Nama : Serevina Sherly Maulida

Nim : 1227050123

Kelas : IF D

**1. Algoritma Brute Force untuk word search puzzle:**

1. Mula-mula, akan diambil kata pertama dari list kata yang akan dicari.
2. Pencarian akan dimulai dari sel ujung kiri atas.
3. Melakukan penyocokan ke arah kanan (timur). Jika panjang kata yang dicari tidak melebihi dari batas ujung puzzle, akan dicocokan untuk setiap karakter.
4. Jika pada langkah 3 kata belum ditemukan, lakukan penyocokan ke arah kanan-bawah (tenggara). Jika panjang kata yang dicari tidak melebihi dari batas ujung puzzle, akan dicocokan untuk setiap karakter.
5. Jika pada langkah 3 dan 4 kata belum ditemukan, lakukan penyocokan ke arah bawah (selatan). Jika panjang kata yang dicari tidak melebihi dari batas ujung puzzle, akan dicocokan untuk setiap karakter.
6. Jika pada langkah 3, 4, dan 5 kata belum ditemukan, lakukan penyocokan ke arah kiribawah (barat daya). Jika panjang kata yang dicari tidak melebihi dari batas ujung puzzle, akan dicocokan untuk setiap karakter.
7. Jika pada langkah 3, 4, 5, dan 6 kata belum ditemukan, lakukan penyocokan ke arah kiri (barat). Jika panjang kata yang dicari tidak melebihi dari batas ujung puzzle, akan dicocokan untuk setiap karakter.
8. Jika pada langkah 3, 4, 5, 6, dan 7 kata belum ditemukan, lakukan penyocokan ke arah kiri-atas (barat laut). Jika panjang kata yang dicari tidak melebihi dari batas ujung puzzle, akan dicocokan untuk setiap karakter.
9. Jika pada langkah 3, 4, 5, 6, 7, dan 8 kata belum ditemukan, lakukan penyocokan ke arah atas (utara). Jika panjang kata yang dicari tidak melebihi dari batas ujung puzzle, akan dicocokan untuk setiap karakter.
10. Jika pada langkah 3, 4, 5, 6, 7, 8, dan 9 kata belum ditemukan, lakukan penyocokan ke arah kanan-atas (timur laut). Jika panjang kata yang dicari tidak melebihi dari batas ujung puzzle, akan dicocokan untuk setiap karakter.
11. Jika kata masih belum ditemukan, pencarian akan geser satu sel dan melakukan iterasi pada langkah 3 sampai 10.
12. Setelah kata ditemukan, akan diambil kata berikutnya dari list kata yang dicari dan melakukan iterasi dimulai dari langkah 2.
13. Iterasi ini akan selesai hingga seluruh kata berhasil dicari atau telah dicocokan dengan puzzle.

***2. Source Code* dengan Bahasa Python**

import time

# Constants

CAPACITY = 100

class Puzzle:

    def \_\_init\_\_(self):

        self.content = [[''] \* CAPACITY for \_ in range(CAPACITY)]

        self.m = 0

        self.n = 0

class WordList:

    def \_\_init\_\_(self):

        self.content = []

        self.count = 0

def file2data(file\_name, P, WL):

    file\_path = f"./test/{file\_name}"

    with open(file\_path, 'r') as file:

        lines = file.readlines()

    # Initialize indices

    i = 0

    # Read puzzle content

    while i < len(lines) and lines[i].strip() != "":

        line = lines[i].strip().replace(" ", "")

        P.content[i][:len(line)] = line

        i += 1

    P.m = i

    P.n = len(line) if i > 0 else 0

    # Read word list content

    while i < len(lines):

        line = lines[i].strip()

        if line:

            WL.content.append(line)

            WL.count += 1

        i += 1

def print\_grid(P, found\_grid):

    for row in range(P.m):

        line = ''

        for col in range(P.n):

            if found\_grid[row][col]:

                line += P.content[row][col] + ' '

            else:

                line += '- '

        print(line)

def search\_word(P, W):

    length = len(W)

    found\_grid = [[False] \* CAPACITY for \_ in range(CAPACITY)]

    for i in range(P.m):

        for j in range(P.n):

            # Define the lambda function for each direction

            directions = {

                "East": lambda k: (i, j + k),

                "Southeast": lambda k: (i + k, j + k),

                "South": lambda k: (i + k, j),

                "Southwest": lambda k: (i + k, j - k),

                "West": lambda k: (i, j - k),

                "Northwest": lambda k: (i - k, j - k),

                "North": lambda k: (i - k, j),

                "Northeast": lambda k: (i - k, j + k)

            }

            for direction, get\_pos in directions.items():

                found = True

                for k in range(length):

                    row, col = get\_pos(k)

                    if not (0 <= row < P.m and 0 <= col < P.n):

                        found = False

                        break

                    if P.content[row][col] != W[k]:

                        found = False

                        break

                if found:

                    for k in range(length):

                        row, col = get\_pos(k)

                        found\_grid[row][col] = True

                    print(f"Word '{W}' found in direction '{direction}' starting at ({i}, {j}):")

                    print\_grid(P, found\_grid)

                    return

    print(f"Word '{W}' not found!")

def main():

    P = Puzzle()

    WL = WordList()

    # Read file

    file\_name = input("\nInput your test file name: ")

    print("\n")

    file2data(file\_name, P, WL)

    # Brute Force

    count = 0

    start = time.time()

    for i in range(WL.count):

        search\_word(P, WL.content[i])

        print("\n")

    end = time.time()

    duration = (end - start) \* 1e6  # Convert to microseconds

    print(f"Time taken: {duration:.0f} microseconds")

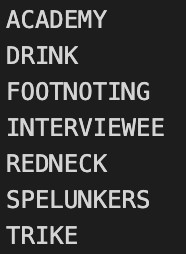
    print(f"Total comparisons: {count} letters\n\n")

if \_\_name\_\_ == "\_\_main\_\_":

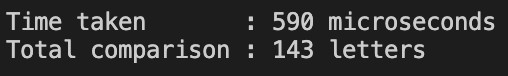
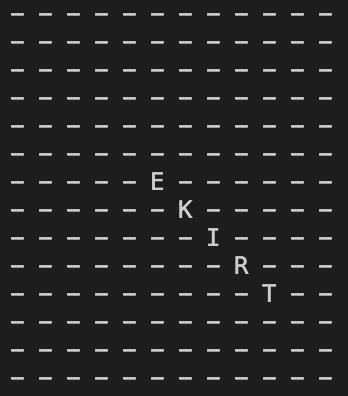
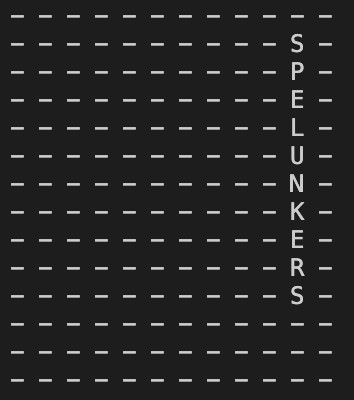
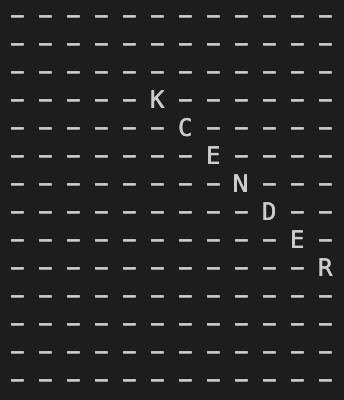
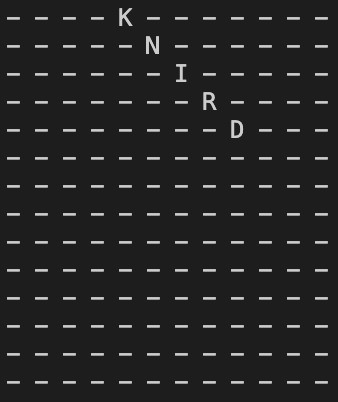
    main()

