters represents
     directions, and returns all the word occurances in the matrix based on given directions (a tuple of the word and
     the occurances count. if the count = 0, returns False)."""
    count = 0
    for direction in find_directions(directions):  # Check the word in all the given directions.
        strings = direction_strings(mat, direction)
        for string in strings:  # Check the word in all the direction strings.
            if len(word) > len(string):  # Irrelevant string.
                continue
            for string_index in range(len(string)-len(word)+1):
                for word_index in range(len(word)):  # Comparing the word indexes to each index of the string.
                    if word[word_index] != string[string_index+word_index]:  # If there is a dismatch,
                        break  # continue to compare the word indexes to the next index of the string.
                    elif word_index == (len(word)-1) and word[word_index] == string[string_index+word_index]:
                        count += 1  # Add to the counter for each match (over all the word indexes.
    if count == 0:
        return False
    else:
        return word, count


def find_words(word_list, matrix, directions):
    """A function that receives a list of words, a matrix in a two-dimensional list, and a string of letters represents
    directions, and returns a list of tuples of the word and the word occurances in the matrix (in given directions, if
    the occurances count is bigger than 0. The function returns False if given an uncertified direction character). """
    if find_directions(directions) == False:
        return False
    return [word_in_strings(word, matrix, directions) for word in word_list if word_in_strings(word, matrix, directions) is


def write_output(results, filename):
    """A function that receives results - a list of tuples of two objects (from find_words function)
     and a name for a file, and creates the file with the results by the given name (overrides an existing file)."""
    with open(filename, 'w') as the_results:
        for result in range(len(results)):
            x, y = results[result]
            a_list = str(x) + ',' + str(y)
            the_results.write(a_list+'\n')


def find_word_in_matrix():
    """A function that receives a name of a wordlist file (\location), a name of a matrix file (\location), a string of
    directions, and a name for output file, and creates a file by the output file name with all the occurances of the
    words in the given matrix file (only when the occurances bigger than 0) on the given directions.
    (Prints an informative error message in case of a problem with the provided parameters)."""
    if len(sys.argv) != 5:
        print('The number of parameters provided is not proper (solely 4 parameters needed).')
        return
    word_file = sys.argv[1]
    matrix_file = sys.argv[2]
    output_file = sys.argv[3]
    directions = sys.argv[4]
    if read_wordlist(word_file) is False:
        print('The wordlist file is not found.')
        return
    word_list = read_wordlist(word_file)
    if read_matrix(matrix_file) is False:
        print('the matrix file is not found.')
        return
    matrix_list = read_matrix(matrix_file)
    if find_words(word_list, matrix_list, directions) is False:
        print('An uncertofied direction character is given.')
        return
    the_results = find_words(word_list, matrix_list, directions)
    write_output(the_results, output_file)
```

```python
if __name__ == "__main__":
    find_word_in_matrix()
```