Laporan Tugas Kecil 1 Strategi Algoritma IF2211 Semester II Tahun 2021/2022

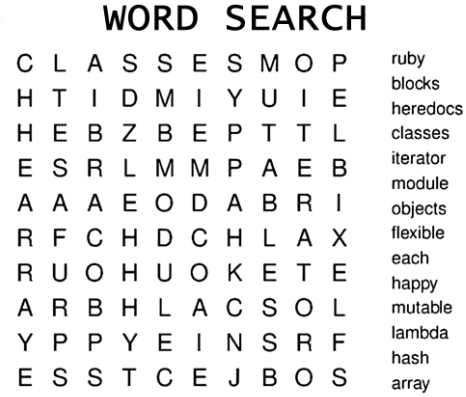
Penyelesaian *Word Search Puzzle* dengan Algoritma *Brute Force*

Nayotama Pradipta - 13520089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

I. Pendahuluan

A. Word Search Puzzle

Word Search Puzzle merupakan permainan yang melibatkan sebuah kumpulan huruf acak dan beberapa kata yang harus dicari. Kumpulan huruf acak ini disusun pada papan/kotak/matriks yang berbneutk segi-empat atau dapat disebut juga sebagai matriks huruf. Kata-kata yang harus dicari dapat ditemukan dalam delapan arah, yaitu vertikal ke atas, vertikal ke bawah, horizontal ke kanan, horizontal ke kiri, diagonal ke kanan atas, diagonal ke kanan bawah, diagonal ke kiri atas, dan diagonal ke kiri bawah. Tujuan akhir dari permainan ini adalah menemukan seluruh kata-kata yang harus dicari pada matriks huruf.



Gambar 1.1 Illustrasi Word Search Puzzle

B. Algoritma Brute Force

Algoritma Brute Force merupakan algoritma pada suatu program yang memiliki karakteristik *straightforward*, sederhana, dan *naïve*. Algoritma ini dapat diimplementasikan dengan sangat mudah karena tidak ada unsur kreativitas dan tidak memandang efisiensi. Algoritma ini secara garis besar akan selalu memakan waktu lebih banyak dan boros memori. Brute Force seringkali digunakan sebagai basis pembanding dengan algoritma lain yang lebih efisien.

II. Langkah-langkah Algoritma Brute Force

Algoritma brute force pada *word search puzzle* adalah dengan cara mencocokkan huruf pada kata yang dicari dengan huruf yang ada di matriks. Agar perbandingan dapat dilakukan, data yang ada di text file harus diubah terlebih dahulu menjadi matriks/2D array dan 1D array. Perbandingan ini dilakukan secara delapan arah jika huruf pertama pada matriks sama dengan huruf pertama kata yang akan dicari. Program akan mengecek huruf kedua pada semua arah terlebih dahulu sebelum lanjut mencari mencari huruf pertama yang sama. Jika huruf kedua kata sama dengan huruf kedua kata yang dicari, maka selanjutnya akan dicek huruf ketiga, keempat, dst. pada arah yang sama. Pencocokan dilakukan hingga salah satu dari dua kondisi berikut terjadi:

1. Huruf tidak cocok dan panjang pencarian lebih kecil daripada panjang kata
2. Huruf selalu cocok dan jumlah huruf sama dengan panjang kata yang dicari

Jika kondisi pertama terjadi, maka program akan melanjutkan pencarian pada huruf selanjutnya pada matriks (mengular). Jika kondisi kedua terjadi, maka program sukses mencari kata, letak kata yang ditemukan akan ditampilkan dan program akan melanjutkan mencari kata selanjutnya. Proses ini akan diulang hingga seluruh kata ditemukan. Program juga memiliki fitur berupa menghitung waktu eksekusi beserta jumlah perbandingan yang dilakukan.

III. Source Program

Program yang saya buat menggunakan bahasa C++ dan dibagi menjadi tiga file sesuai dengan konsep modularitas. Ketiga file ini bernama function.hpp, function.cpp, dan main.cpp. Berikut adalah source program:

File function.hpp

```
extern int xAxis[];
extern int yAxis[];

bool isValid(std::string userinput);
void getSpec(std::string userinput, int *matrixCol, int *matrixRow, int
*words);
bool searchMatrix(char *matrix, std::string word, int row, int col, int
matrixRow, int matrixCol, int *numOfCheck, int *x, int *y);
void findPattern(char *matrix, std::string word, int matrixRow, int
matrixCol, int *numOfCheck, int *x, int *y);
void printSolution(int matrixRow, int matrixCol, int row, int col, int x,
int y);
```

File function.cpp

```cpp
#include <iostream>
#include "function.hpp"

int xAxis[] = { -1, -1, -1,  0, 0,  1, 1, 1 };
int yAxis[] = { -1,  0,  1, -1, 1, -1, 0, 1 };

bool isValid(std::string userinput){
    return (userinput == "small.txt" || userinput == "medium1.txt" ||
userinput == "medium2.txt" ||userinput == "medium3.txt" || userinput ==
"medium4.txt" || userinput == "medium5.txt" || userinput == "hard1.txt" ||
userinput == "hard2.txt");
}
void getSpec(std::string userinput, int *matrixCol, int *matrixRow, int
*words){
    // Get array and matrix size depending on the user input
    if (userinput == "small.txt"){ // P.S. Small Crossword Puzzle saya ambil
contoh dari spesifikasi tucil karena tidak ada di web pada pdf file
        *matrixCol = 8;
        *matrixRow = 7;
        *words = 8;
    }
    else if (userinput == "medium1.txt" || userinput == "medium2.txt" ||
userinput == "medium3.txt" || userinput == "medium4.txt" || userinput ==
"medium5.txt"){ // Medium berukuran 22x20
        *matrixCol = 15;
        *matrixRow = 15;
        *words = 15;
    }
    else { // Large berukuran 34x32
        *matrixCol = 22;
        *matrixRow = 20;
        *words = 20;
    }
}


bool searchMatrix(char *matrix, std::string word, int row, int col, int
matrixRow, int matrixCol, int *numOfCheck, int *x, int *y){
```

```
    // Core/Main function to search and find the words inside the matrix
    // If first letter is not equal -> break
    if (*(matrix+row*matrixCol+col) != word[0]){
        return false;
    }
    int path;
    for (path = 0; path < 8; path++){
        int a;
        int ra = row + xAxis[path];
        int ca = col + yAxis[path];
        for (a = 1; a < word.length(); a++){
            if (ra >= matrixRow || ca >= matrixCol || ra < 0 || ca < 0){
                *numOfCheck += 1;
                break;
            }
            if (*(matrix+ra*matrixCol+ca) != word[a]){
                *numOfCheck += 1;
                break;
            }
            ra += xAxis[path];
            ca += yAxis[path];
        }
        if (a == word.length()){
            *x = xAxis[path];
            *y = yAxis[path];
            return true;
        }
    }
    return false;
}
void printSolution(std::string word, int matrixRow, int matrixCol, int row,
int col, int x, int y){
    // This function is used to display the solution in a matrix form (Assume
that the word has already been found)
    char matrixSolution[matrixRow][matrixCol];
    for (int i = 0; i < matrixRow; i++){
        for (int j = 0; j < matrixCol; j++){
            matrixSolution[i][j] = '-';
```

```
        }
    }
    matrixSolution[row][col] = word[0];
    for (int i = 1; i < word.length(); i++){
        matrixSolution[row+x*i][col+y*i] = word[i];
    }
    for (int i = 0; i < matrixRow; i++){
        for (int j = 0; j < matrixCol; j++){
            std::cout << matrixSolution[i][j] << " ";
        }
        std::cout << "\n";
    }
    std::cout << "\n";
}
void findPattern(char *matrix, std::string word, int matrixRow, int matrixCol,
int *numOfCheck, int *x, int *y){
    for (int row = 0; row < matrixRow; row++){
        for (int col = 0; col < matrixCol; col++){
            if (searchMatrix(matrix, word, row, col, matrixRow, matrixCol,
numOfCheck, x, y)){
                printSolution(word, matrixRow, matrixCol, row, col, *x, *y);
            }
            *numOfCheck += 1;
        }
        *numOfCheck += 1;
    }
}
```

File main.cpp:

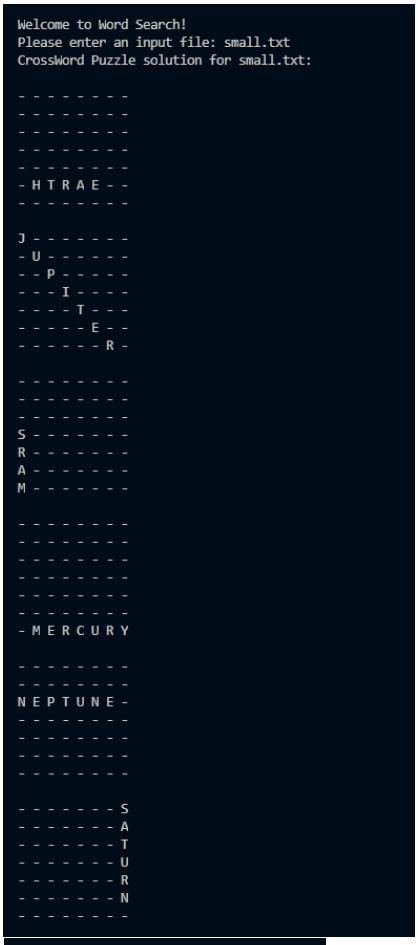
```cpp
#include <iostream>
#include "function.hpp"
#include <bits/stdc++.h>


using namespace std;
int main(){
    clock_t start, end;
    string userinput;
    string output;
    char alfabet = 0;
    bool allfound = false;
    int matrixCol = 0;
    int matrixRow = 0;
    int line = 0;
    int i = 0;
    int j = 0;
    int k = -1;
    int words;
    int numOfCheck = 0;
    int checktest = 0;
    int x = 0;
    int y = 0;
    cout << "Welcome to Word Search!\n";
    cout << "Please enter an input file: ";
    cin >> userinput;

    // Invalid input
    while (!isValid(userinput)){
        cout << "Invalid file name! Please input the correct file: ";
        cin >> userinput;
    }

    getSpec(userinput, &matrixCol, &matrixRow, &words);
    // Create 2D and 1D Array to store text file into array
    char crossWordMatrix[matrixRow][matrixCol];
    string wordsToSearch[words];
    // Read File
```

