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

# Konstanta

KAPASITAS = 100

class TekaTeki:

    def \_\_init\_\_(self):

        self.isi = [[''] \* KAPASITAS for \_ in range(KAPASITAS)]

        self.baris = 0

        self.kolom = 0

class DaftarKata:

    def \_\_init\_\_(self):

        self.isi = []

        self.jumlah = 0

def file2data(nama\_berkas, teka\_teki, daftar\_kata):

    jalur\_berkas = f"./test/{nama\_berkas}"

    with open(jalur\_berkas, 'r') as berkas:

        baris = berkas.readlines()

    # Inisialisasi indeks

    i = 0

    # Baca isi teka-teki

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

        baris\_teka\_teki = baris[i].strip().replace(" ", "")

        teka\_teki.isi[i][:len(baris\_teka\_teki)] = baris\_teka\_teki

        i += 1

    teka\_teki.baris = i

    teka\_teki.kolom = len(baris\_teka\_teki) if i > 0 else 0

    # Baca isi daftar kata

    while i < len(baris):

        baris\_kata = baris[i].strip()

        if baris\_kata:

            daftar\_kata.isi.append(baris\_kata)

            daftar\_kata.jumlah += 1

        i += 1

def cetak\_grid(teka\_teki, grid\_ditemukan):

    for baris in range(teka\_teki.baris):

        baris\_teks = ''

        for kolom in range(teka\_teki.kolom):

            if grid\_ditemukan[baris][kolom]:

                baris\_teks += teka\_teki.isi[baris][kolom] + ' '

            else:

                baris\_teks += '- '

        print(baris\_teks)

def cari\_kata(teka\_teki, kata):

    panjang = len(kata)

    grid\_ditemukan = [[False] \* KAPASITAS for \_ in range(KAPASITAS)]

    jumlah\_perbandingan = 0  # Menginisialisasi jumlah perbandingan

    for i in range(teka\_teki.baris):

        for j in range(teka\_teki.kolom):

            arah = {

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

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

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

                "Barat Daya": lambda k: (i + k, j - k),

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

                "Barat Laut": lambda k: (i - k, j - k),

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

                "Timur Laut": lambda k: (i - k, j + k)

            }

            for arah, posisi in arah.items():

                ditemukan = True

                for k in range(panjang):

                    baris, kolom = posisi(k)

                    # Tambah perbandingan

                    jumlah\_perbandingan += 1

                    if not (0 <= baris < teka\_teki.baris and 0 <= kolom < teka\_teki.kolom):

                        ditemukan = False

                        break

                    if teka\_teki.isi[baris][kolom] != kata[k]:

                        ditemukan = False

                        break

                if ditemukan:

                    for k in range(panjang):

                        baris, kolom = posisi(k)

                        grid\_ditemukan[baris][kolom] = True

                    print(f"Kata '{kata}' ditemukan dalam arah '{arah}' mulai dari ({i}, {j}):")

                    cetak\_grid(teka\_teki, grid\_ditemukan)

                    return jumlah\_perbandingan  # Kembalikan jumlah perbandingan

    print(f"Kata '{kata}' tidak ditemukan!")

    return jumlah\_perbandingan

def main():

    teka\_teki = TekaTeki()

    daftar\_kata = DaftarKata()

    # Baca berkas

    nama\_berkas = input("\nMasukkan nama berkas uji Anda: ")

    print("\n")

    file2data(nama\_berkas, teka\_teki, daftar\_kata)

    # Metode Brute Force

    total\_jumlah\_perbandingan = 0  # Total perbandingan

    awal = time.time()

    for i in range(daftar\_kata.jumlah):

        # Perbarui total\_jumlah\_perbandingan dengan jumlah perbandingan yang dikembalikan oleh cari\_kata

        total\_jumlah\_perbandingan += cari\_kata(teka\_teki, daftar\_kata.isi[i])

        print("\n")

    akhir = time.time()

    durasi = (akhir - awal) \* 1e6  # Konversi ke mikrodetik

    print(f"Waktu yang dihabiskan: {durasi:.0f} mikrodetik")

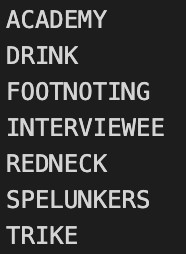
    print(f"Total perbandingan: {total\_jumlah\_perbandingan} huruf\n\n")

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

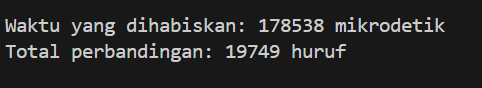
    main()

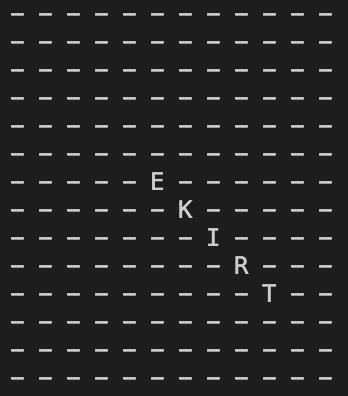
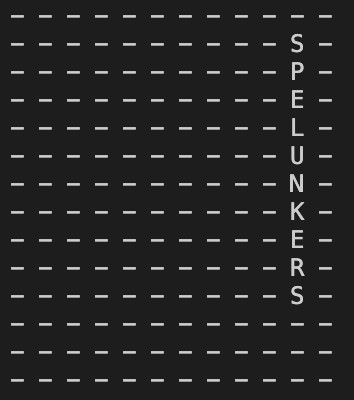
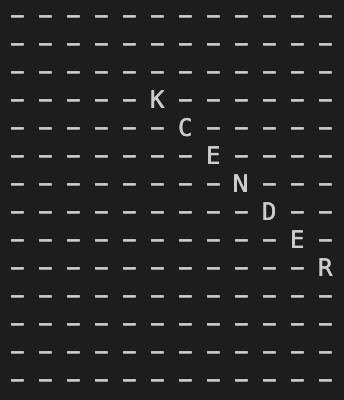
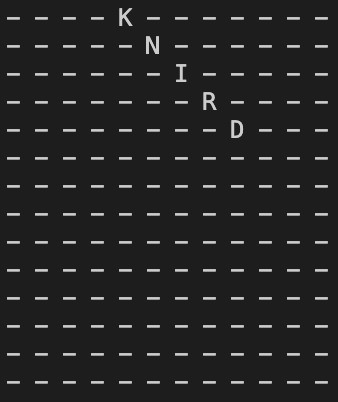
# *3. Screenshot* dari *Input* dan *Output* dari Program