# *3. Screenshot* dari *Input* dan *Output* dari Program

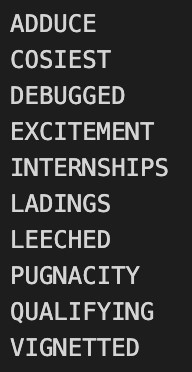
# 1. *Small* • Input 1



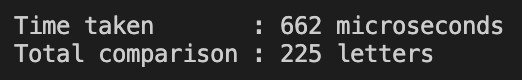
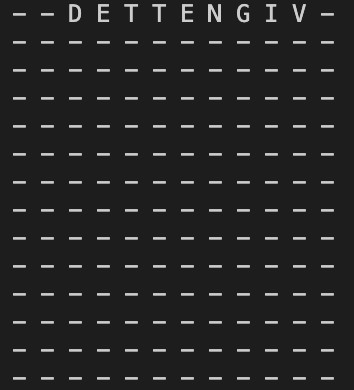
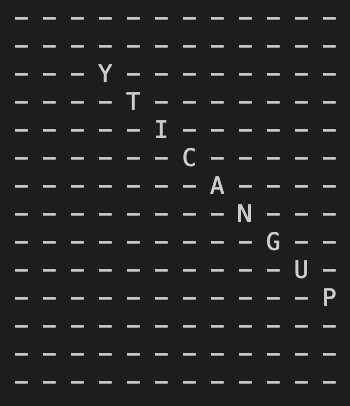
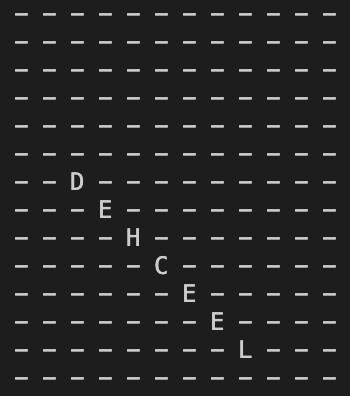
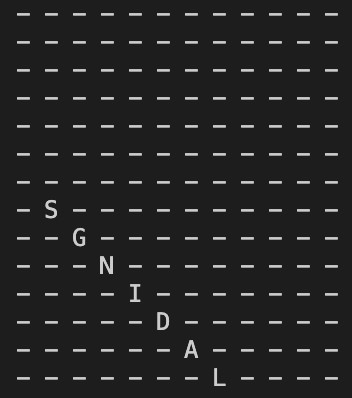
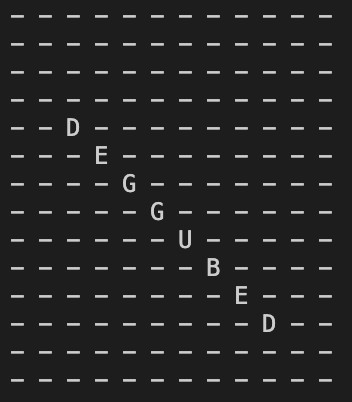
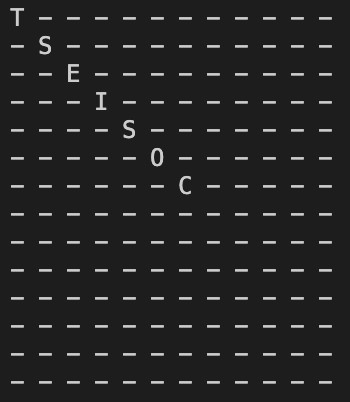
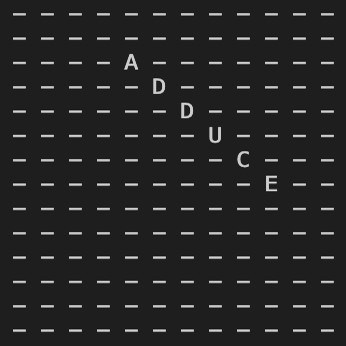
# • Output 1



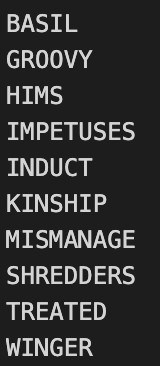
# • Input 2



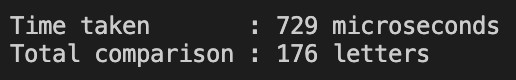
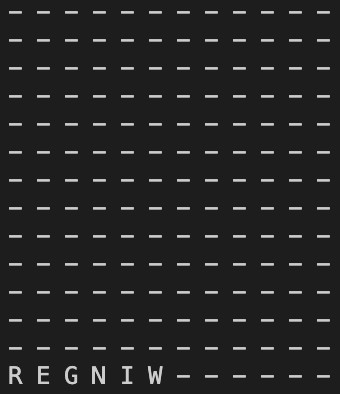
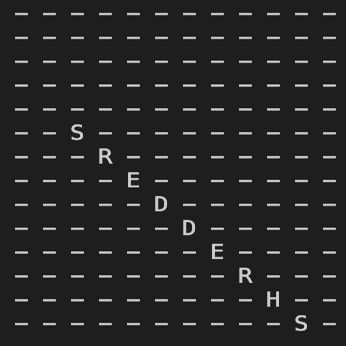
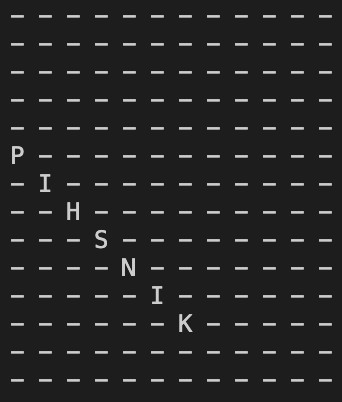
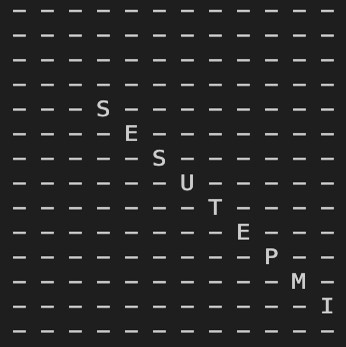
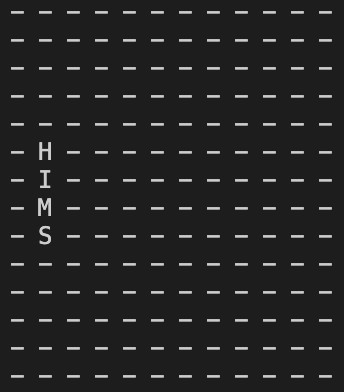
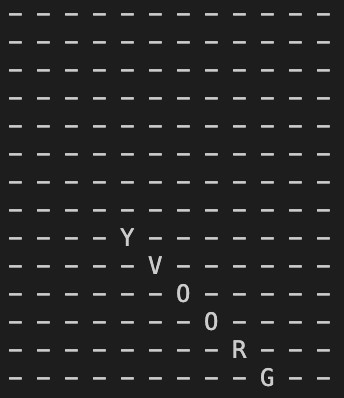
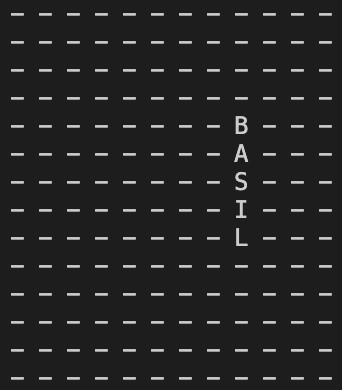
# •Ouput 2