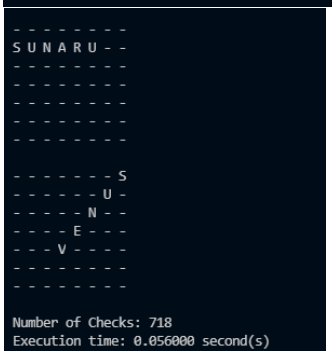
```
FILE* inputFile = fopen(userinput.c_str(), "r");
// Read crossword and store inside matrix/2D array
while (line != matrixRow){
    alfabet = getc(inputFile);
    if (alfabet != '\n'){
        crossWordMatrix[i][j] = alfabet;
    }
    if (alfabet == '\n'){
        line++;
        i++;
        j=0;
    } else {
        if (alfabet != ' '){
            j++;
        }
    }
}
// Read words and store inside 1D array
while (!feof(inputFile)){
    alfabet = getc(inputFile);
    if (alfabet != '\n'){
        wordsToSearch[k] += alfabet;
        if (alfabet == ' '){
            wordsToSearch[k].pop_back();
        }
    } else {
        k++;
    }
}
// Erase last character of the last word
wordsToSearch[k].pop_back();
// Close file
fclose(inputFile);
// Algoritma Brute Force
cout << "CrossWord Puzzle solution for " << userinput << ": \n\n";
start = clock();
for (int m = 0; m < words; m++){
```

```
        findPattern((char *)crossWordMatrix, wordsToSearch[m], matrixRow,
matrixCol, &numOfCheck, &x, &y);
    }
    // Testing Display Array of String
    end = clock();
    double time_taken = double(end - start)/ double(CLOCKS_PER_SEC);
    cout << "Number of Checks: " << numOfCheck << "\n";
    cout << "Execution time: " << fixed << time_taken << setprecision(5)
<< " second(s)";
    cout << "\n";

}
```

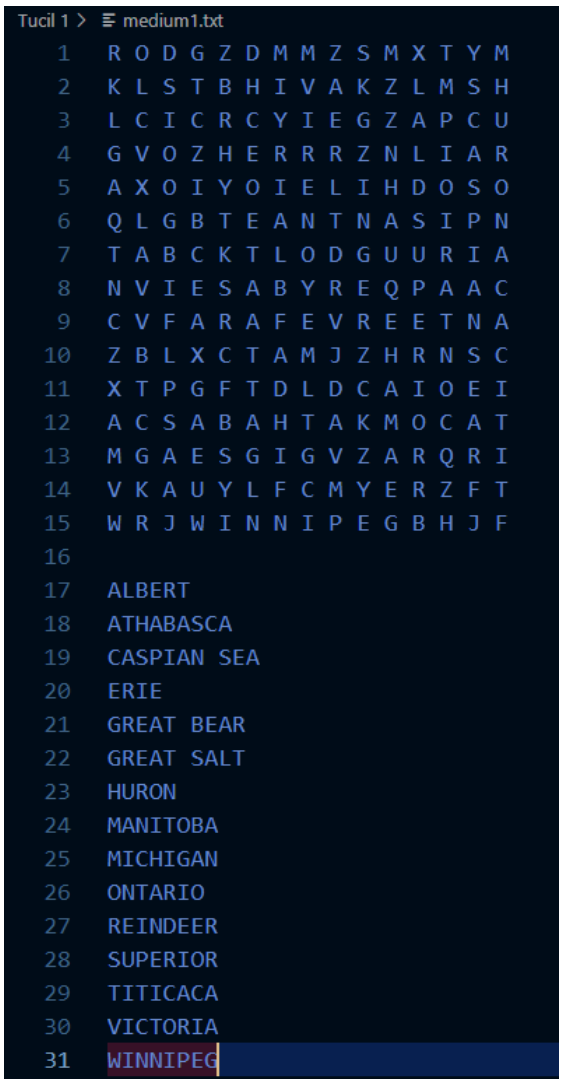
IV. Screenshot Input & Output

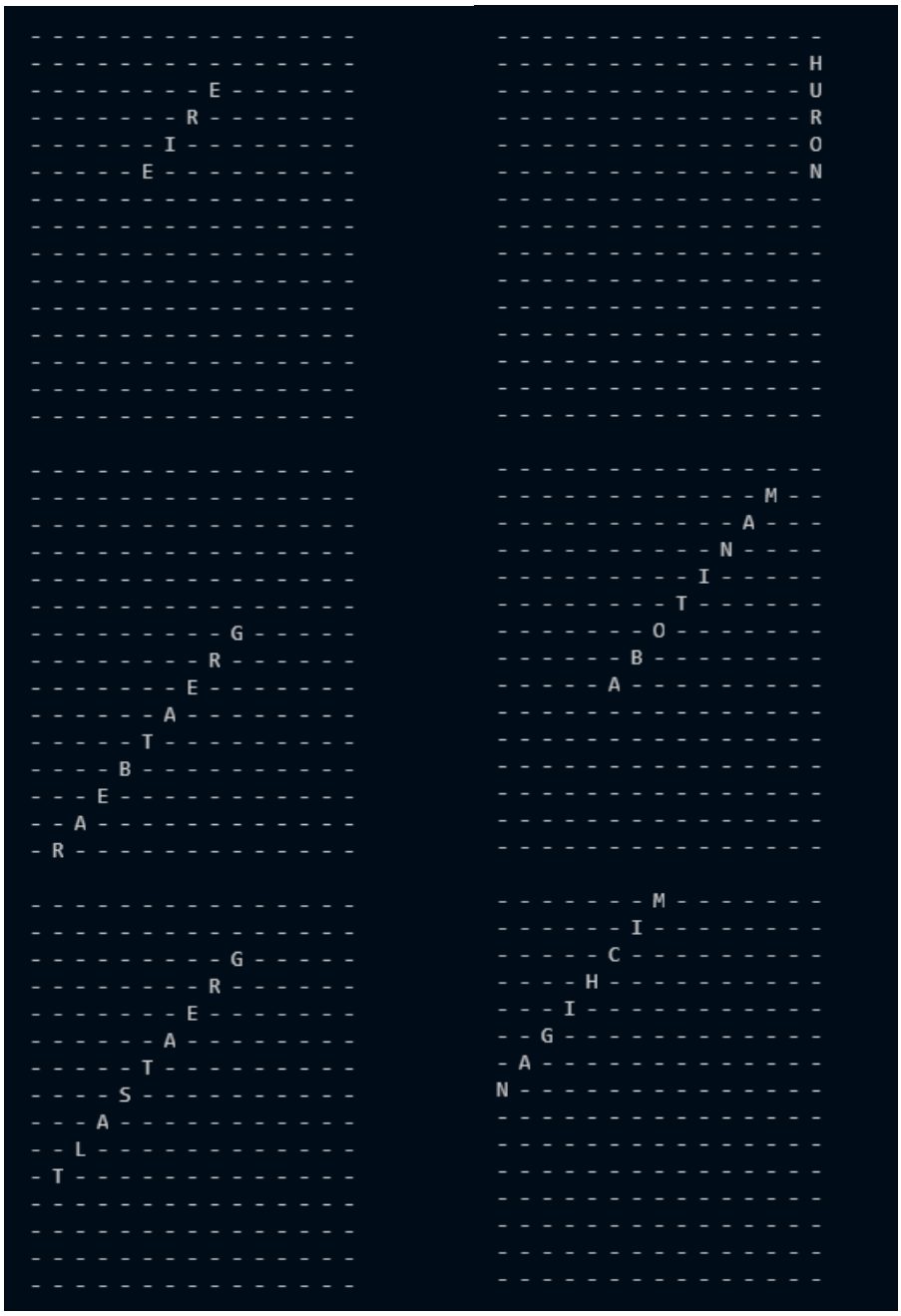
1. Input & Output Small/Easy (Diambil dari contoh spesifikasi Tugas Kecil):