# 1. *Small* • Input 1

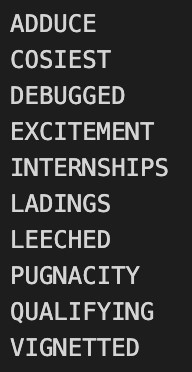


# • Output 1



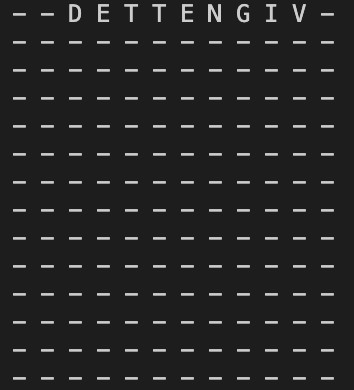
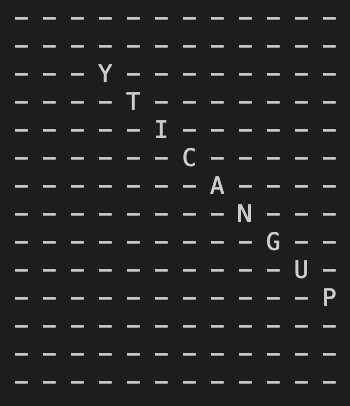
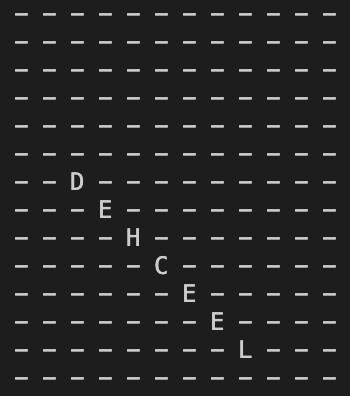
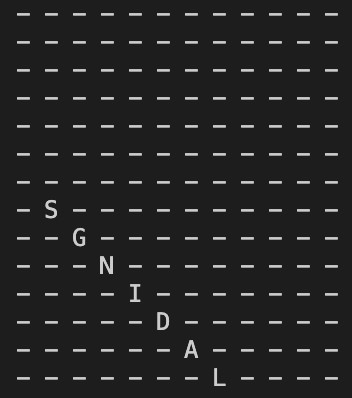
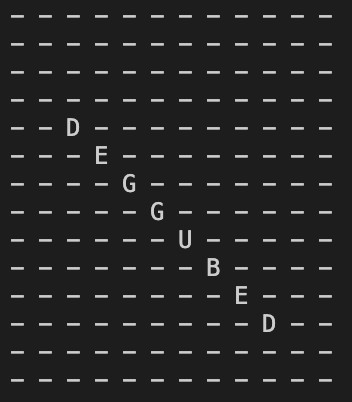
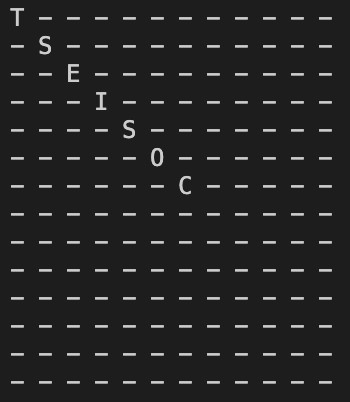
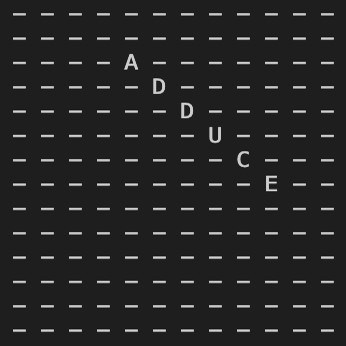


# • Input 2

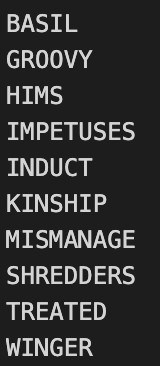


# •Ouput 2

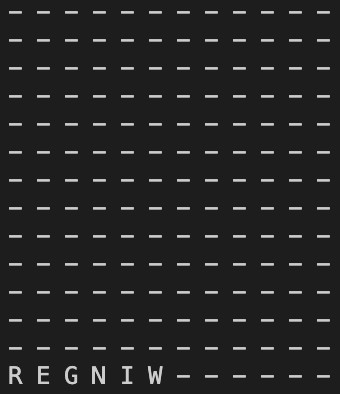
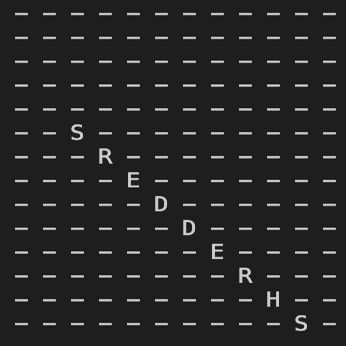
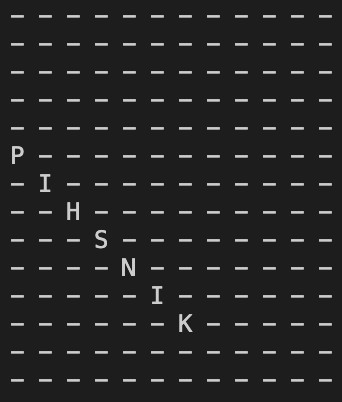
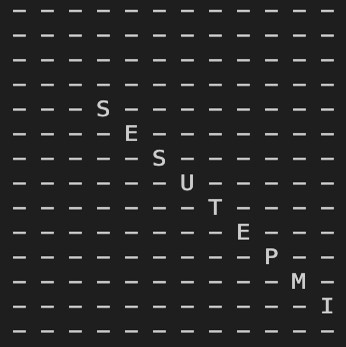
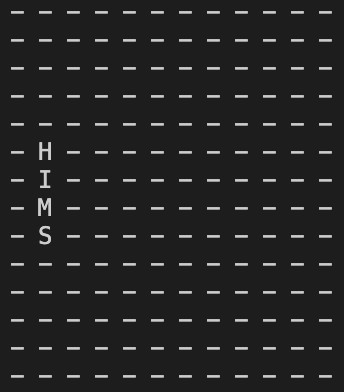
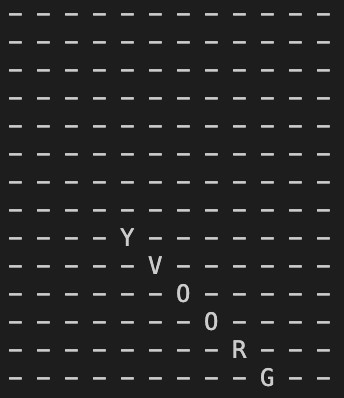
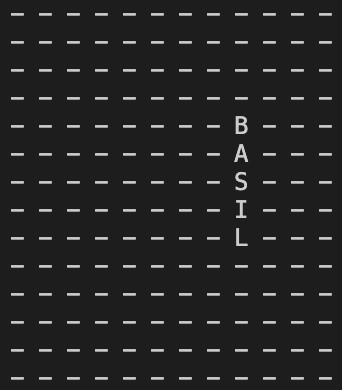