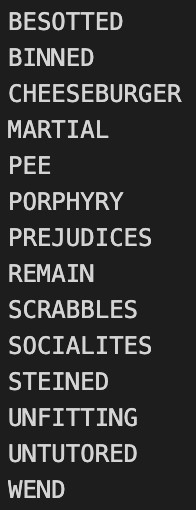
# • Input 3



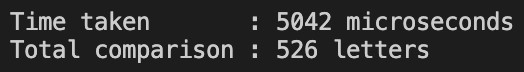
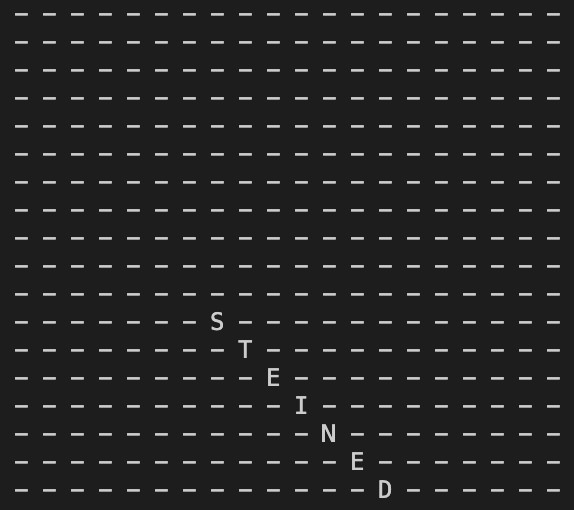
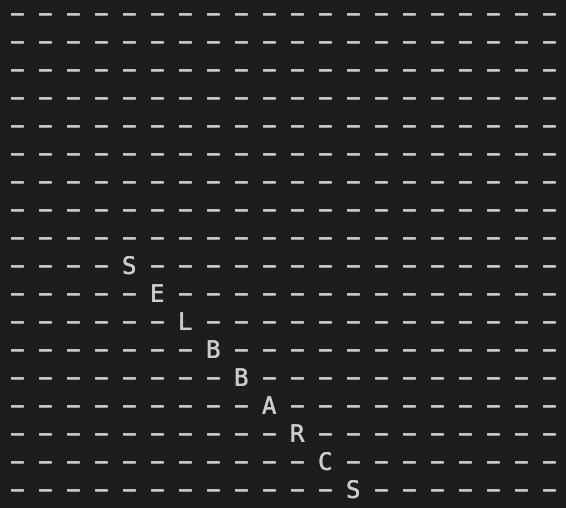
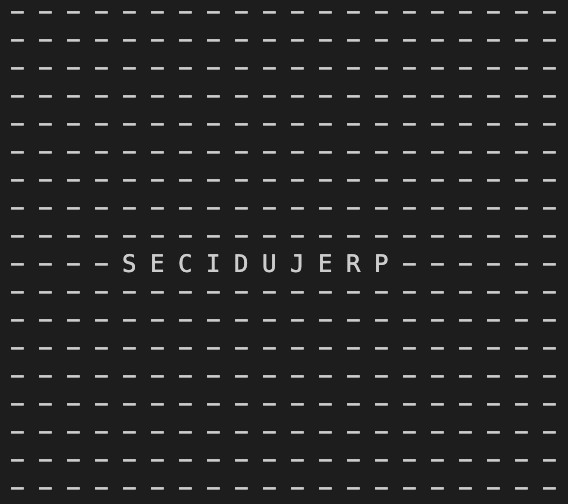
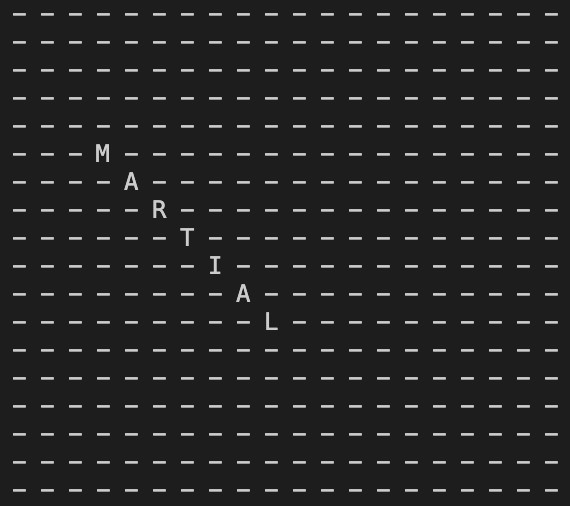
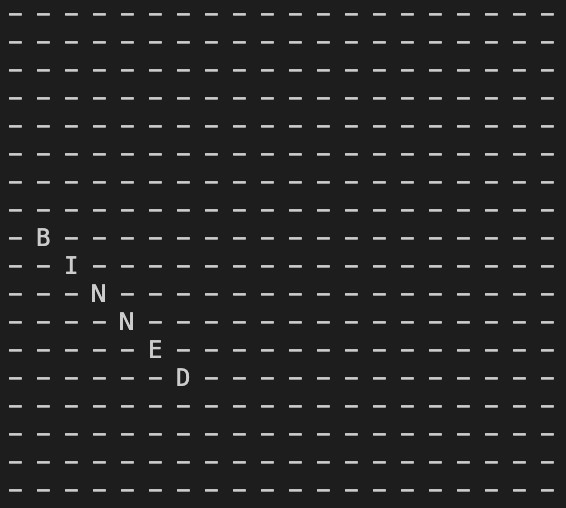
# • Output 3