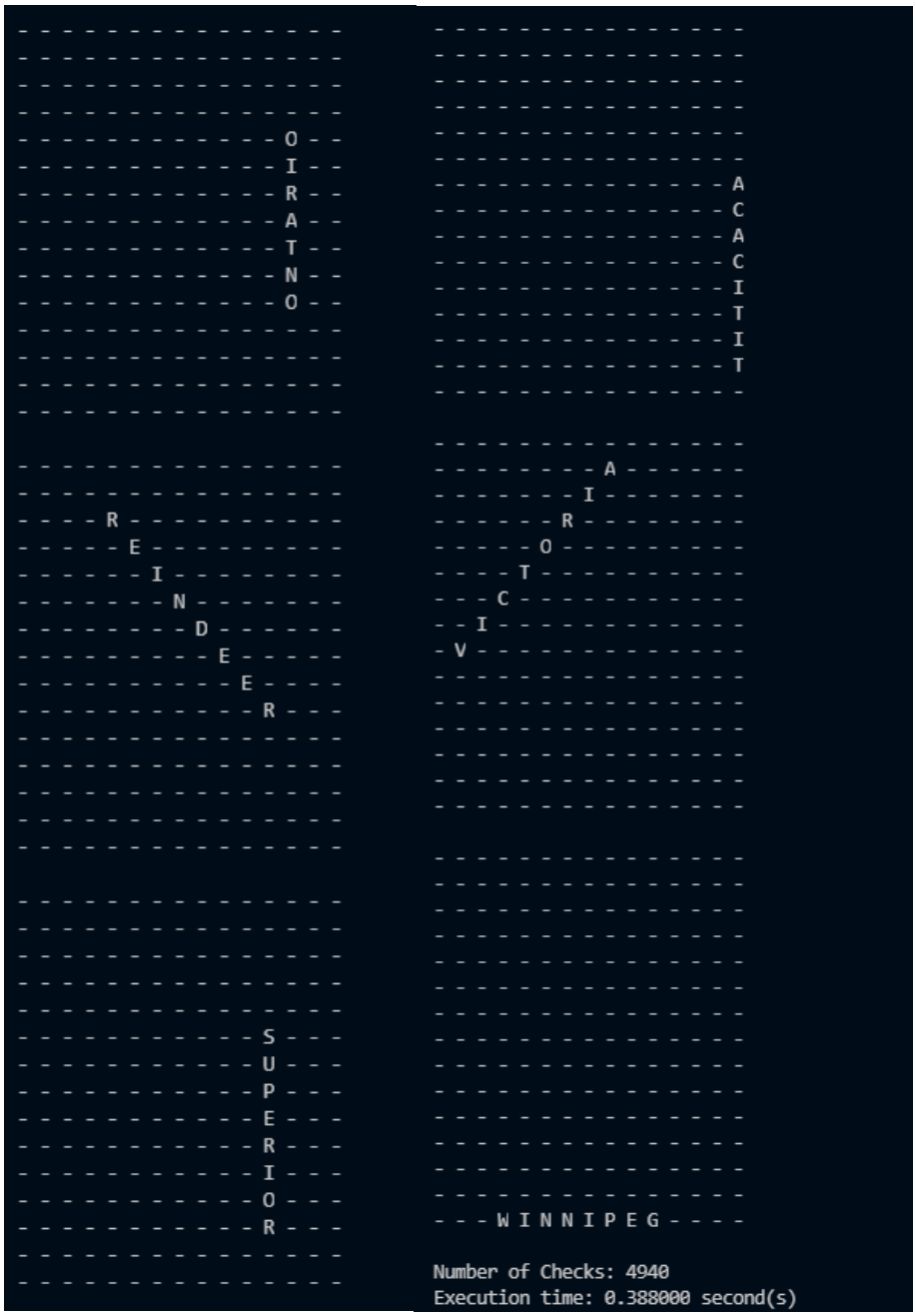




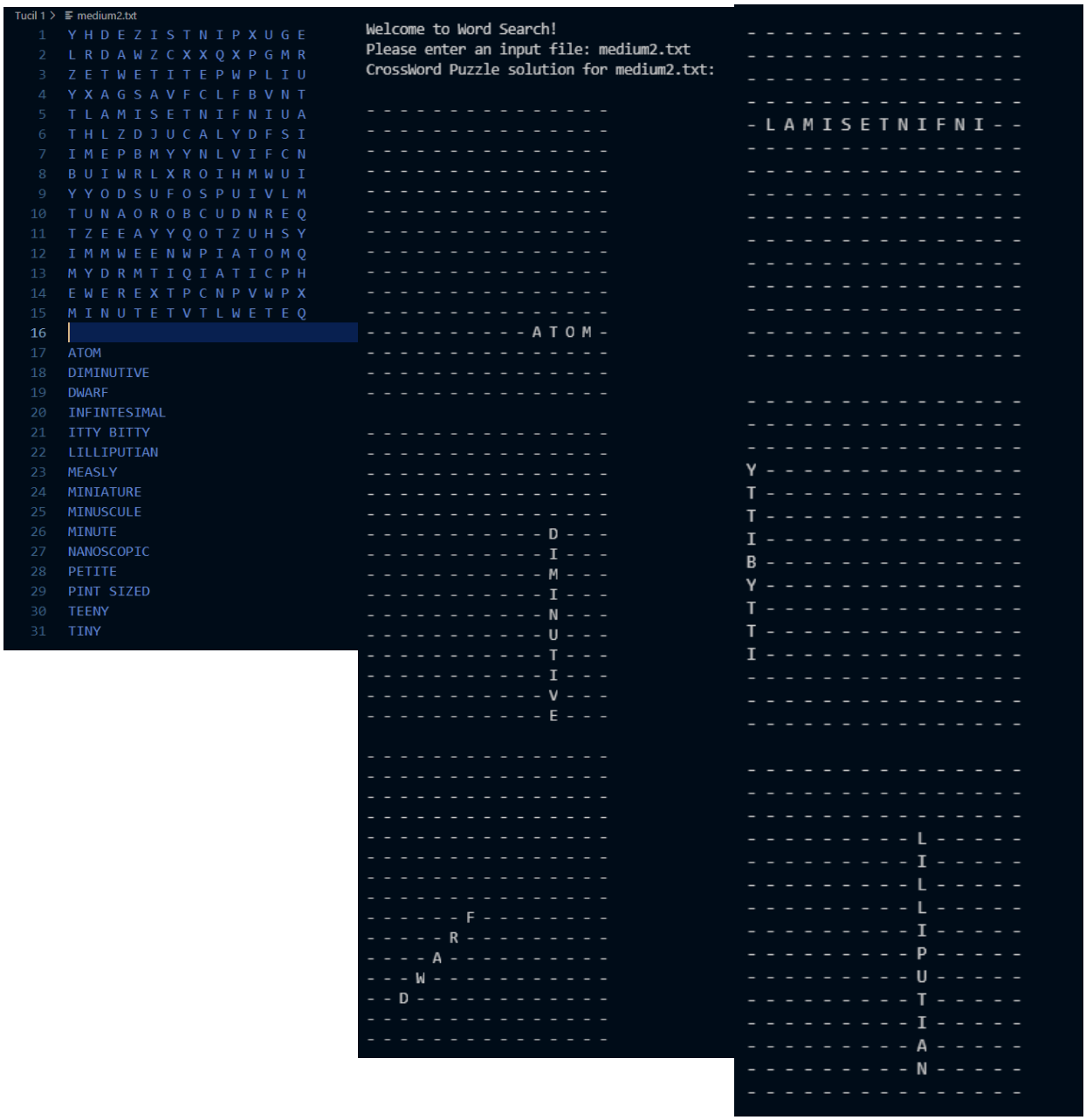
2. Input & Output Medium 1

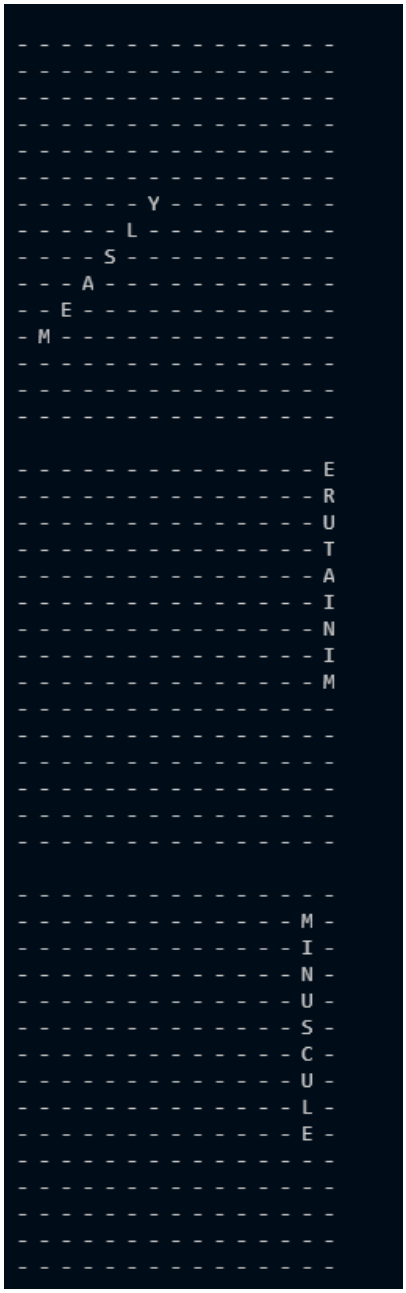


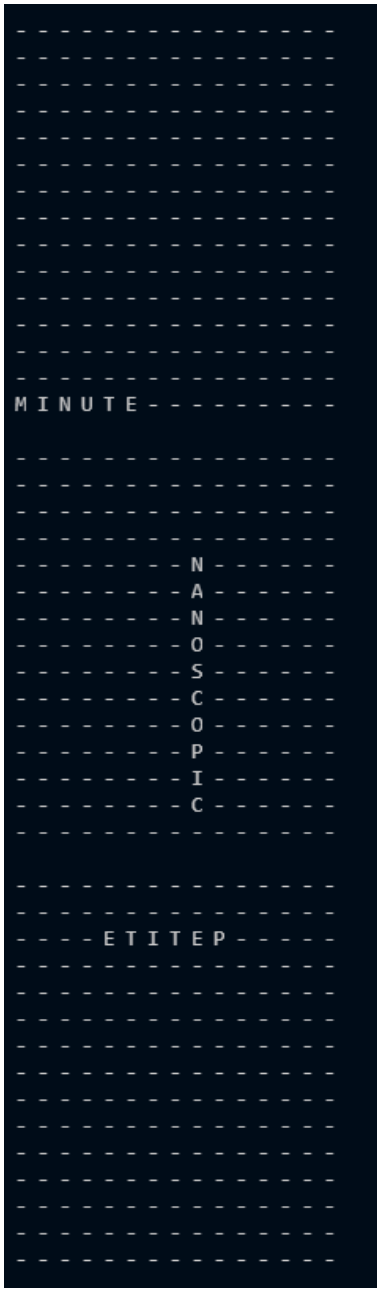


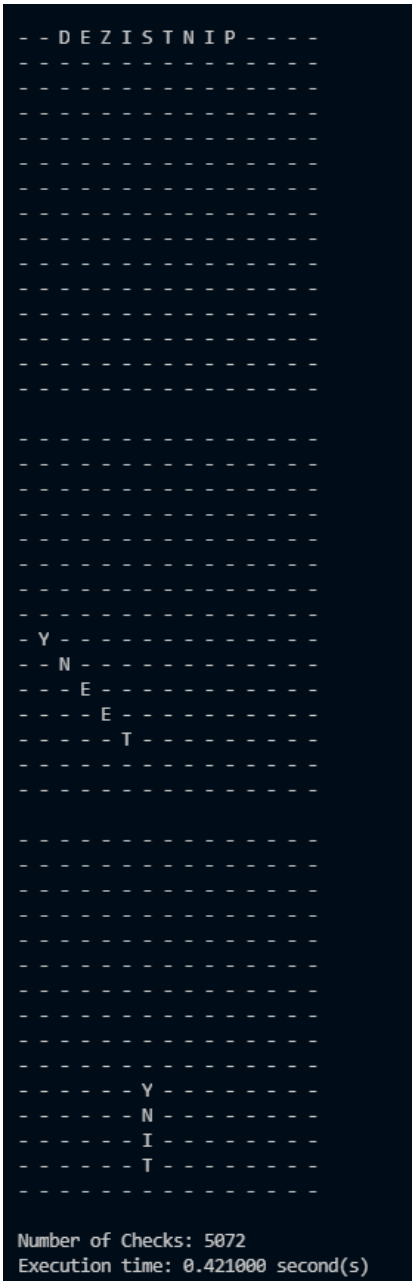

O
I
R
A
T
N
O
A
C
A
C
I
T
I
T
A
I
R
O
T
C
I
V
R
E
I
N
D
E
E
R
S
U
P
E
R
I
O
R
W I N N I P E G
Number of Checks: 4940
Execution time: 0.388000 second(s)