# • Input 3

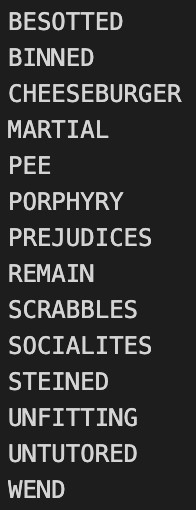


# • Output 3

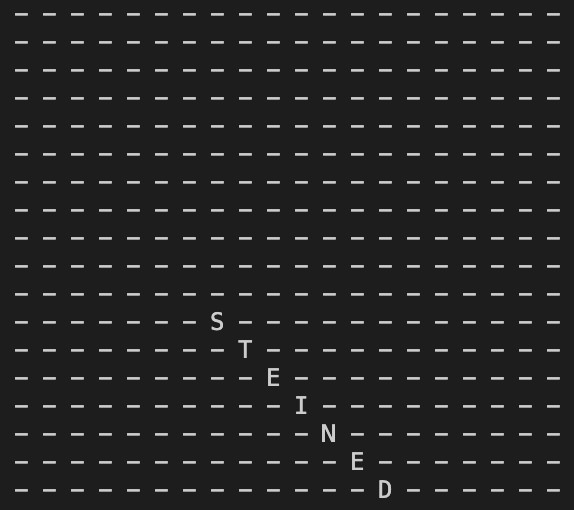
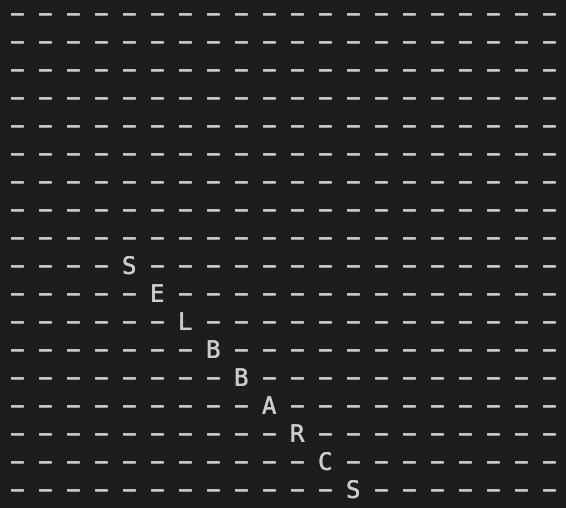
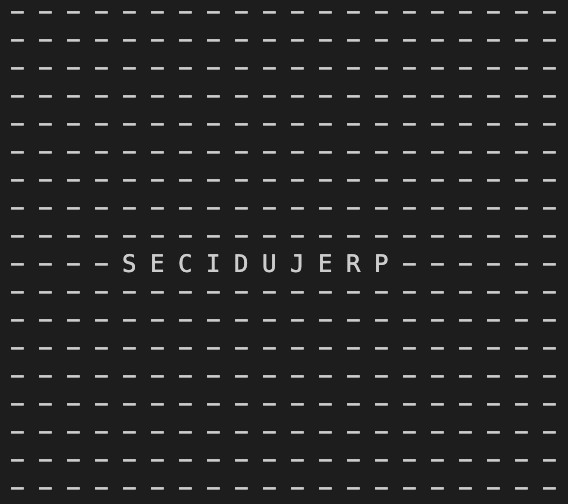
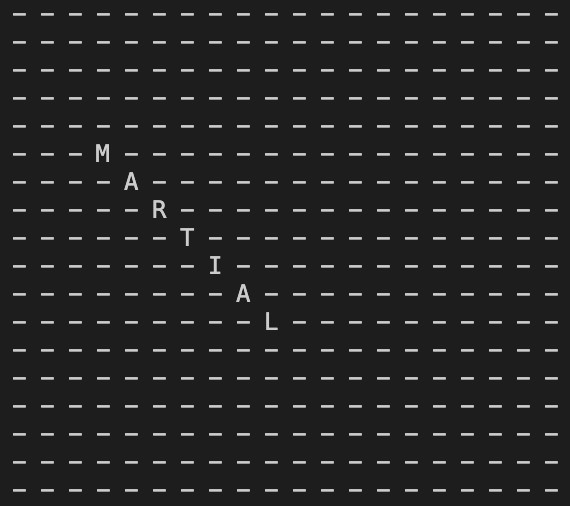
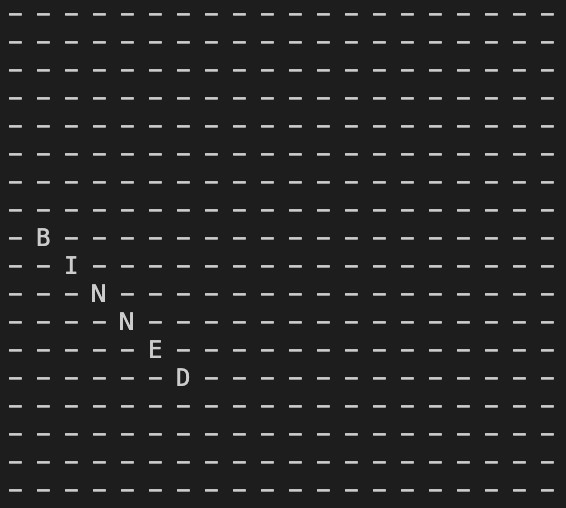