# 2. *Medium* • Input 4



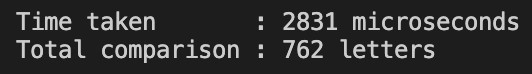
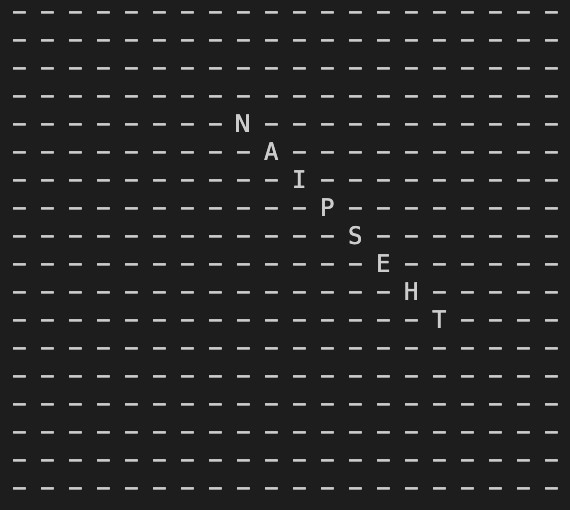
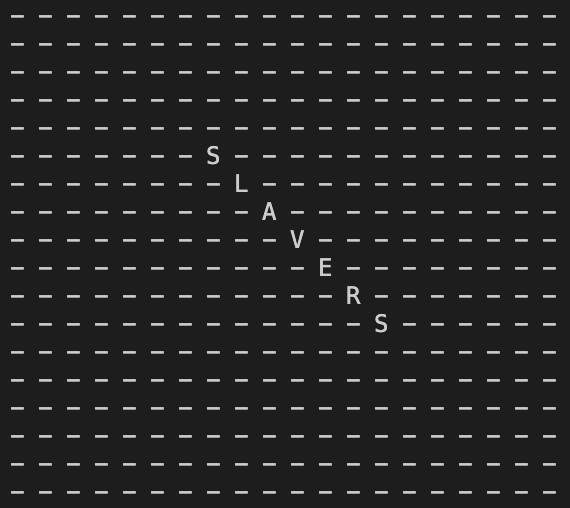
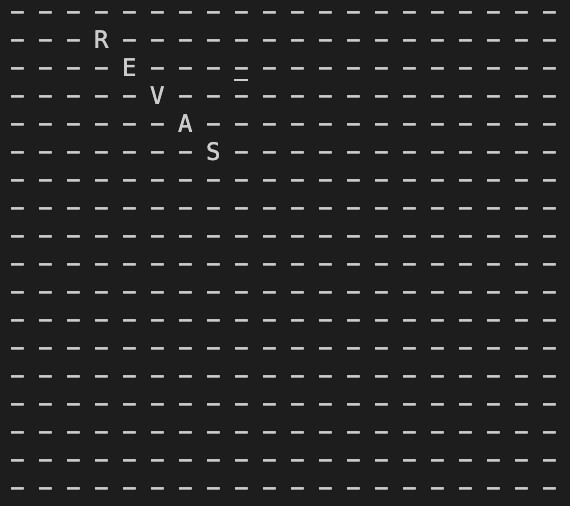
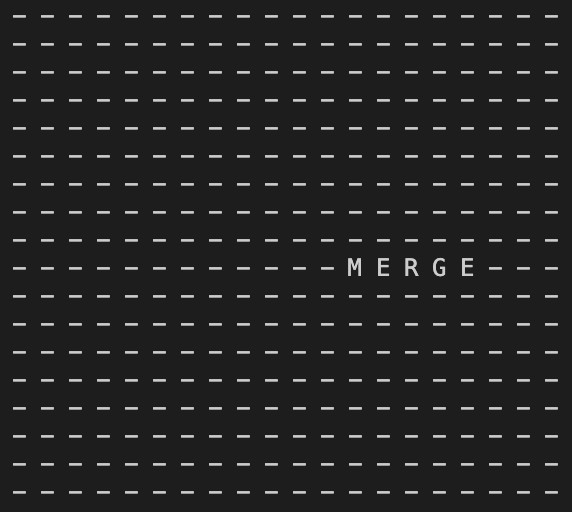
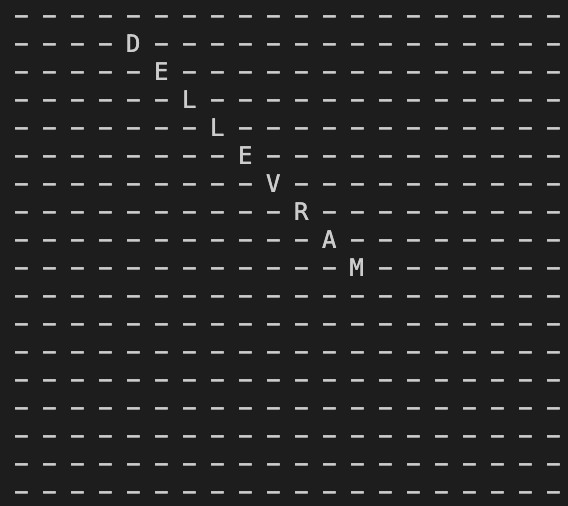
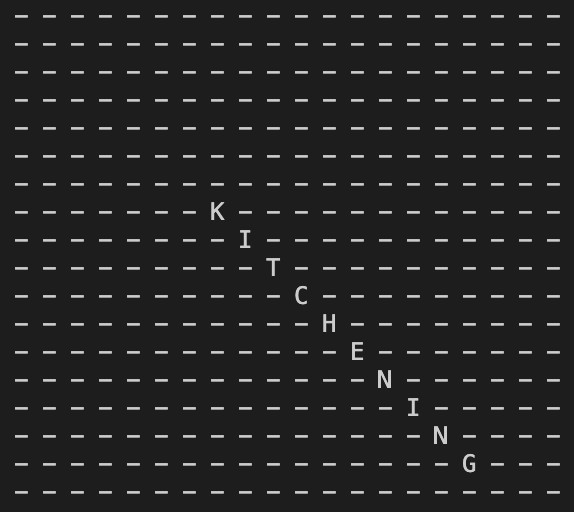
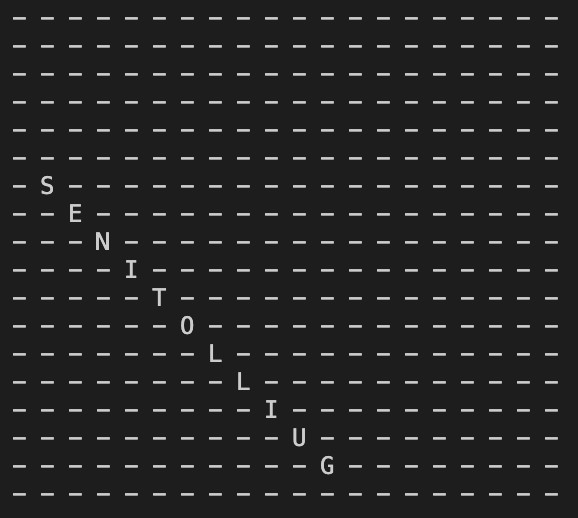
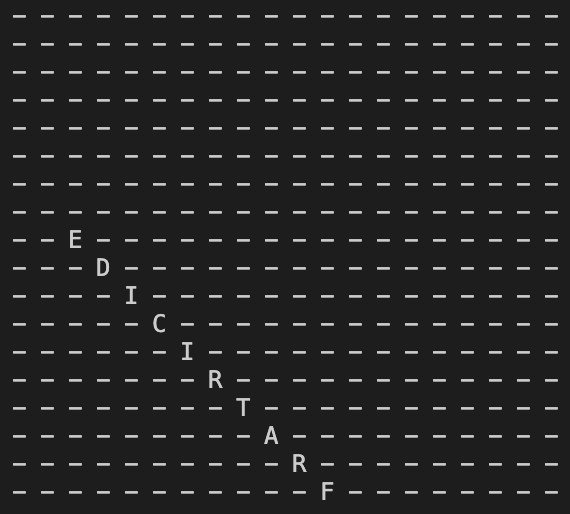
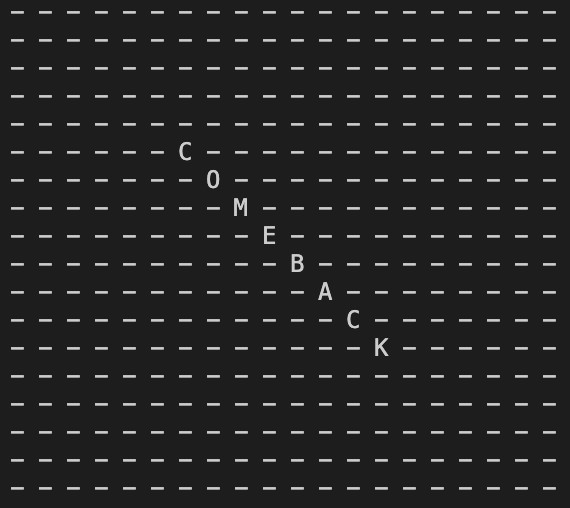
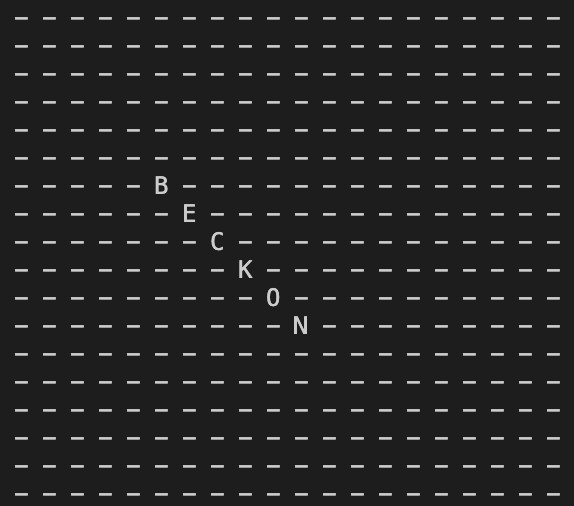
# • Output 4