3. Input & Output Medium 2

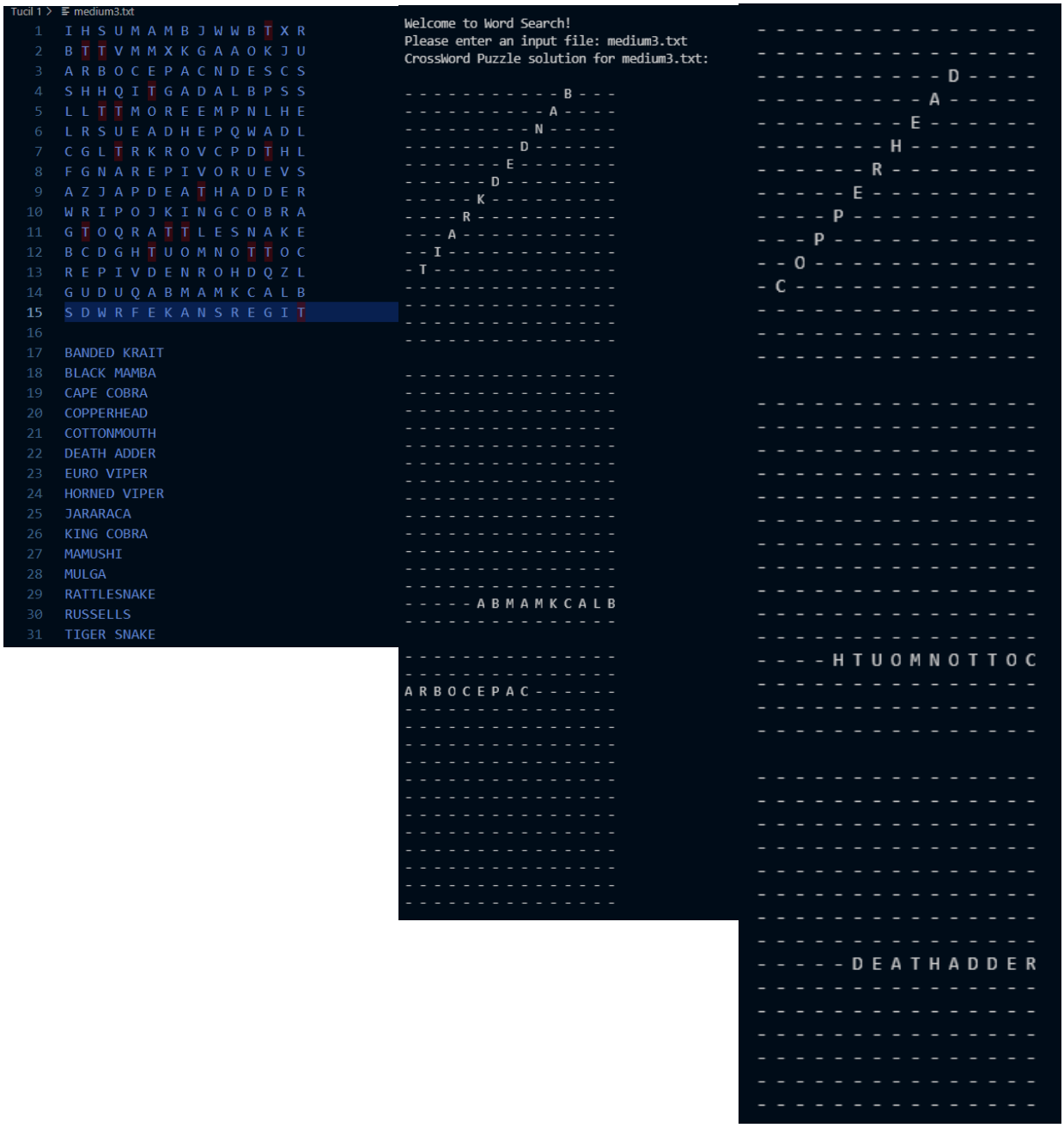


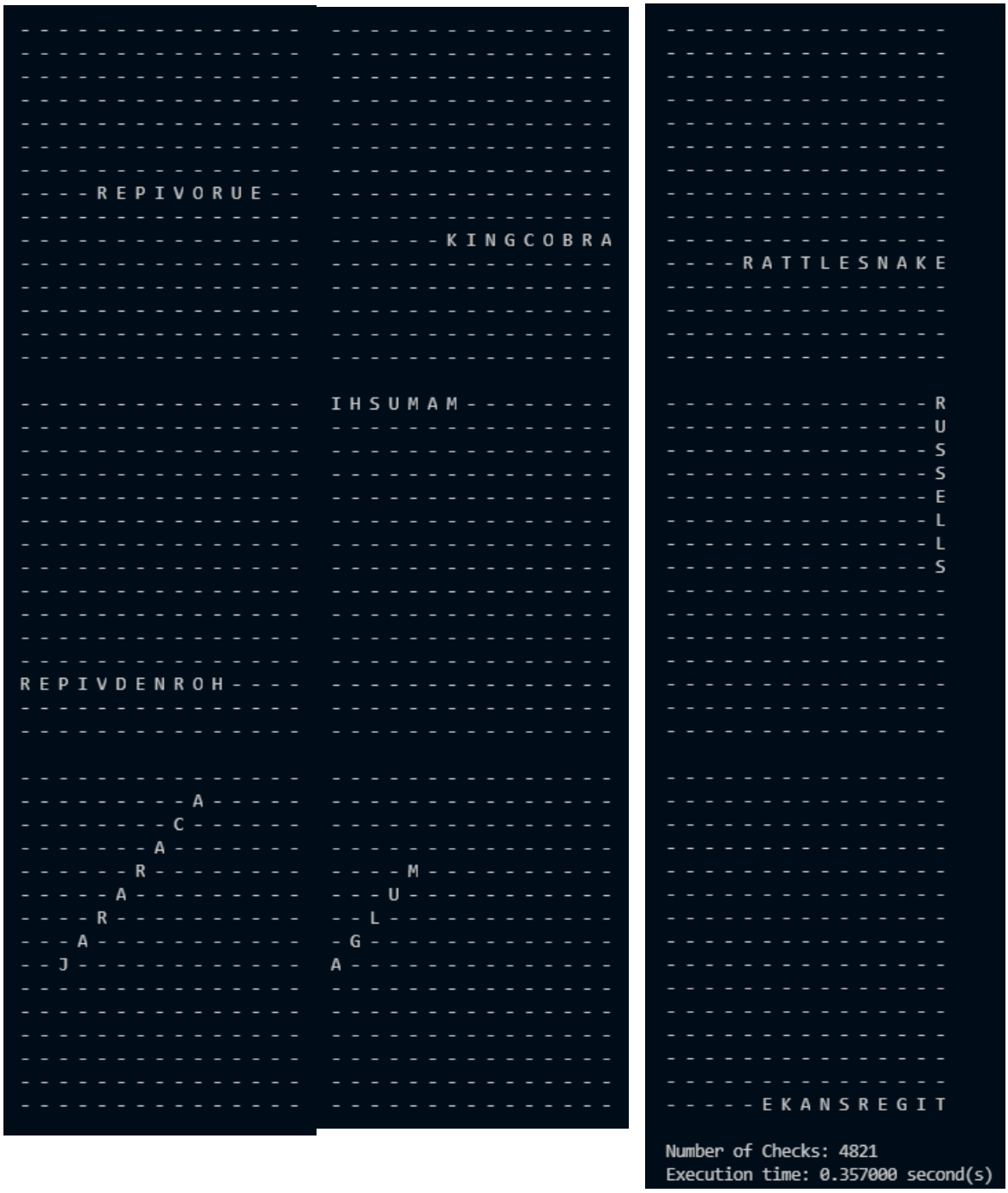

Y
L
S
A
E
M
ERUTAINIM
MINUSCULE


MINUTE
NANOSCOPIC
ETITEP

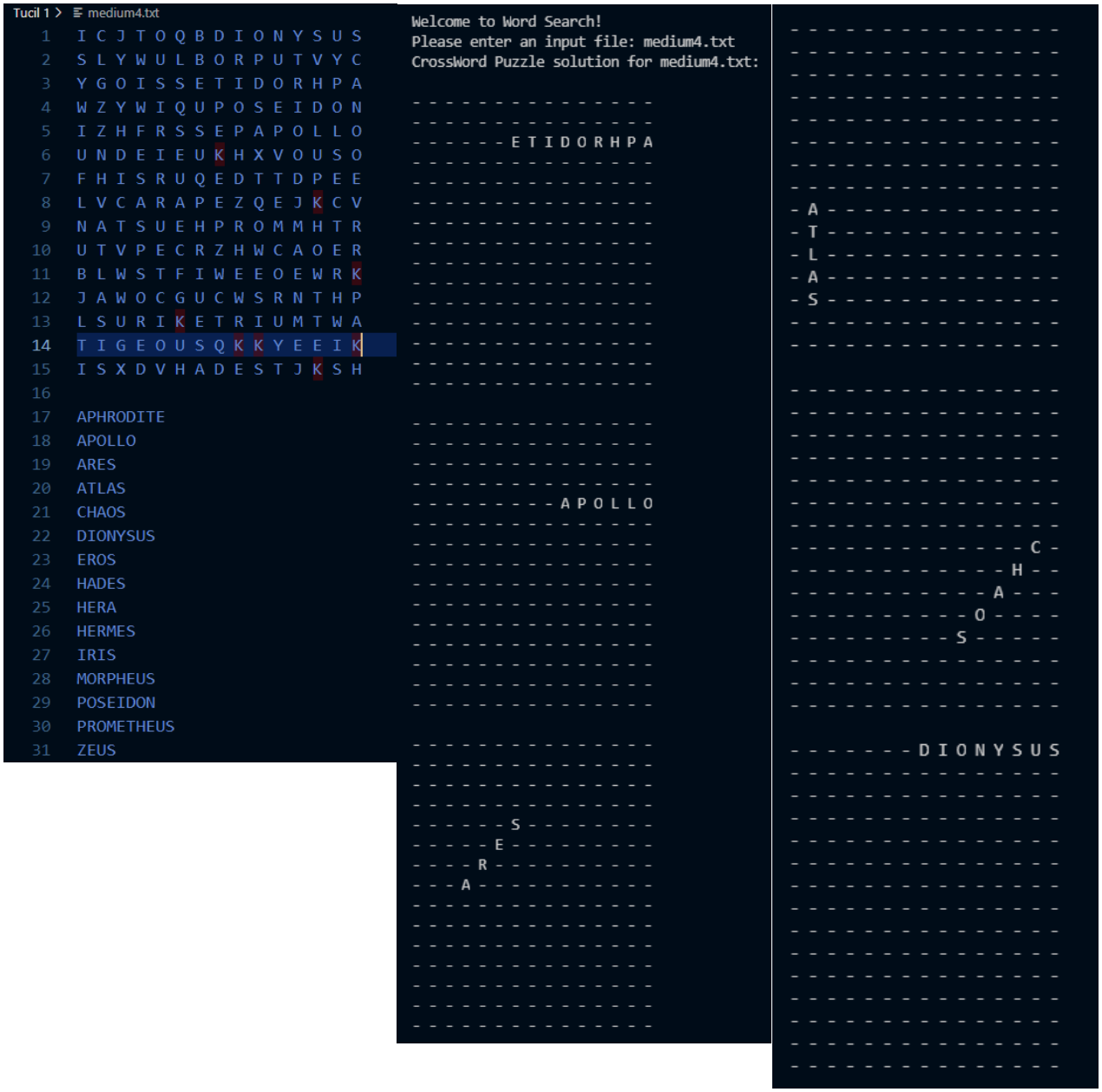

DEZISTNIP
Y
N
E
E
T
Y
N
I
T
Number of Checks: 5072
Execution time: 0.421000 second(s)