# 2. *Medium1* • Input 4



# • Output 4

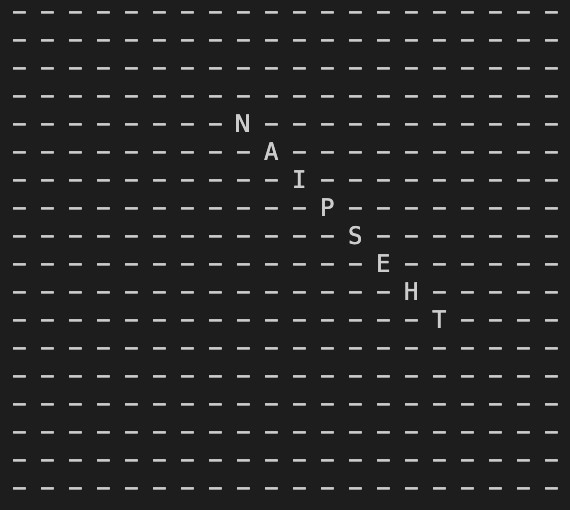
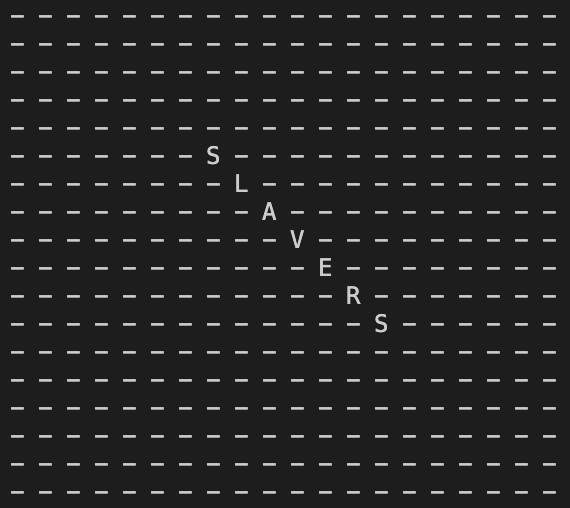
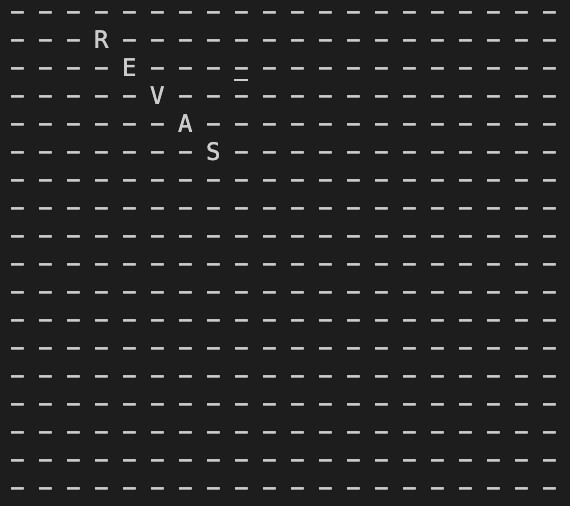
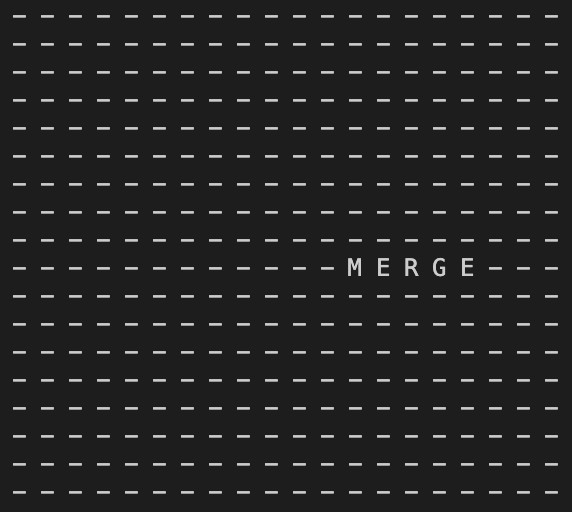
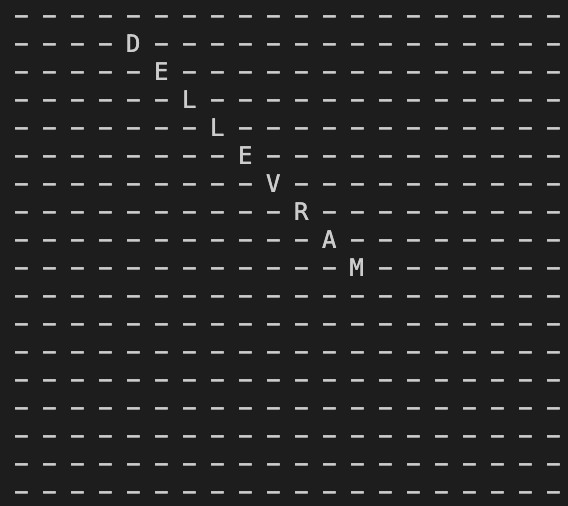
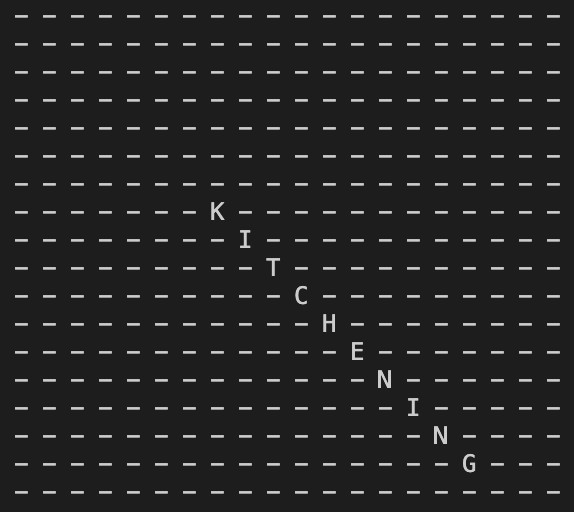
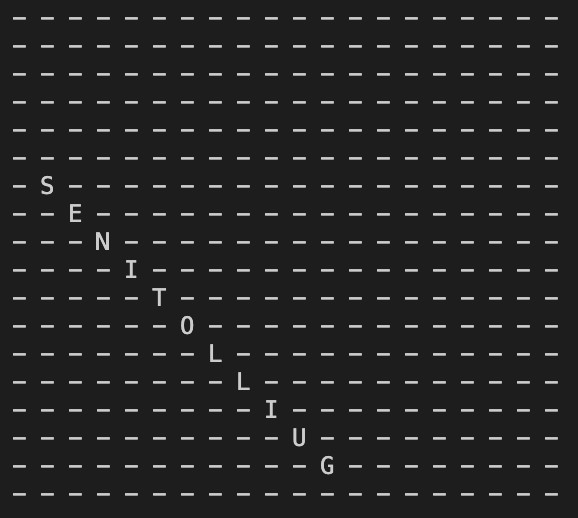
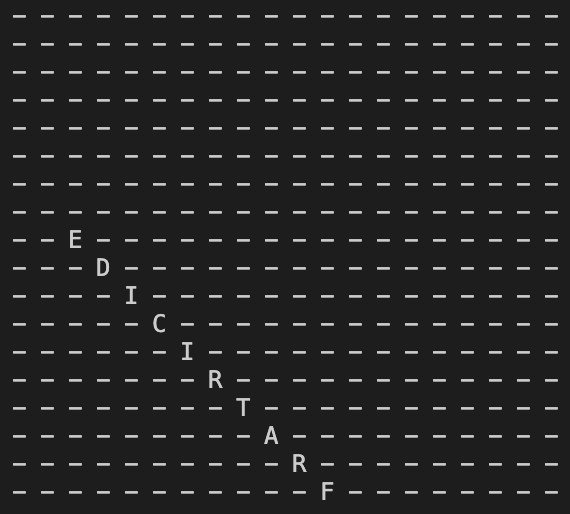
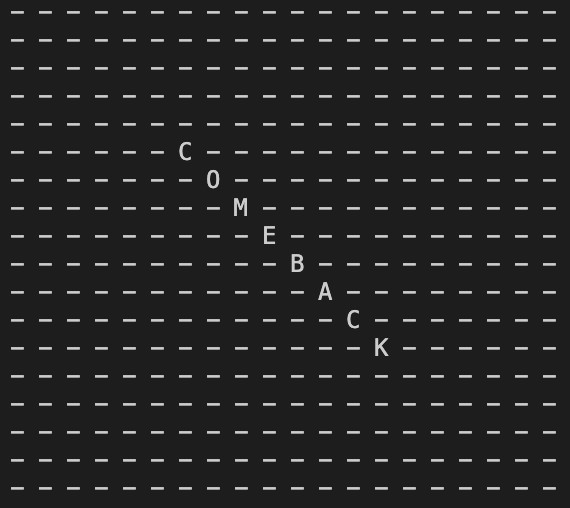
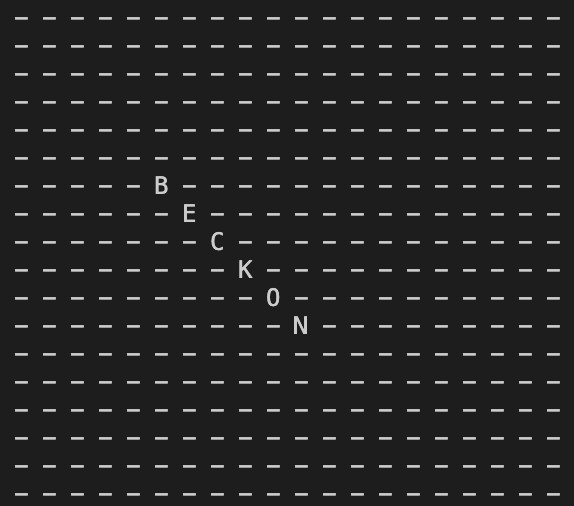