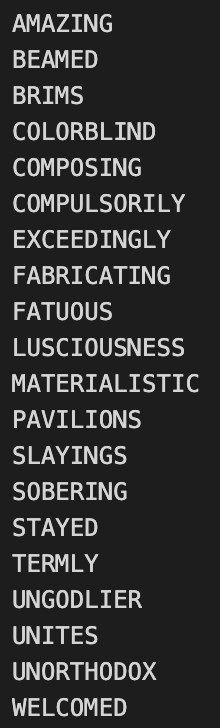
# • Input 5



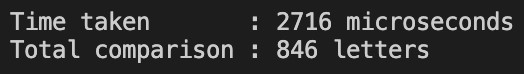
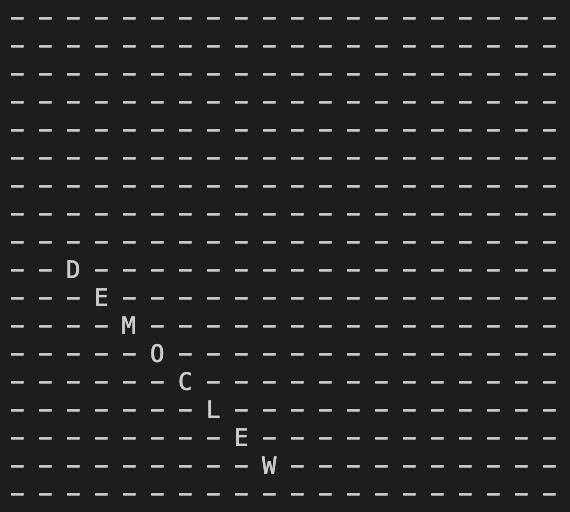
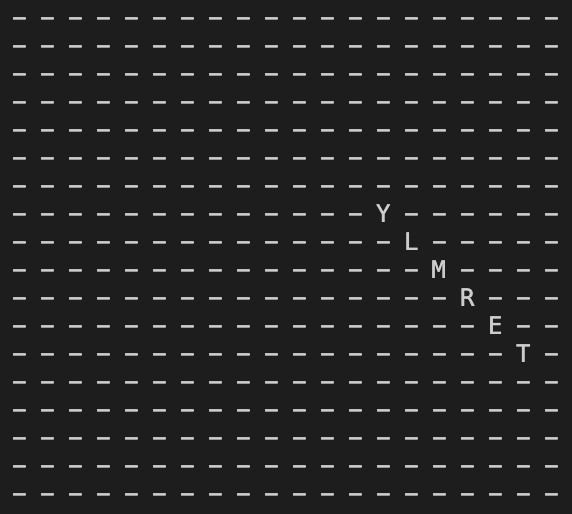
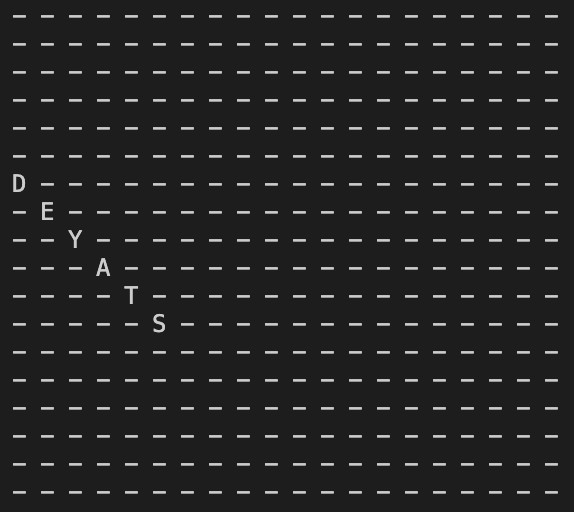
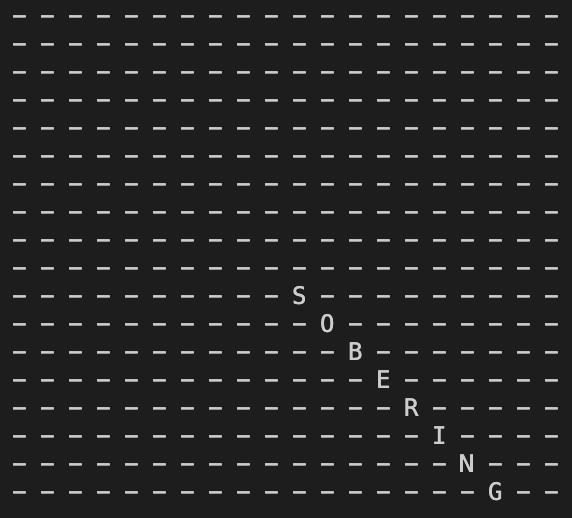
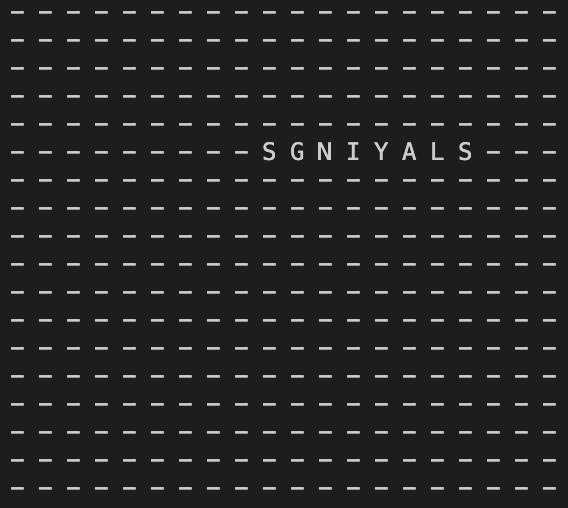
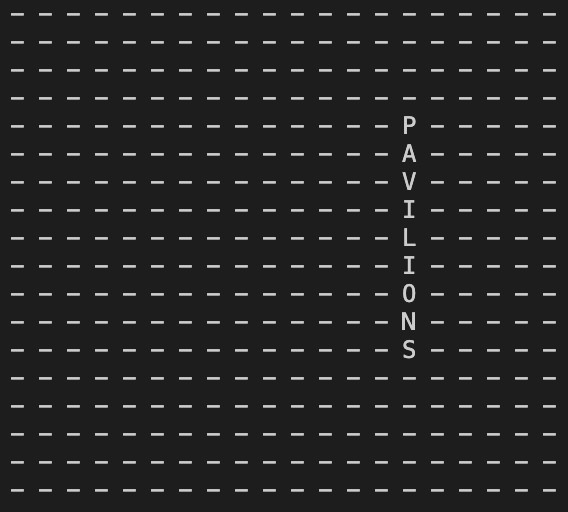