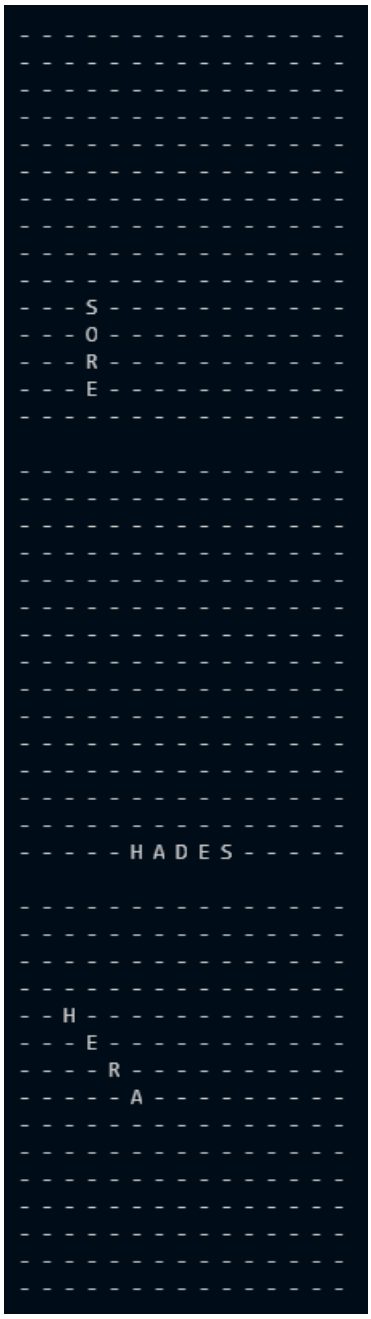
4. Input & Output Medium 3

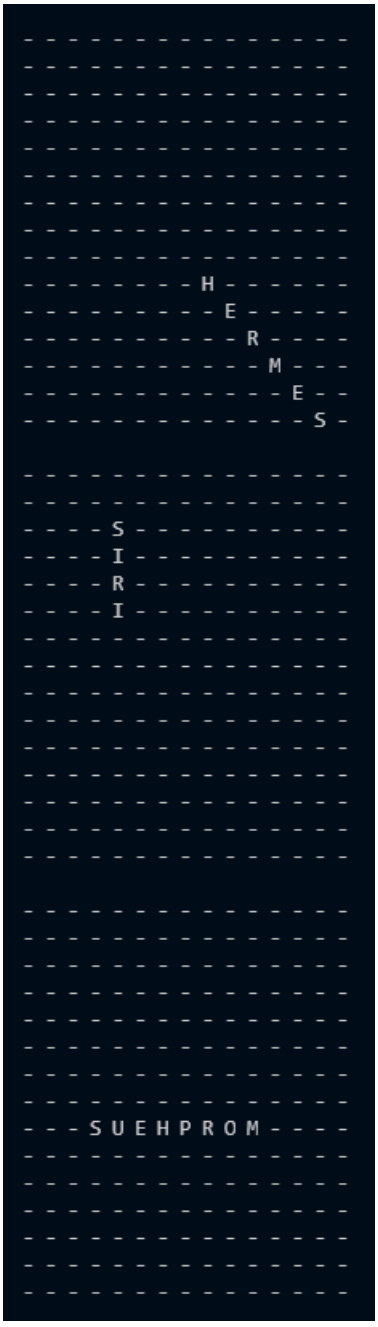


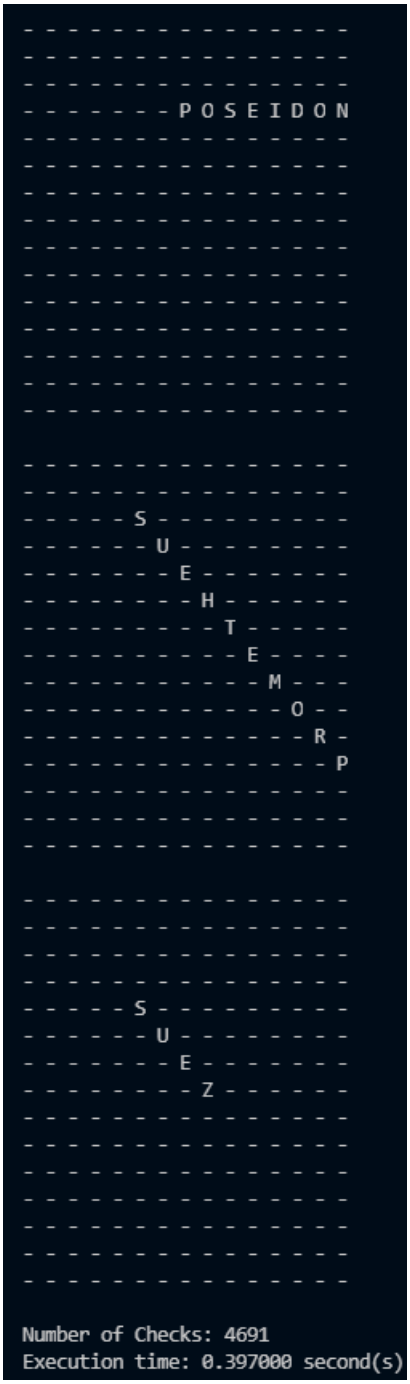

REPIVORUE
KINGCOBRA
IHSUMAM
RATTLESNAKE
RUSSELLS
REPIVDENROH
EKANSREGIT
Number of Checks: 4821
Execution time: 0.357000 second(s)