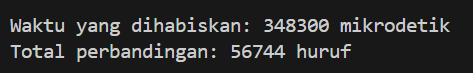


# • Input 5 (medium 2)

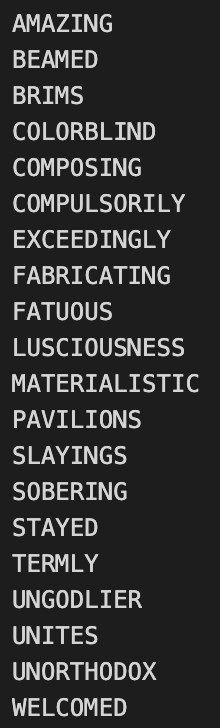


# • Output 5

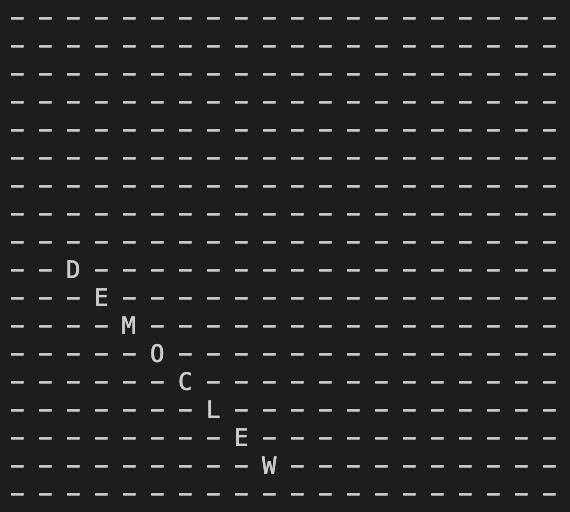
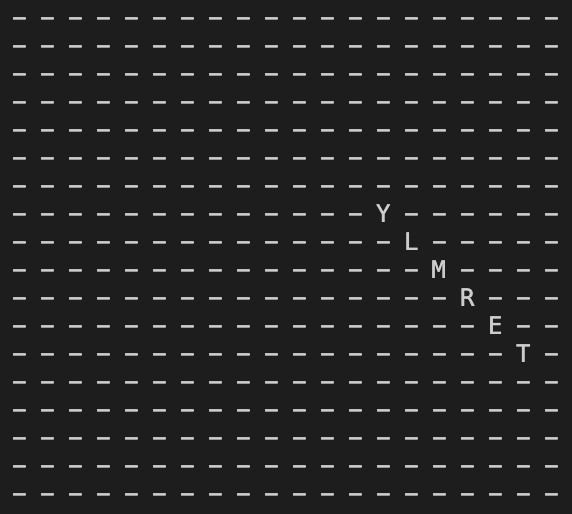
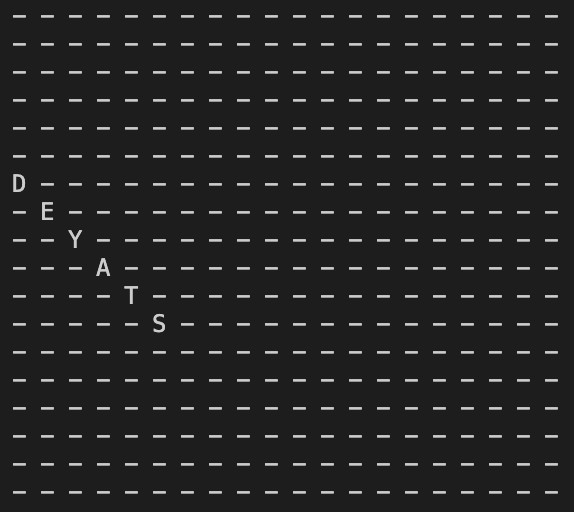
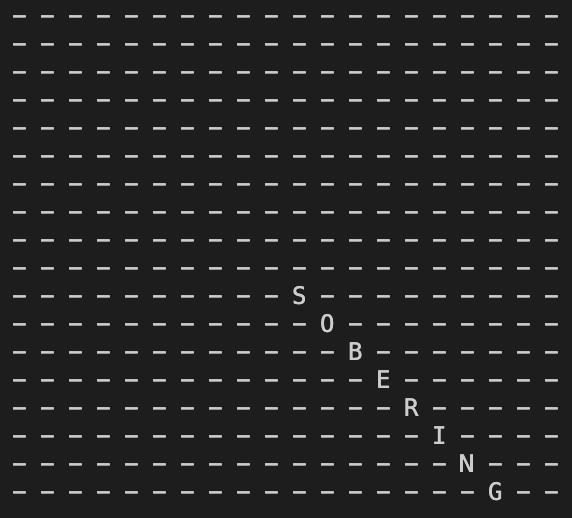
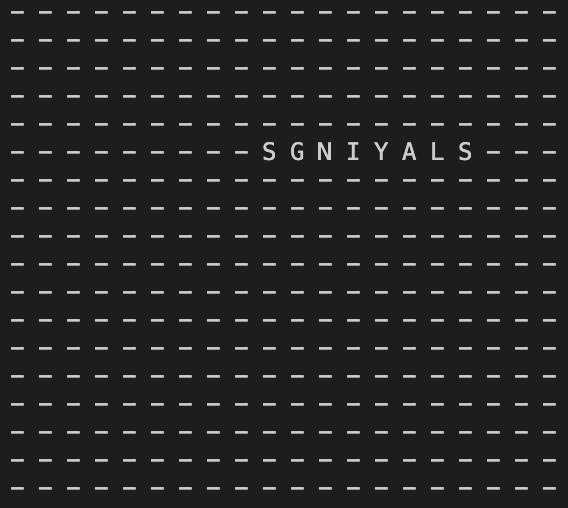
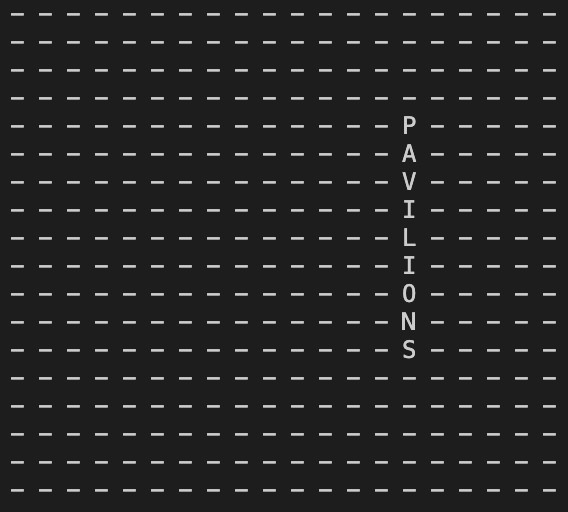