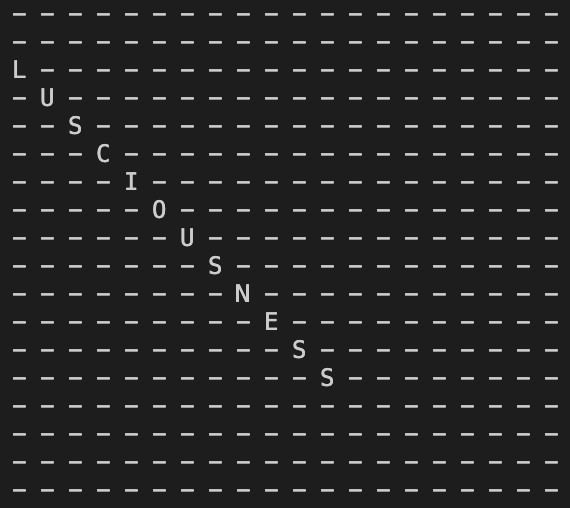
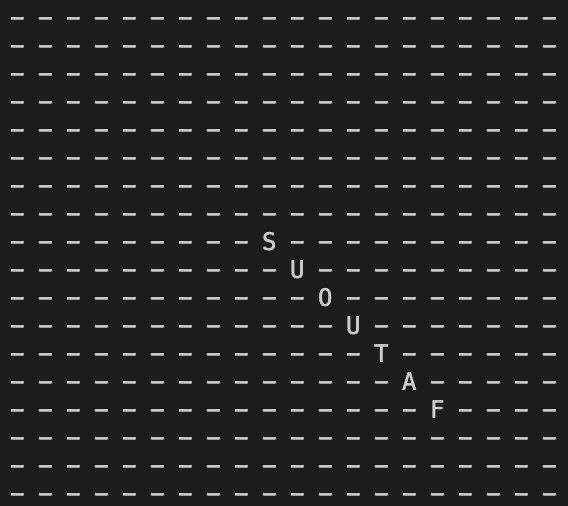
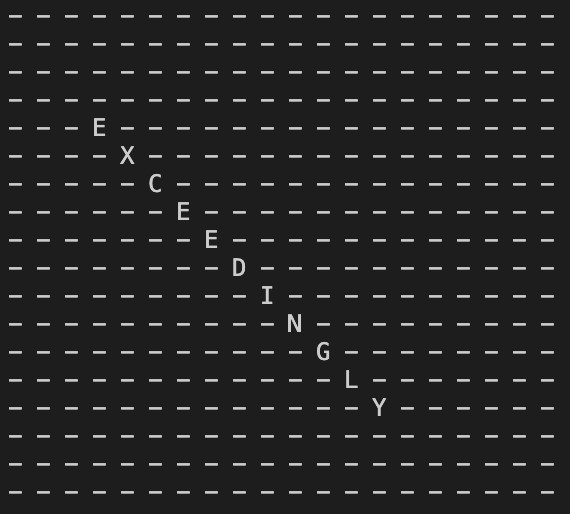
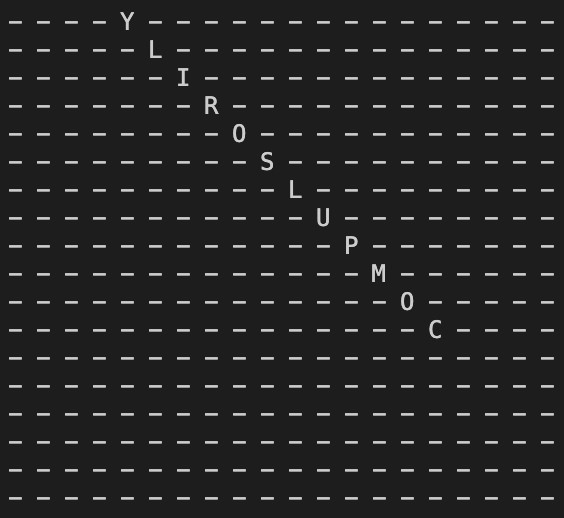
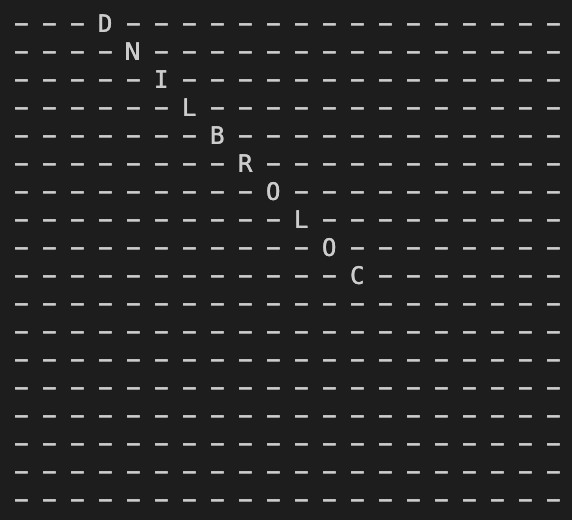
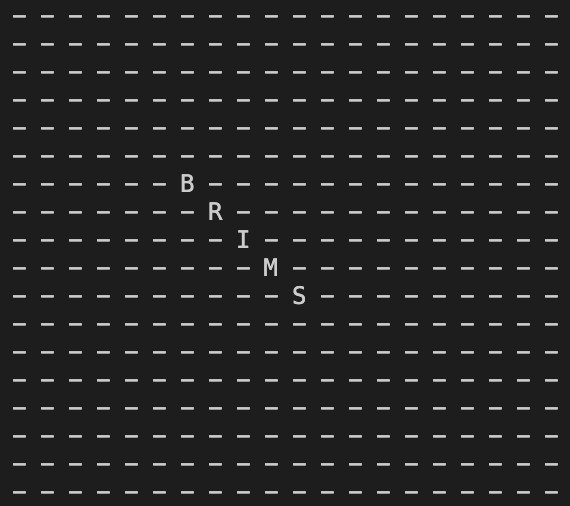
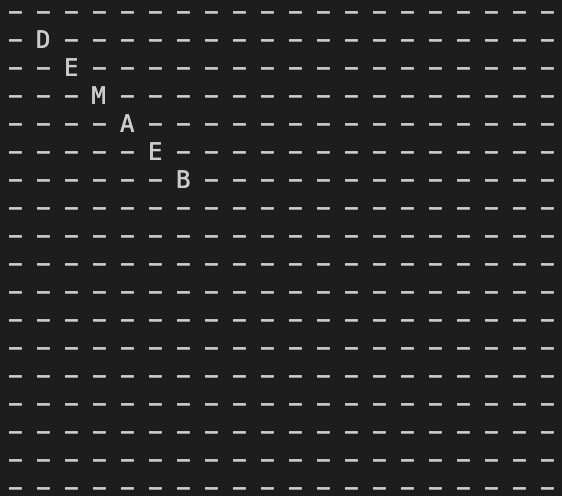
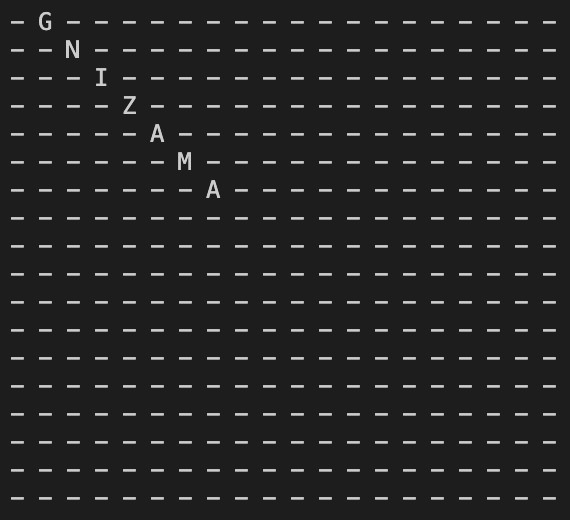
# • Output 5