5. Input & Output Medium 4

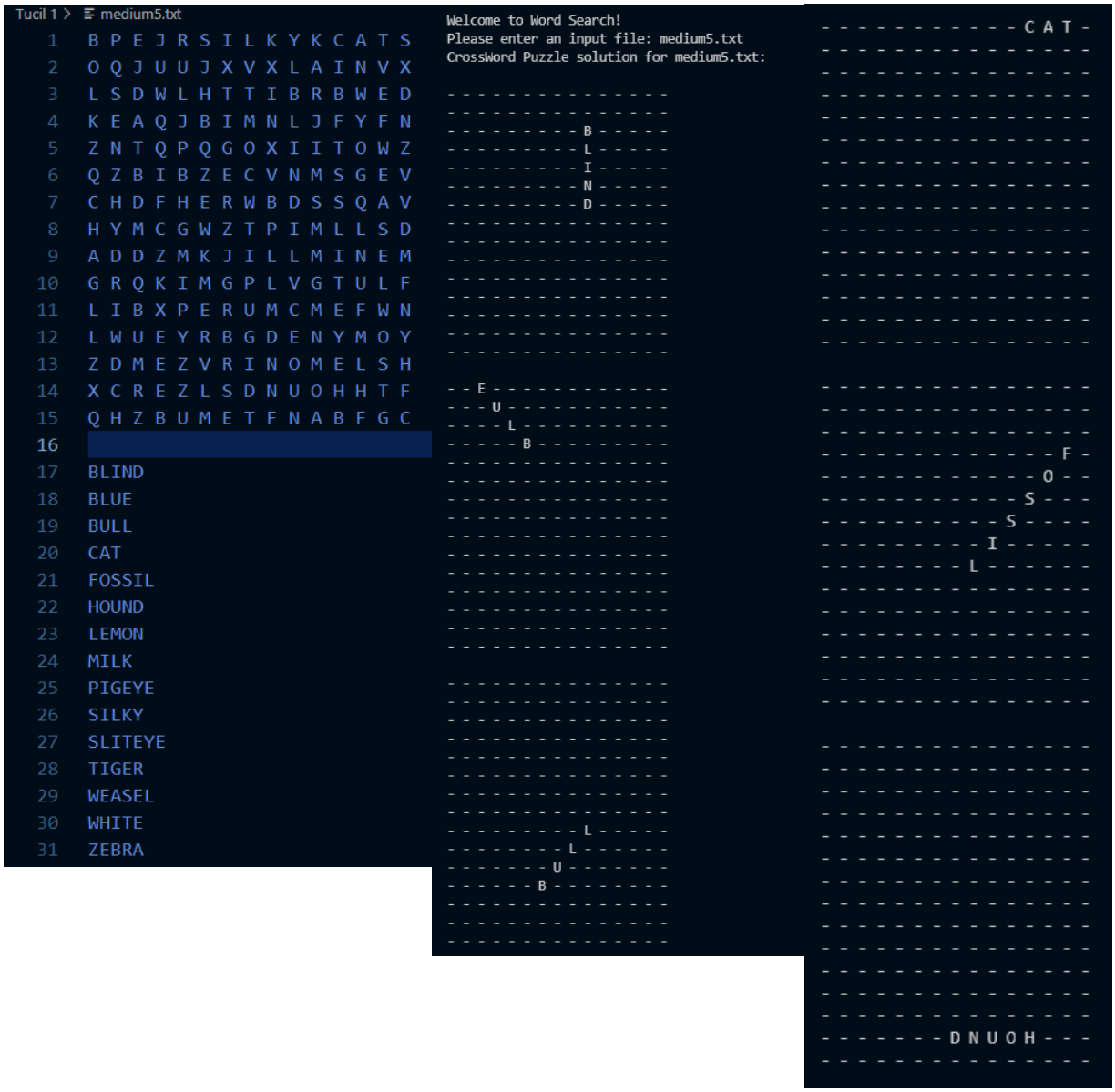


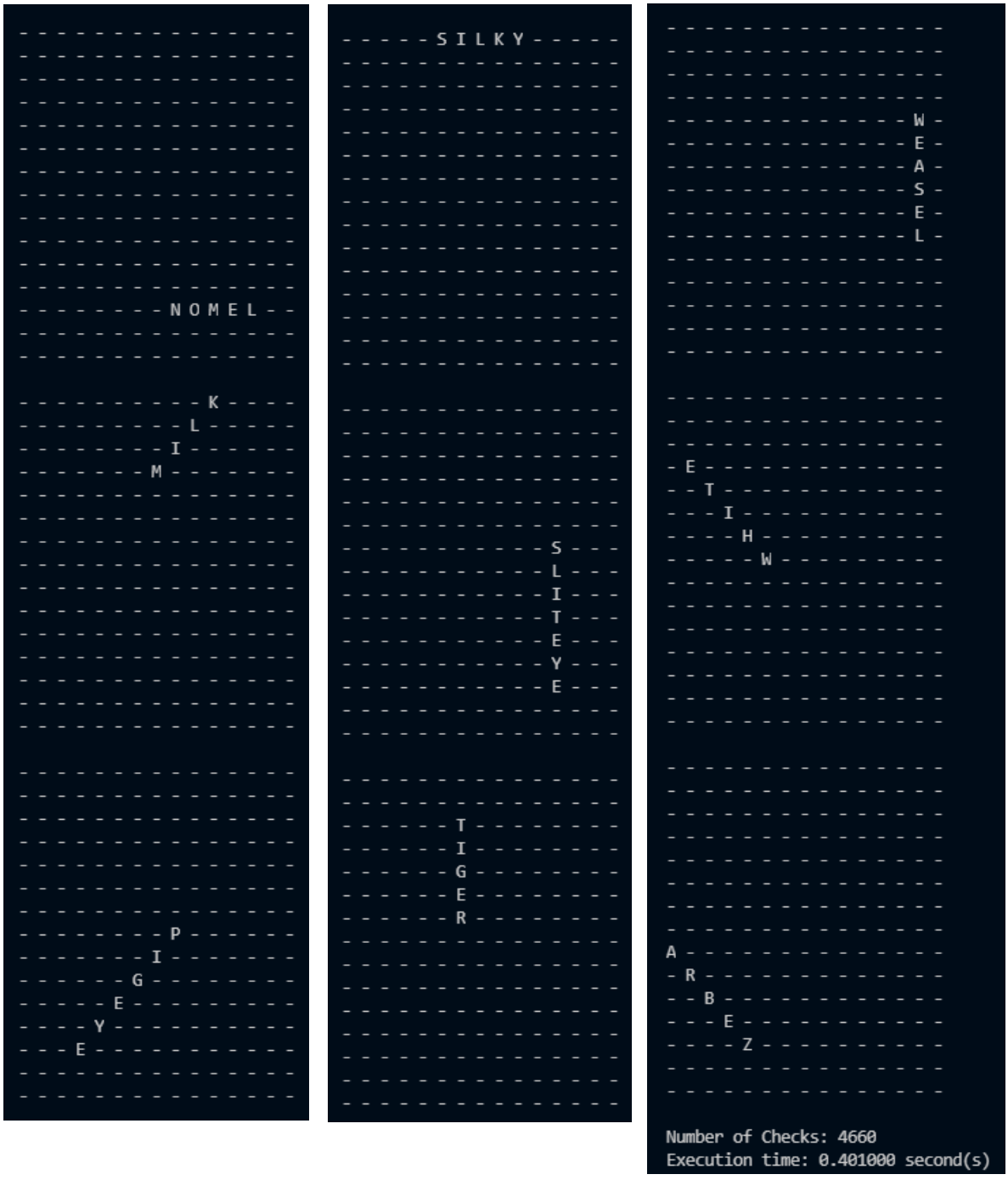

SORE
HADES
HERA


HERMES
SIRI
SUEHPROM

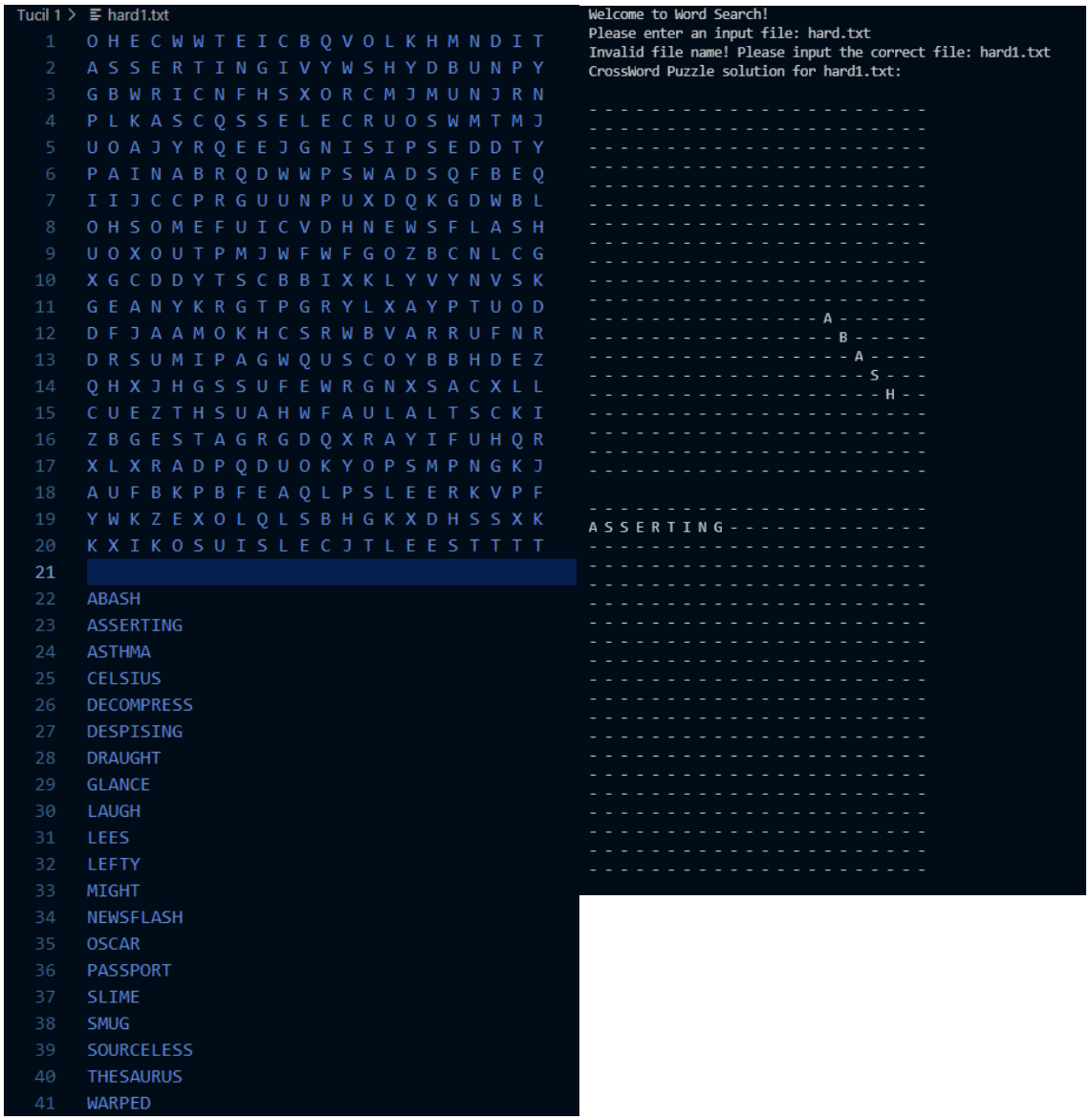

POSEIDON
SUTEHTEMOR P
SUEZ
Number of Checks: 4691
Execution time: 0.397000 second(s)