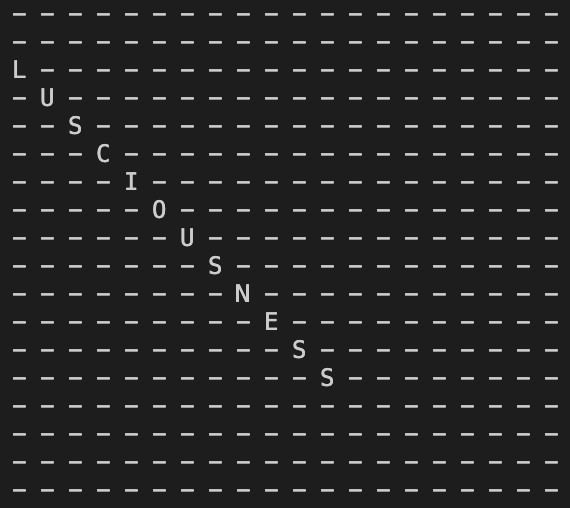
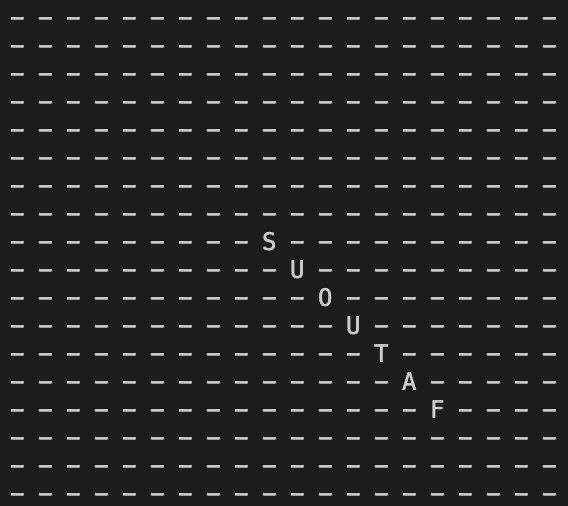
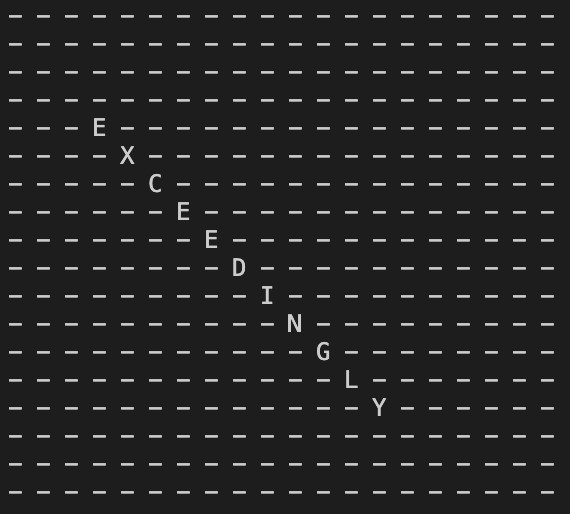
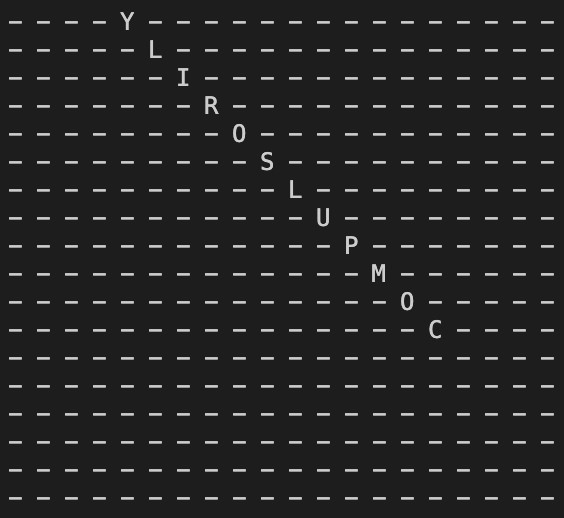
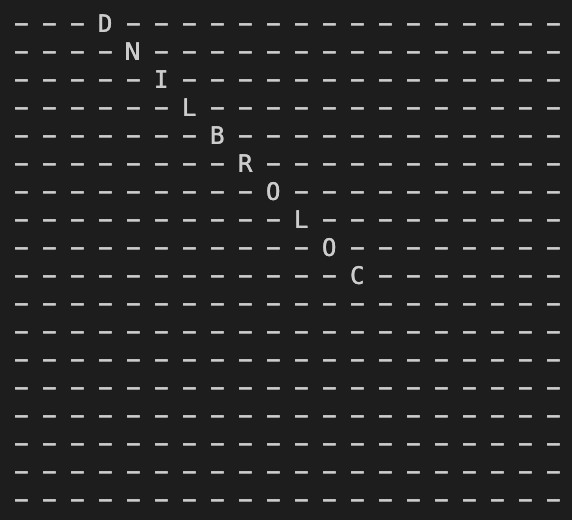
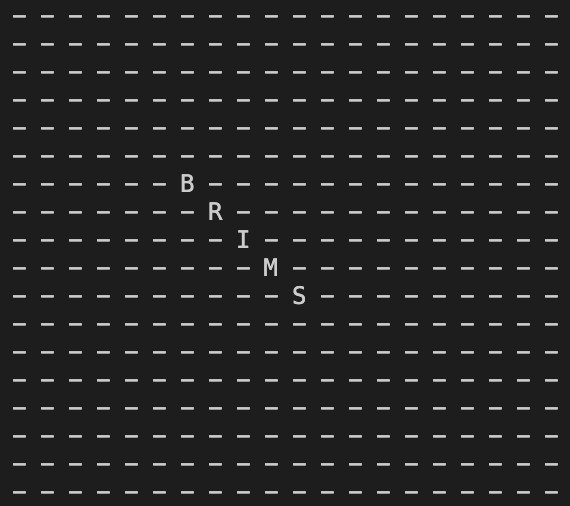
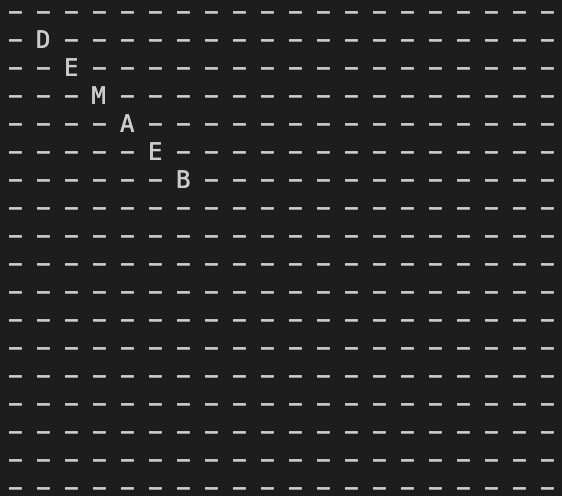
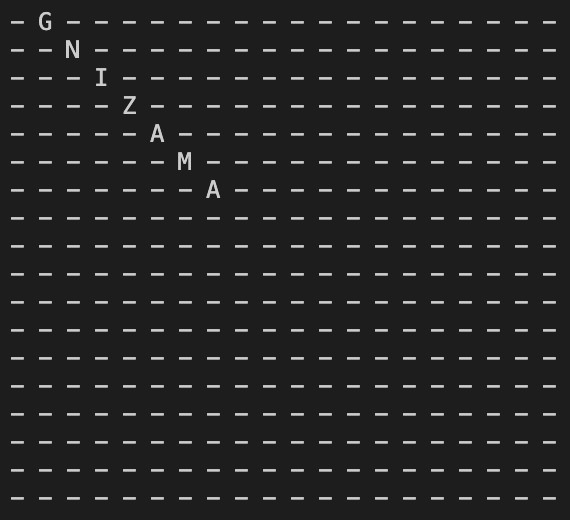