# • Input 6



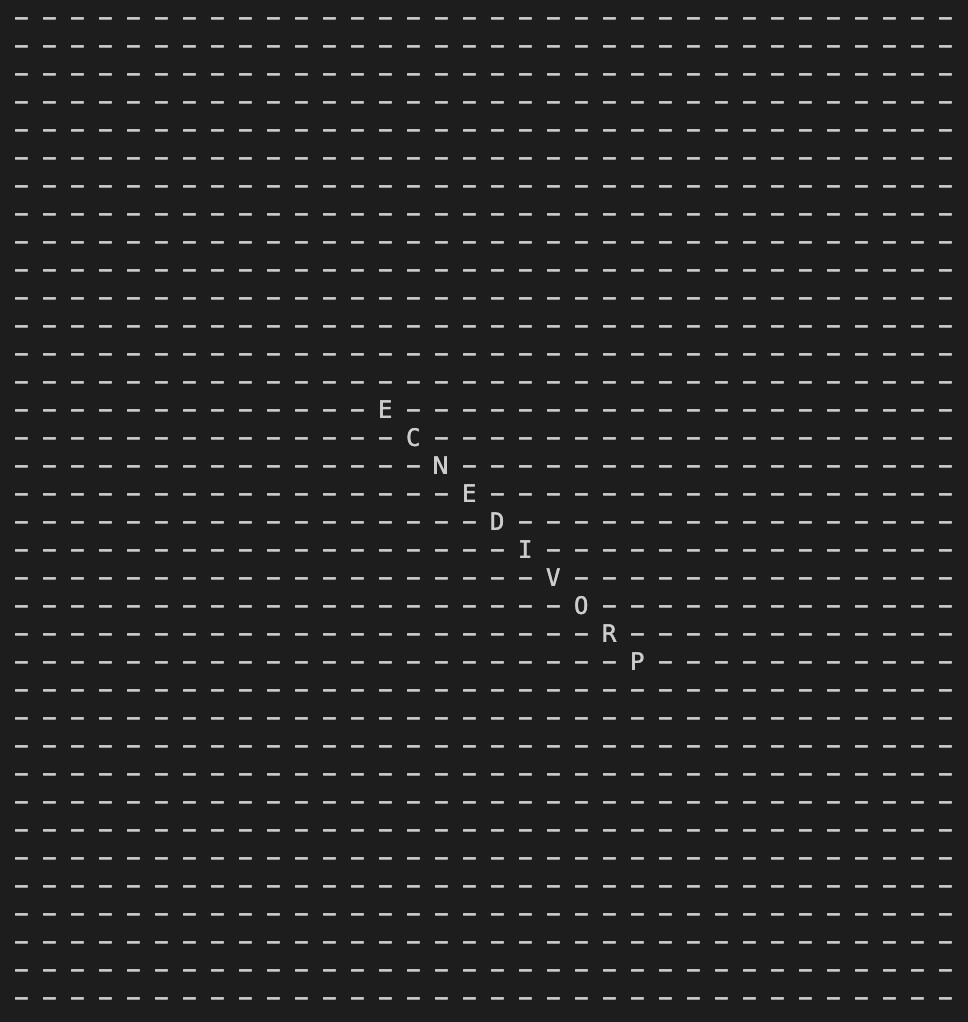
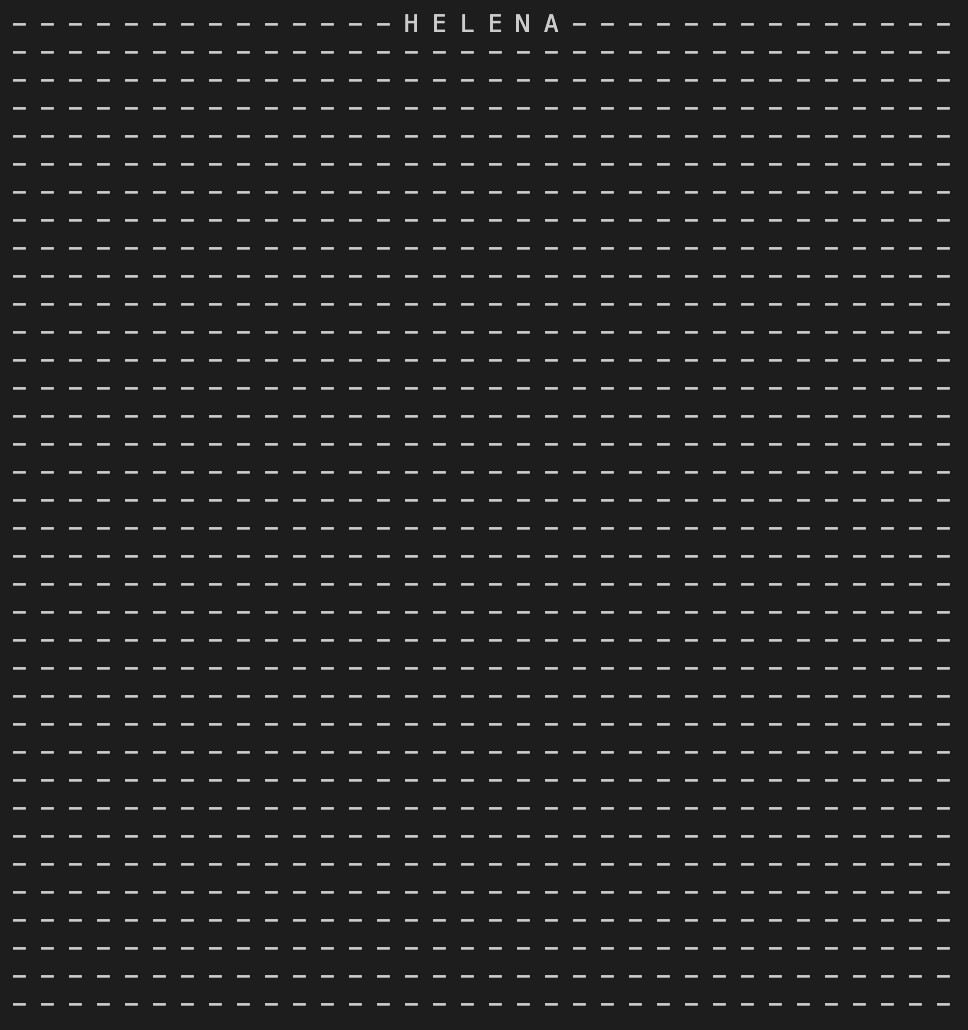
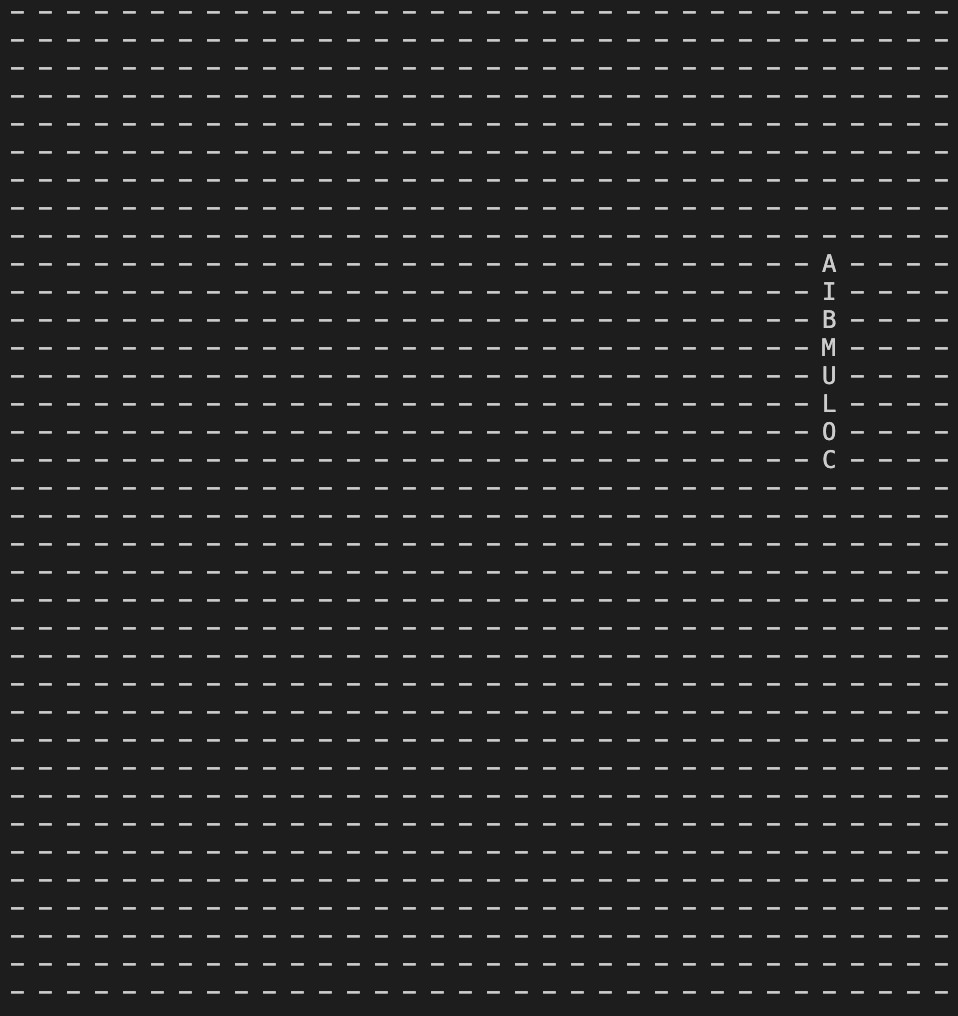
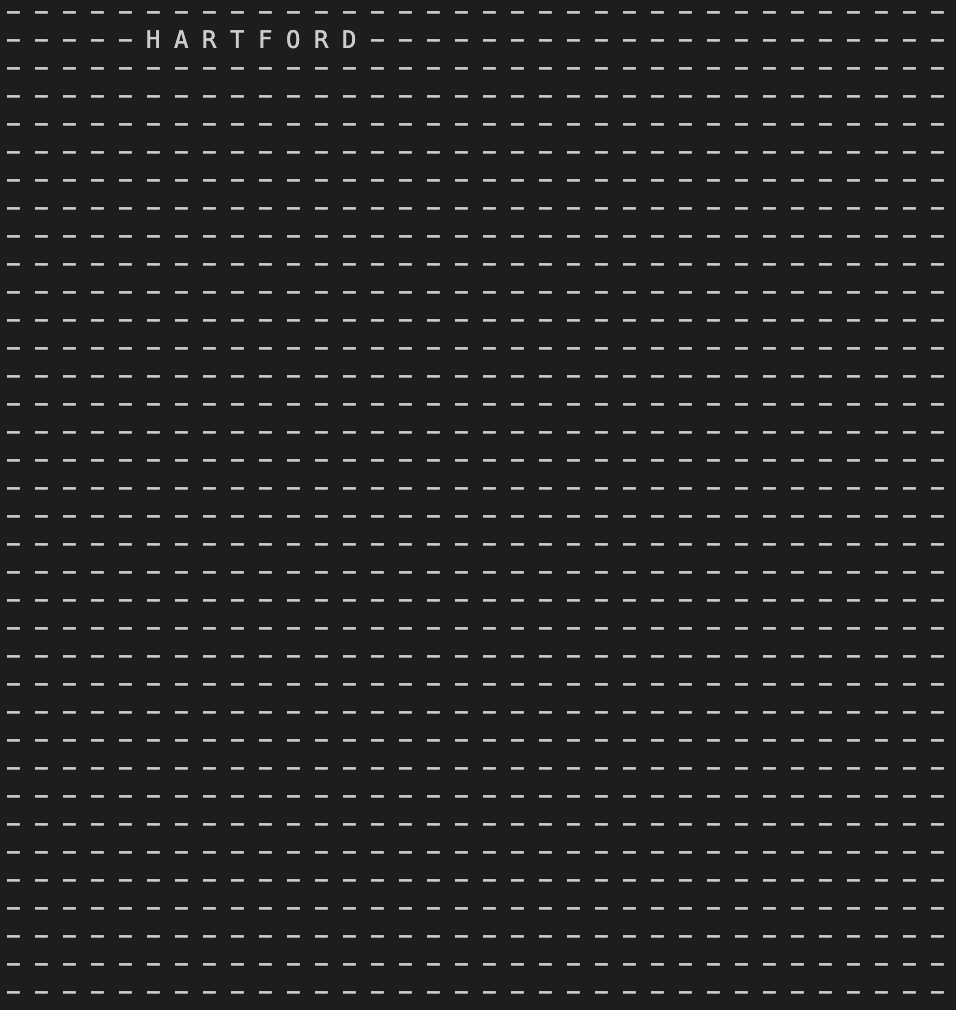
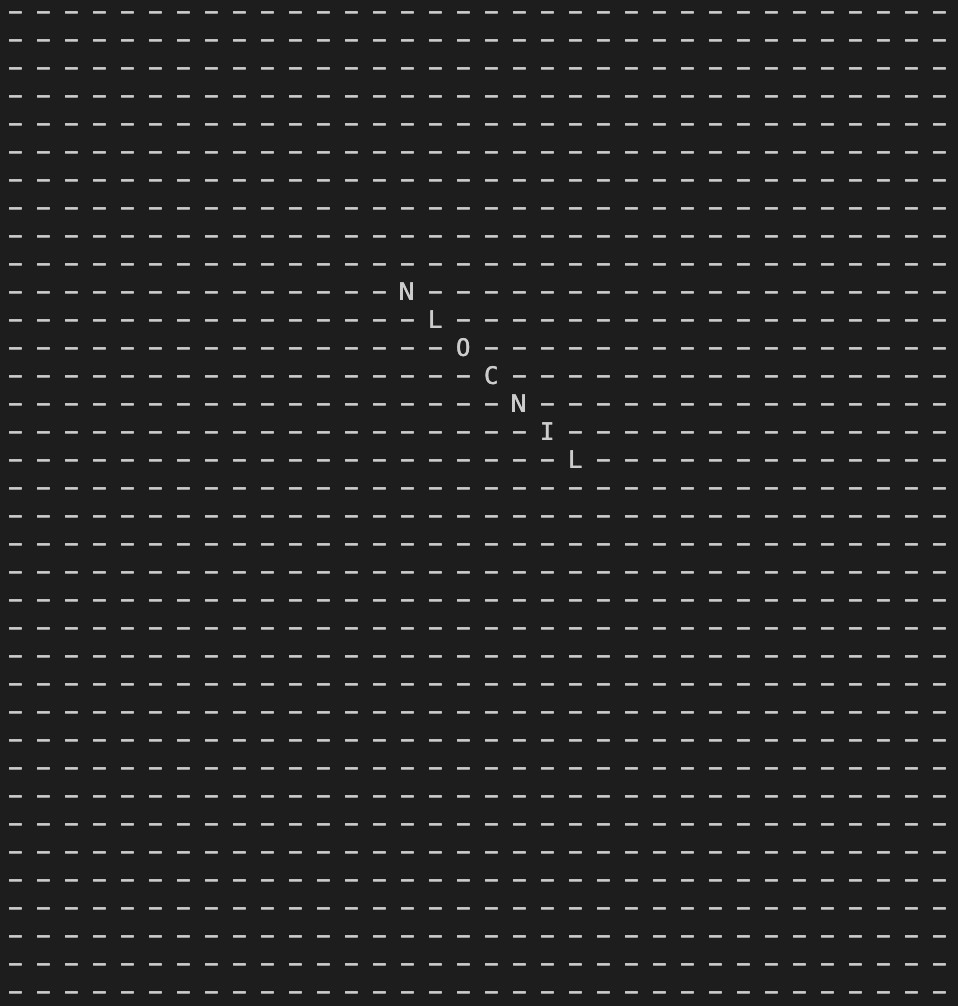
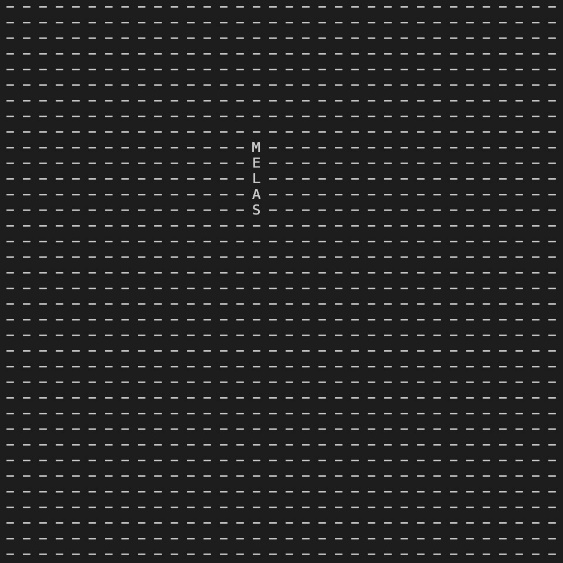
# • Output 6



# 3. *Large* • Input 6



# • Output 6



# *Link Google Drive*

1. **Google Drive** https://drive.google.com/drive/folders/1ssMj0HC2UsAzhBcS2wC6FbmCkH8xaZD-?u%20sp=sharing
2. **Repository GitHub**

<https://github.com/serevinaash/PuzzleGame>

# V. *Checklist*

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi tanpa kesalahan (*no syntax error*). | ✓ |  |
| 2. Program berhasil *running*. | ✓ |  |
| 3. Program berhasil membaca *file* masukan dan menuliskan luaran. | ✓ |  |
| 4. Program berhasil menemukan semua kata di dalam *puzzle*. | ✓ |  |