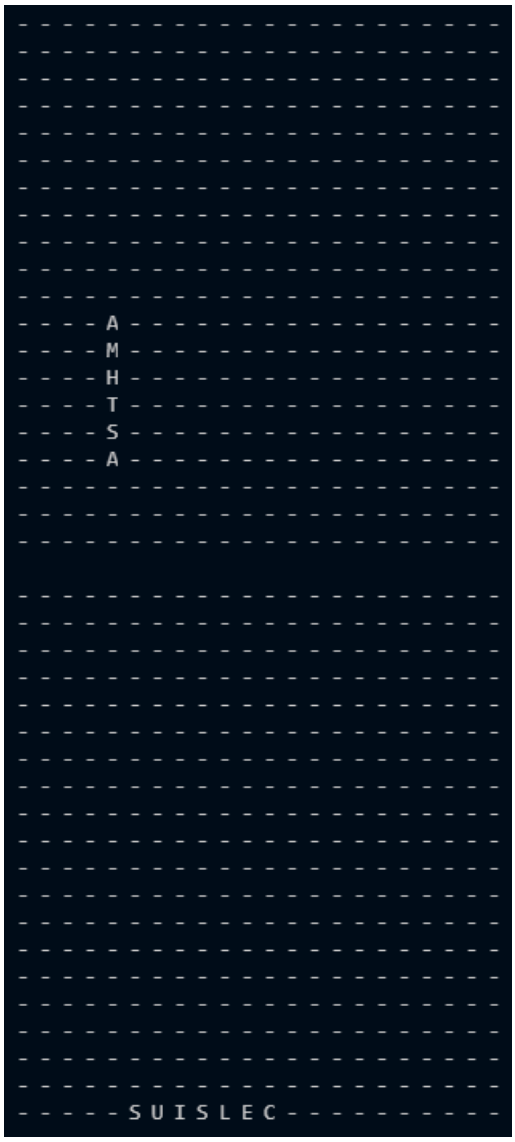
6. Input & Output Medium 5



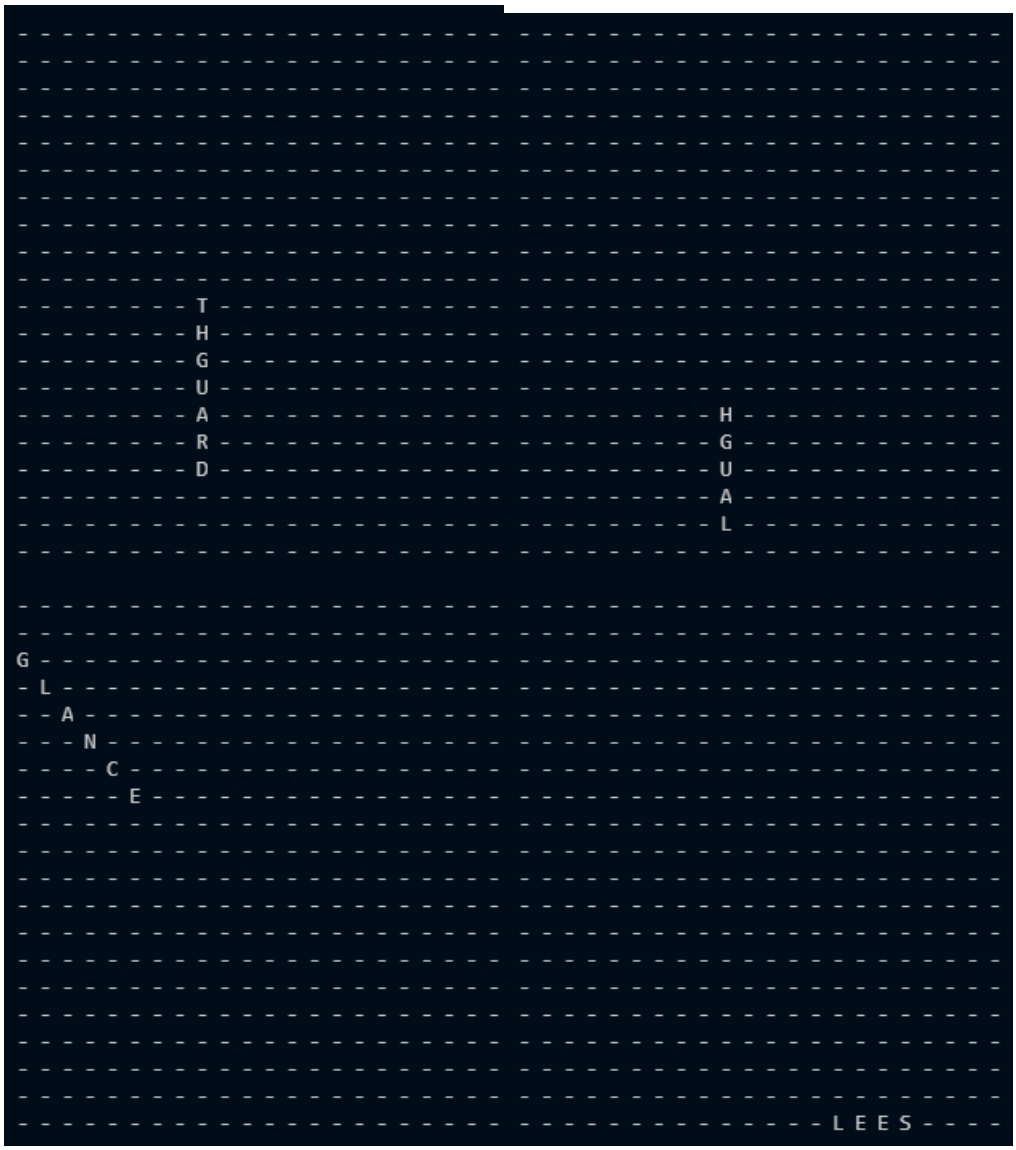

NOMEL
K
L
I
M
P
I
G
E
Y
E
SILKY
S
L
I
T
E
Y
E
T
I
G
E
R
W
E
A
S
E
L
E
T
I
H
W
A
R
B
E
Z
Number of Checks: 4660
Execution time: 0.401000 second(s)