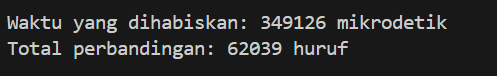


# • Input 6 (Medium 3)



# • Output 6

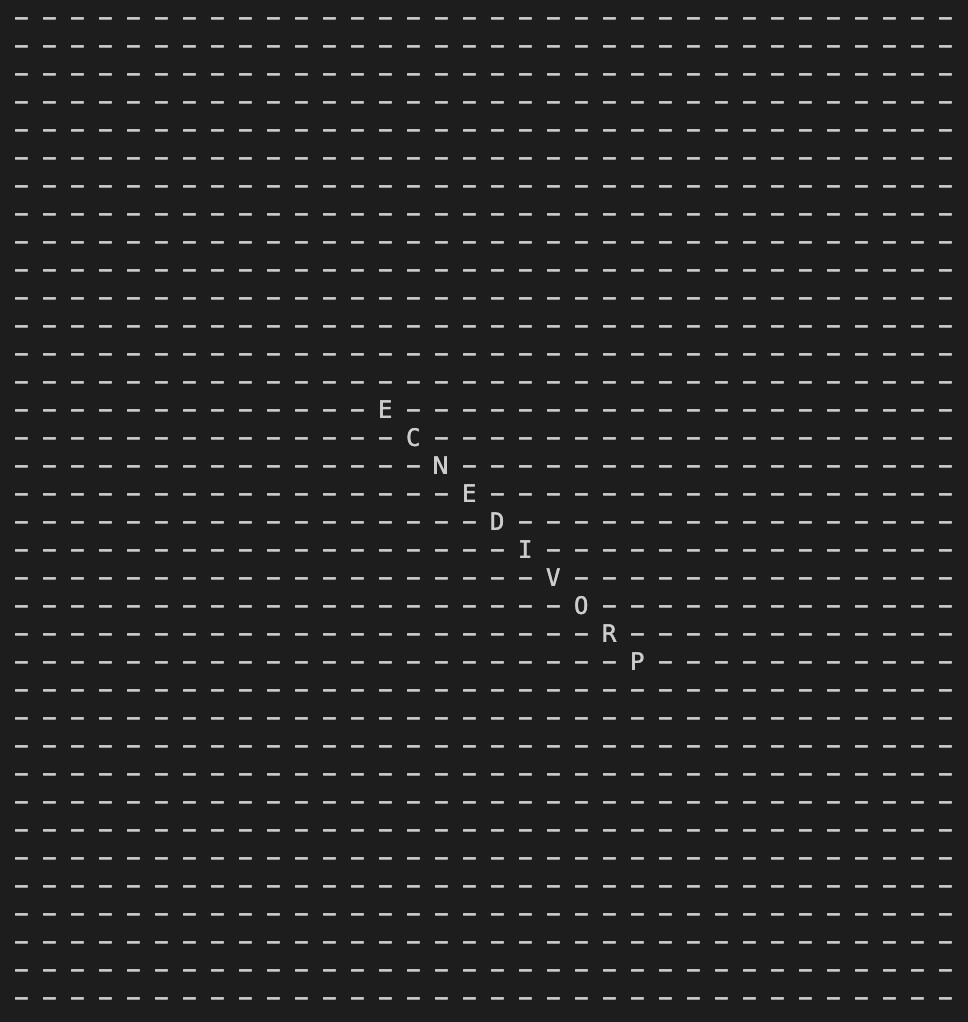
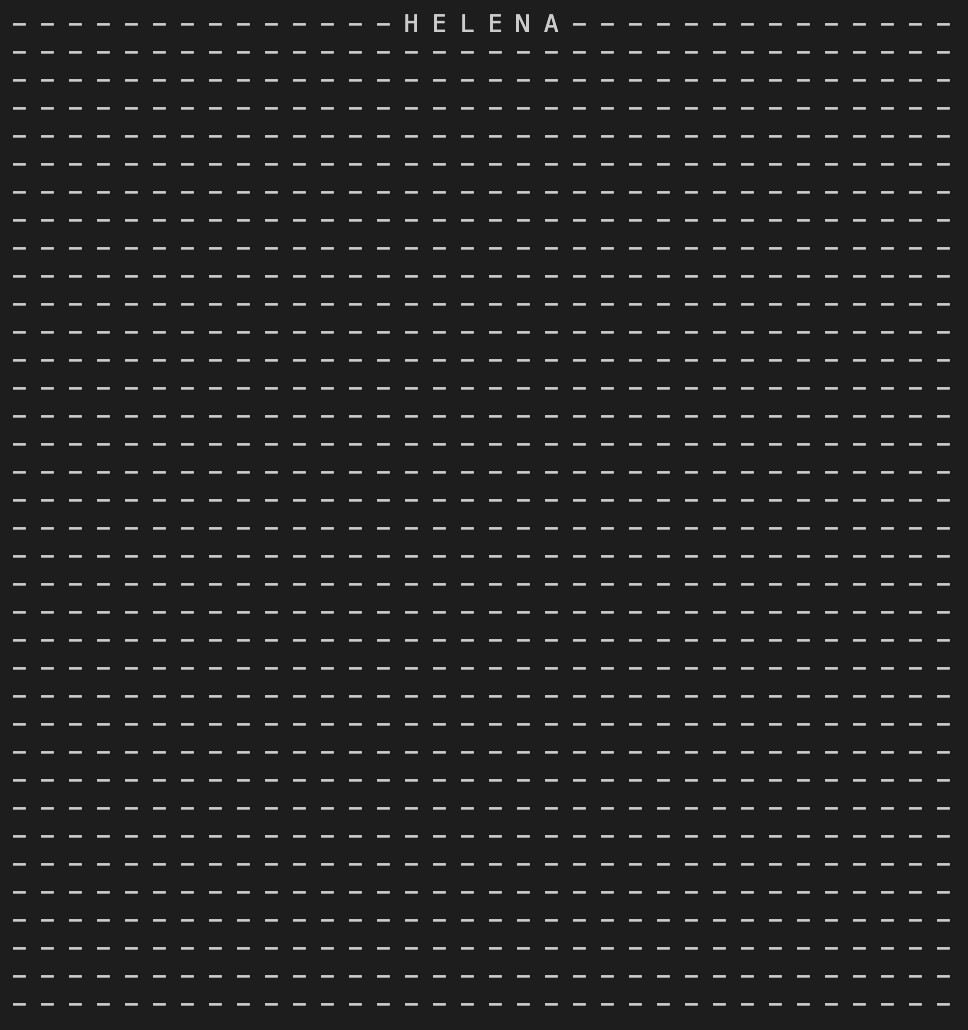
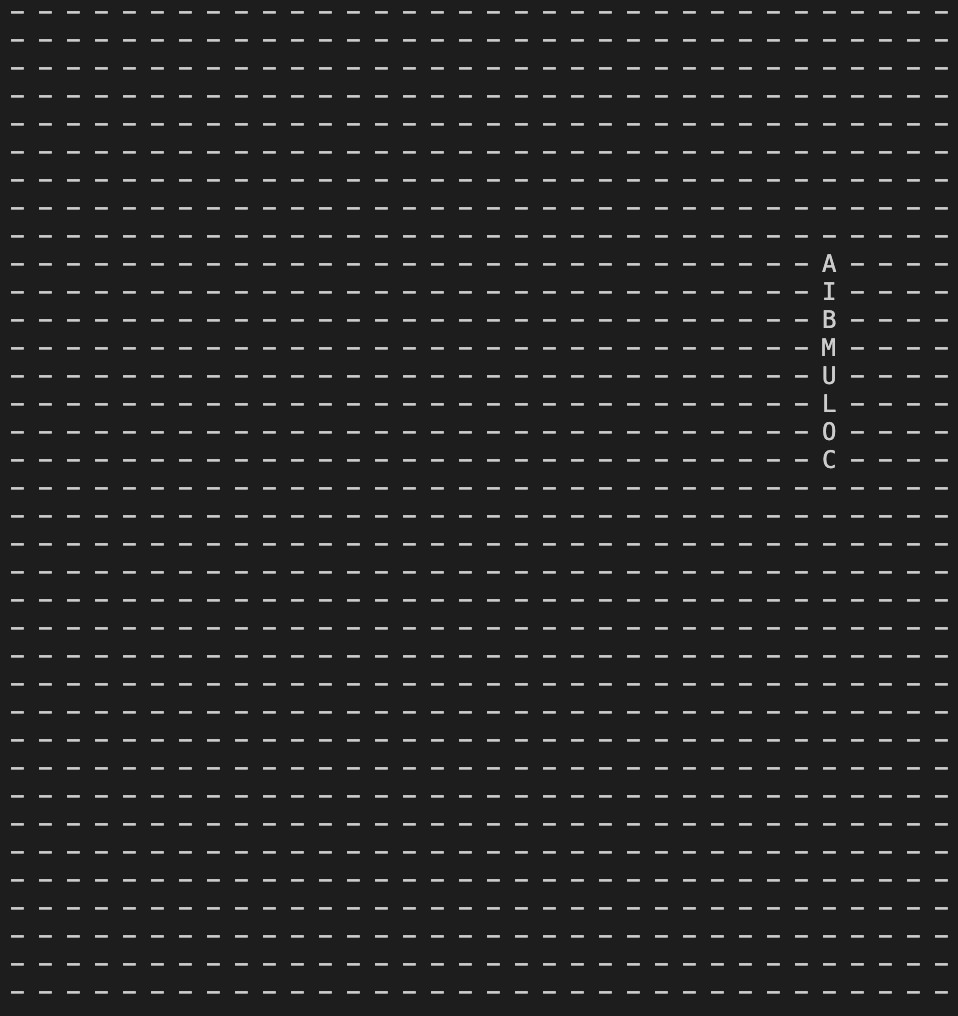
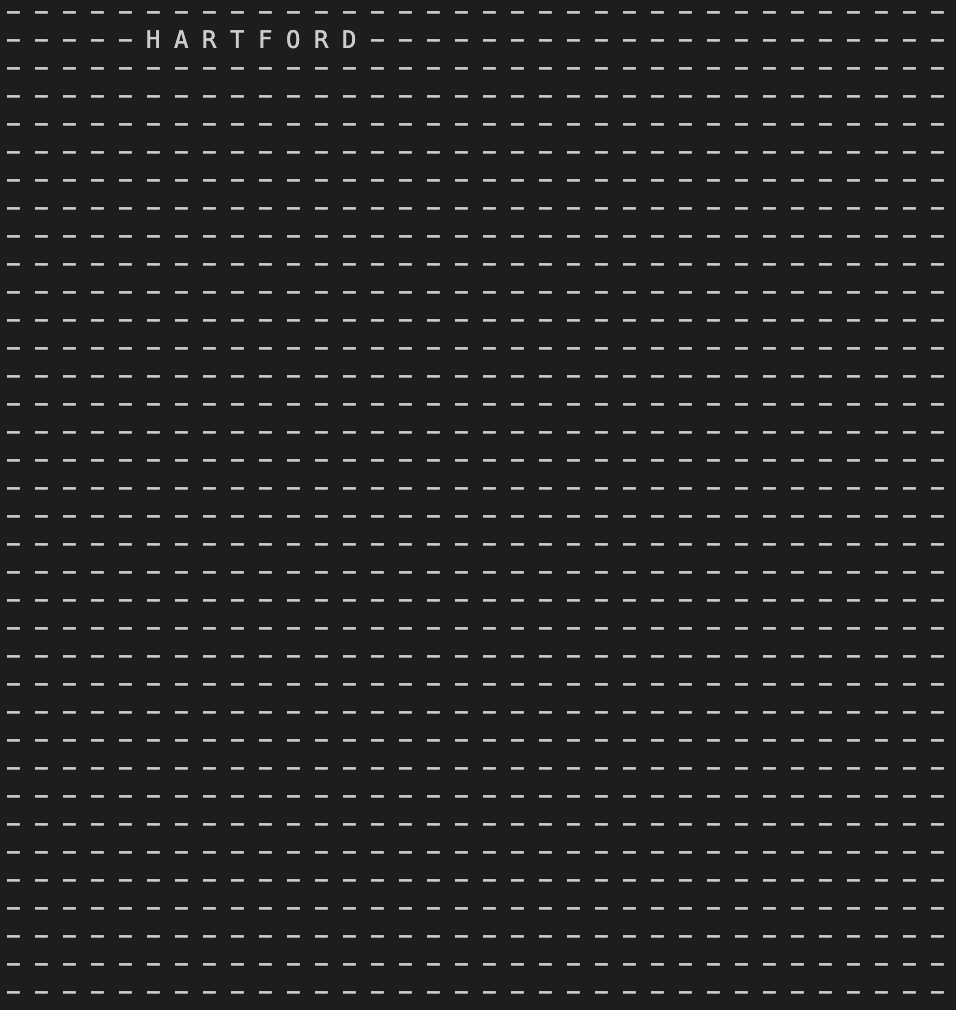
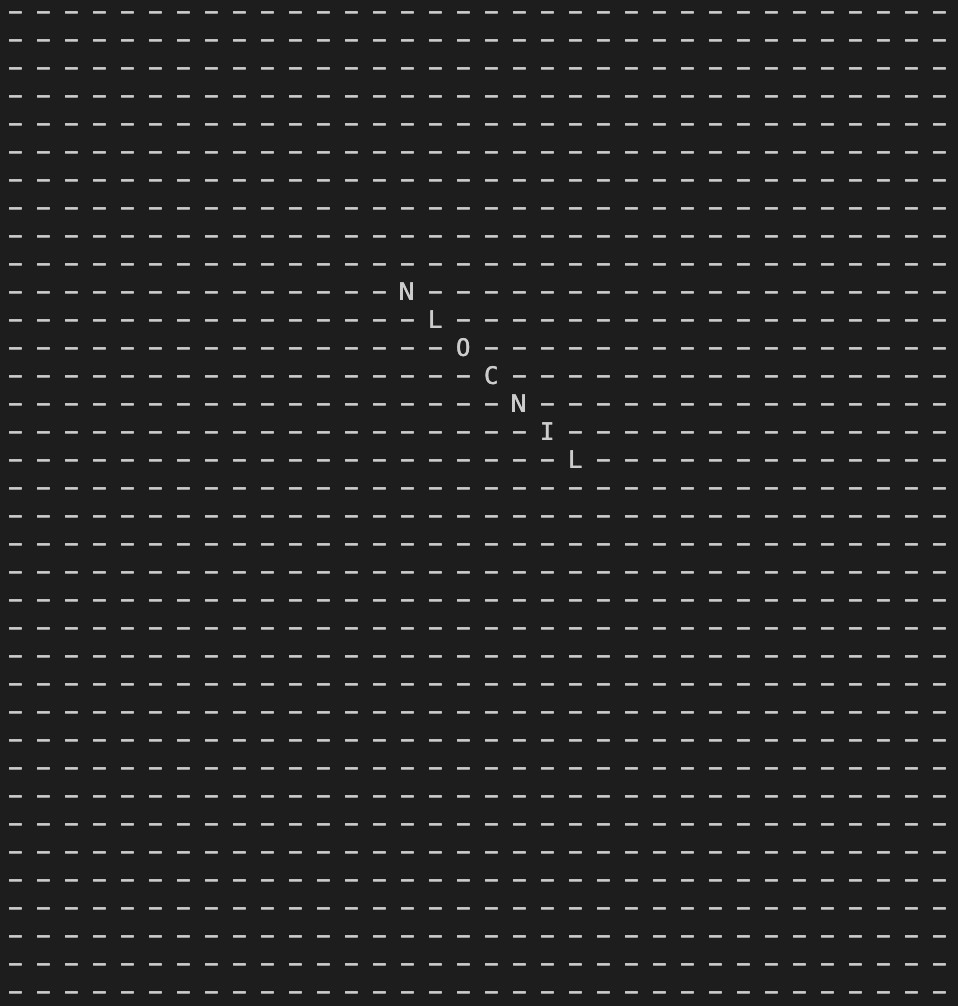
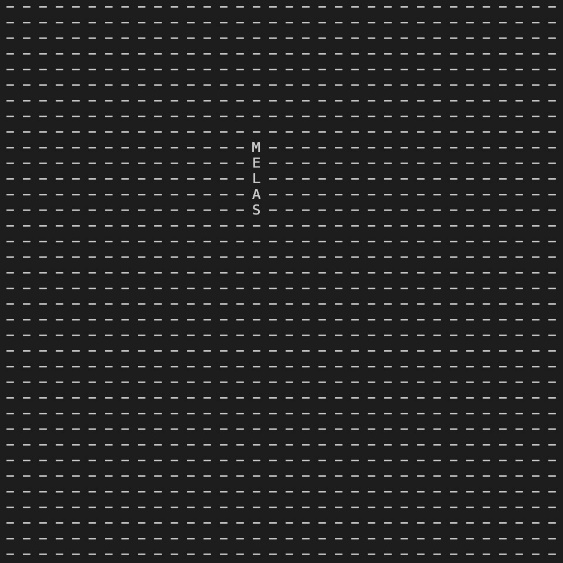


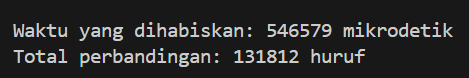


# 3. *Large* • Input 6



# • 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*. | ✓ |  |