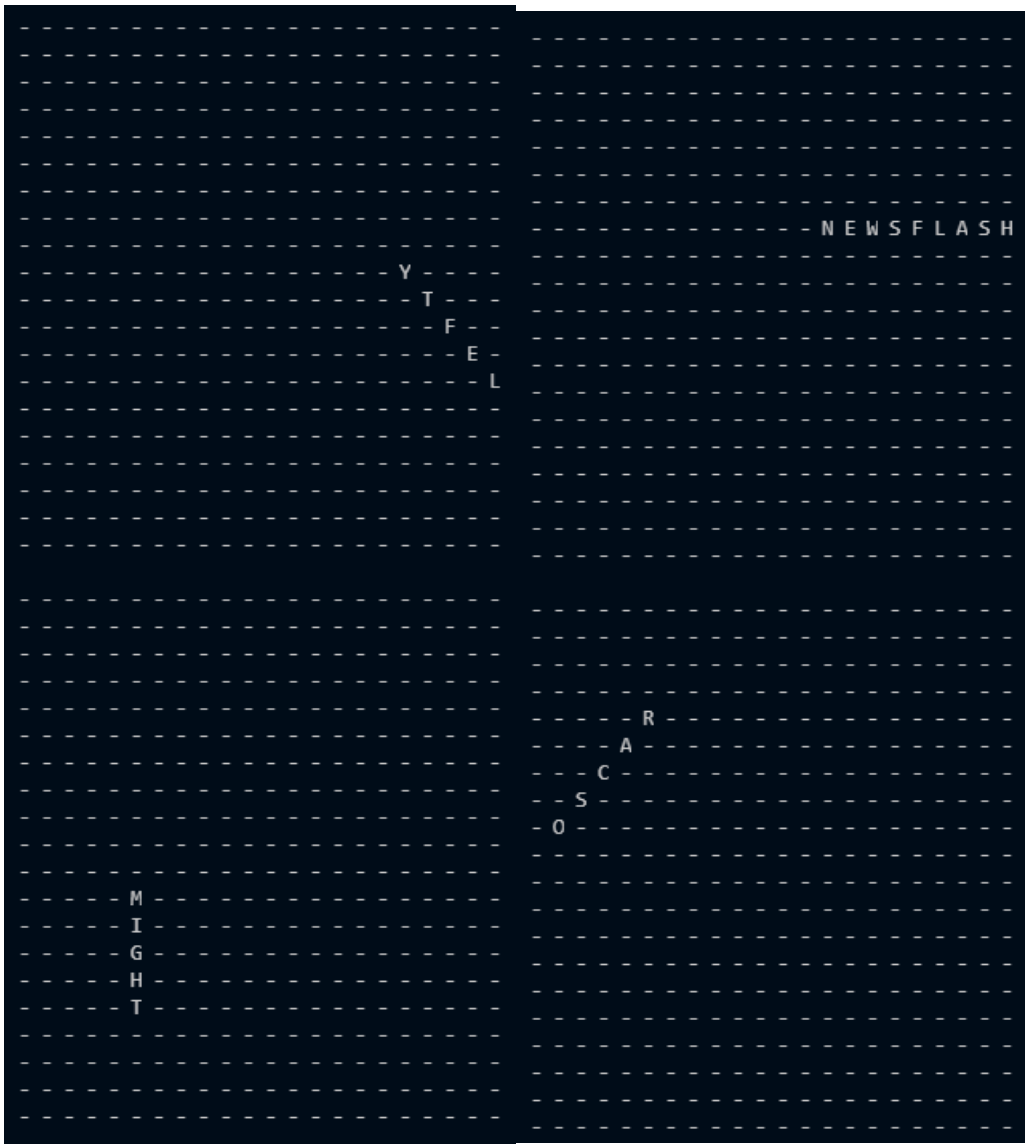
7. Input & Output Hard 1

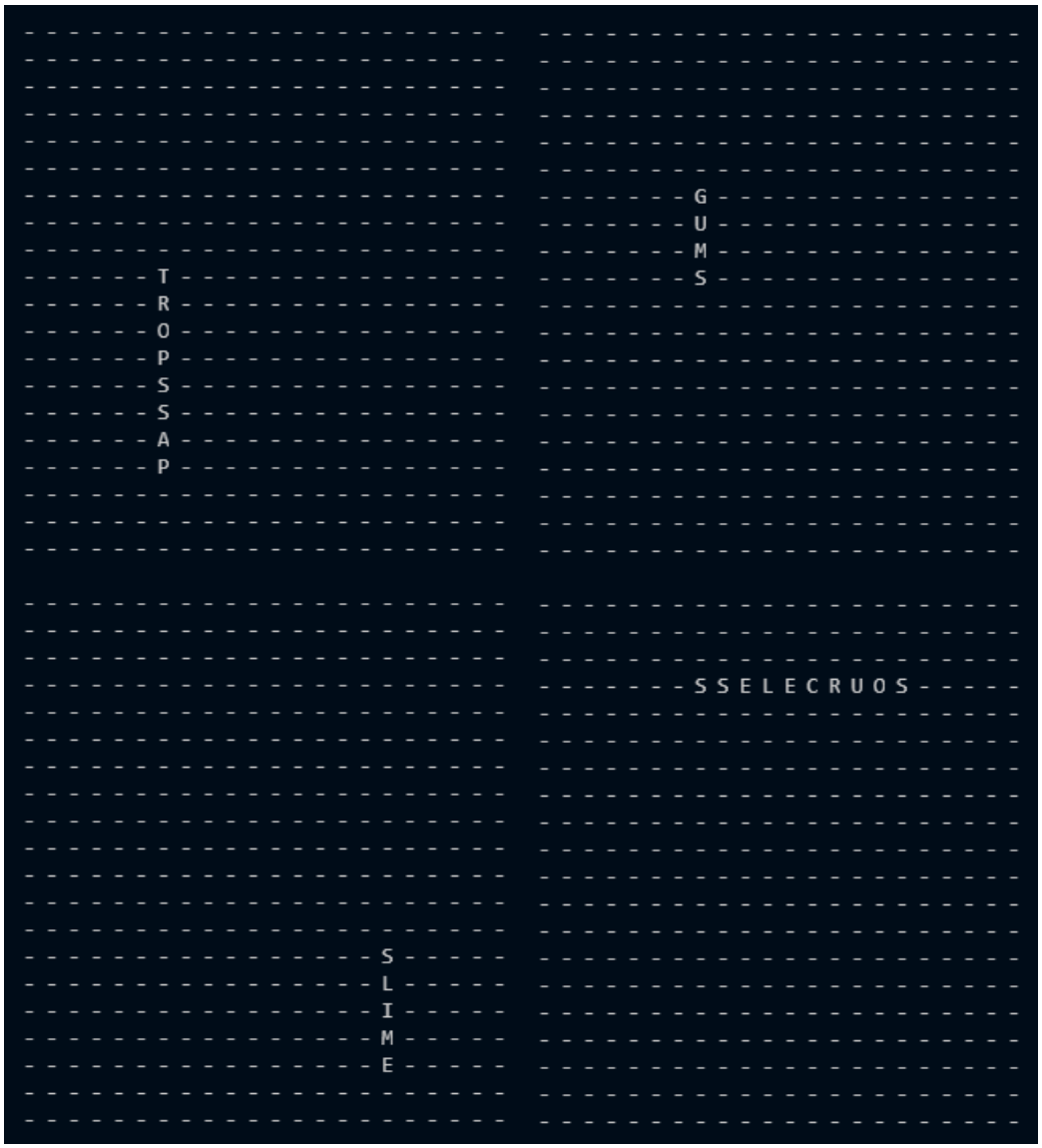


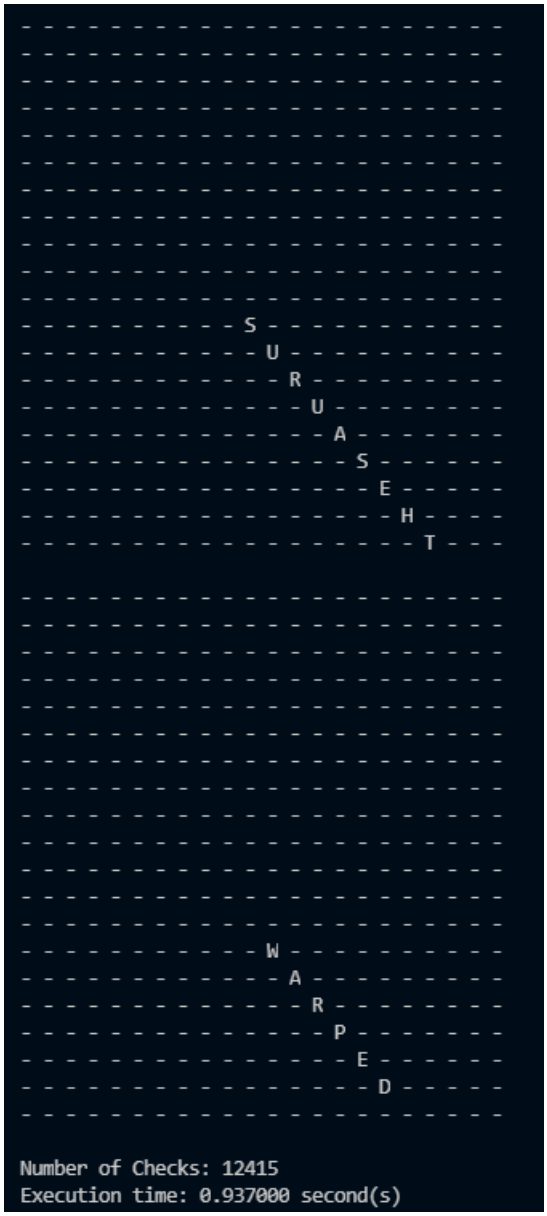

AMHTSA
SUISLEC


SSERPMOCED
GNISIPSED

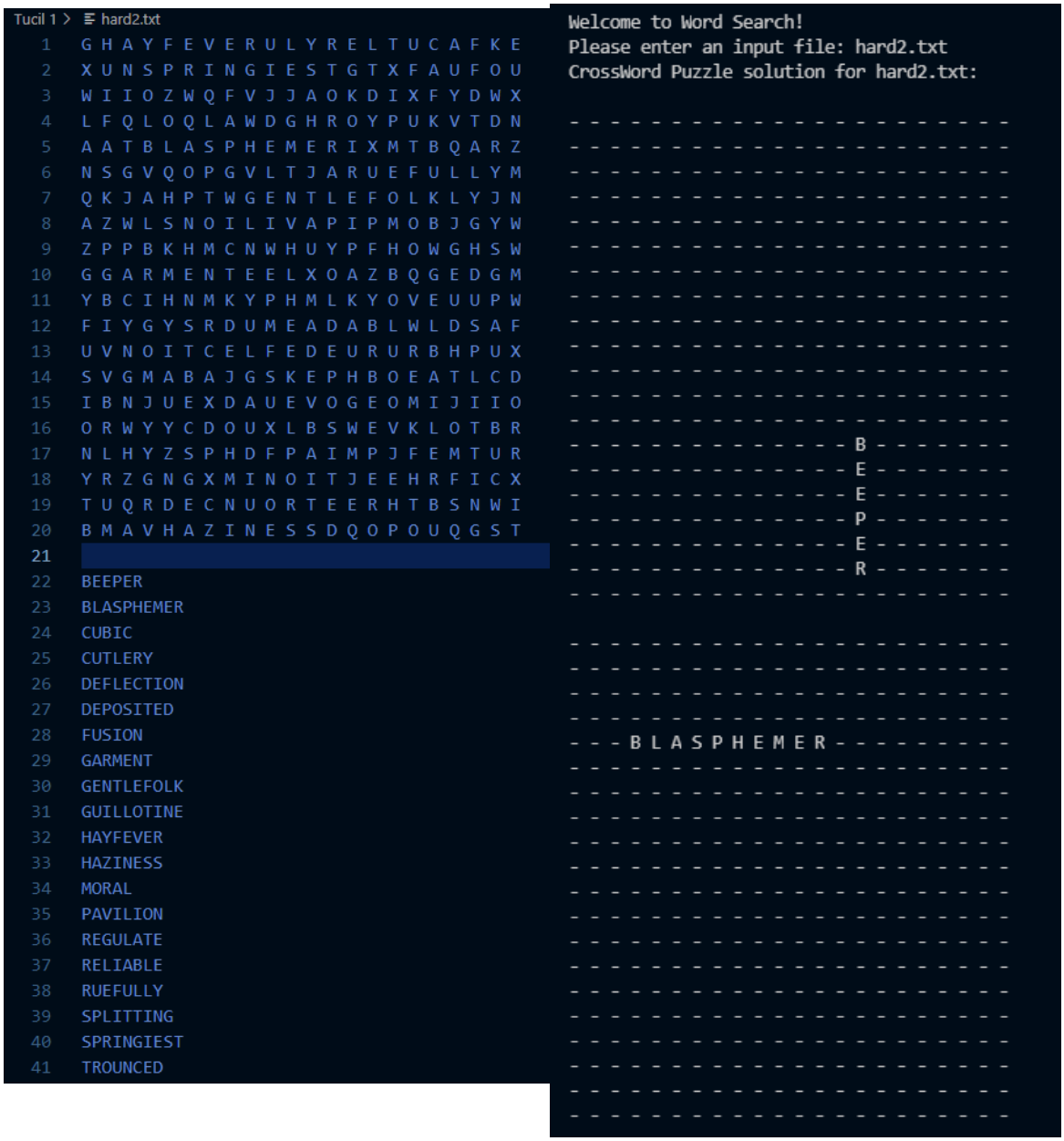

THGUARD
HGUAL
GLANCE
LEES


NEWSFLASH
YTFEL
MIGHT
RACSO


TROPSSAP
GUMS
SSELECRUOS
SLIME

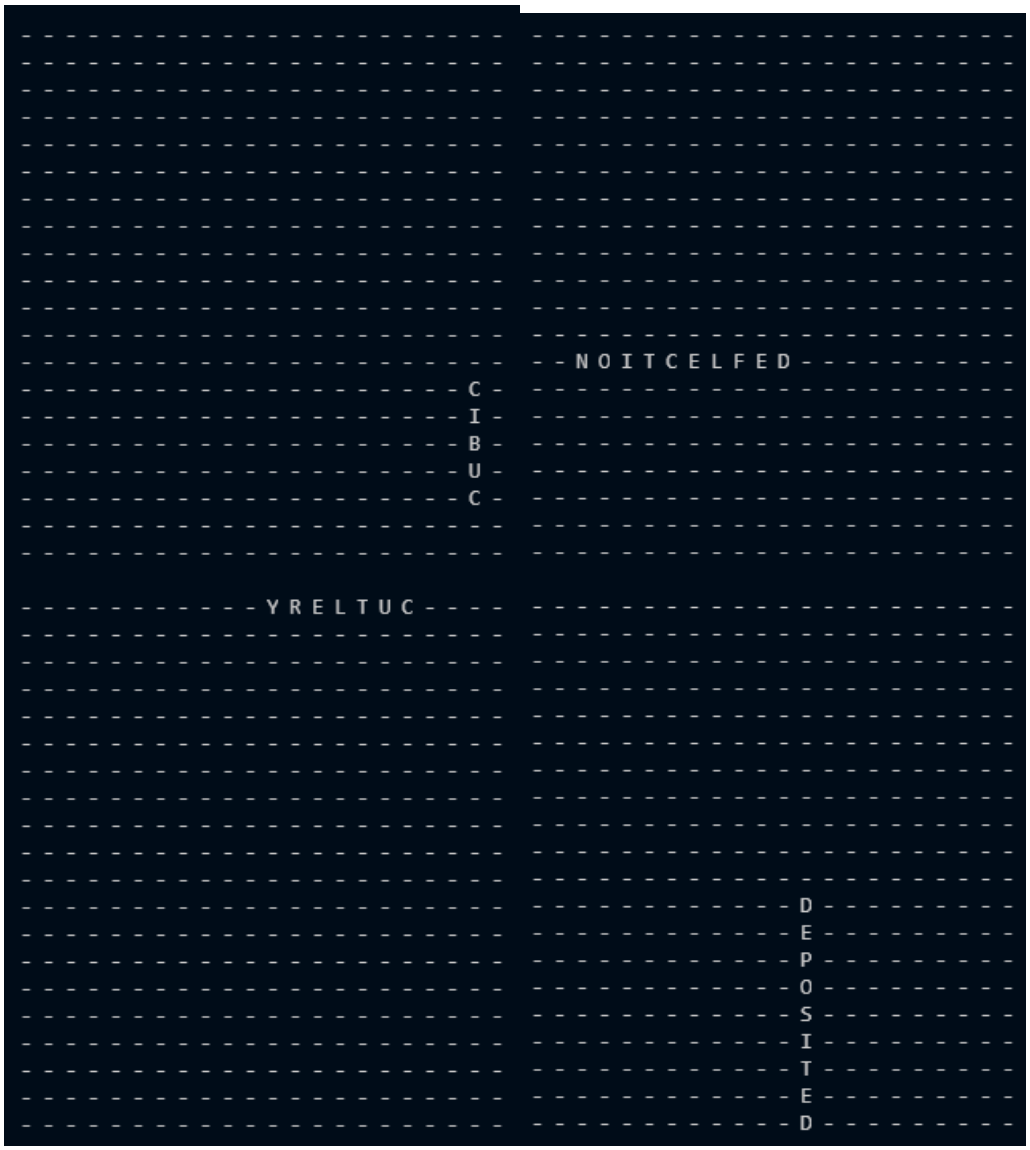

S
U
R
U
A
S
E
H
T
W
A
R
P
E
D
Number of Checks: 12415
Execution time: 0.937000 second(s)

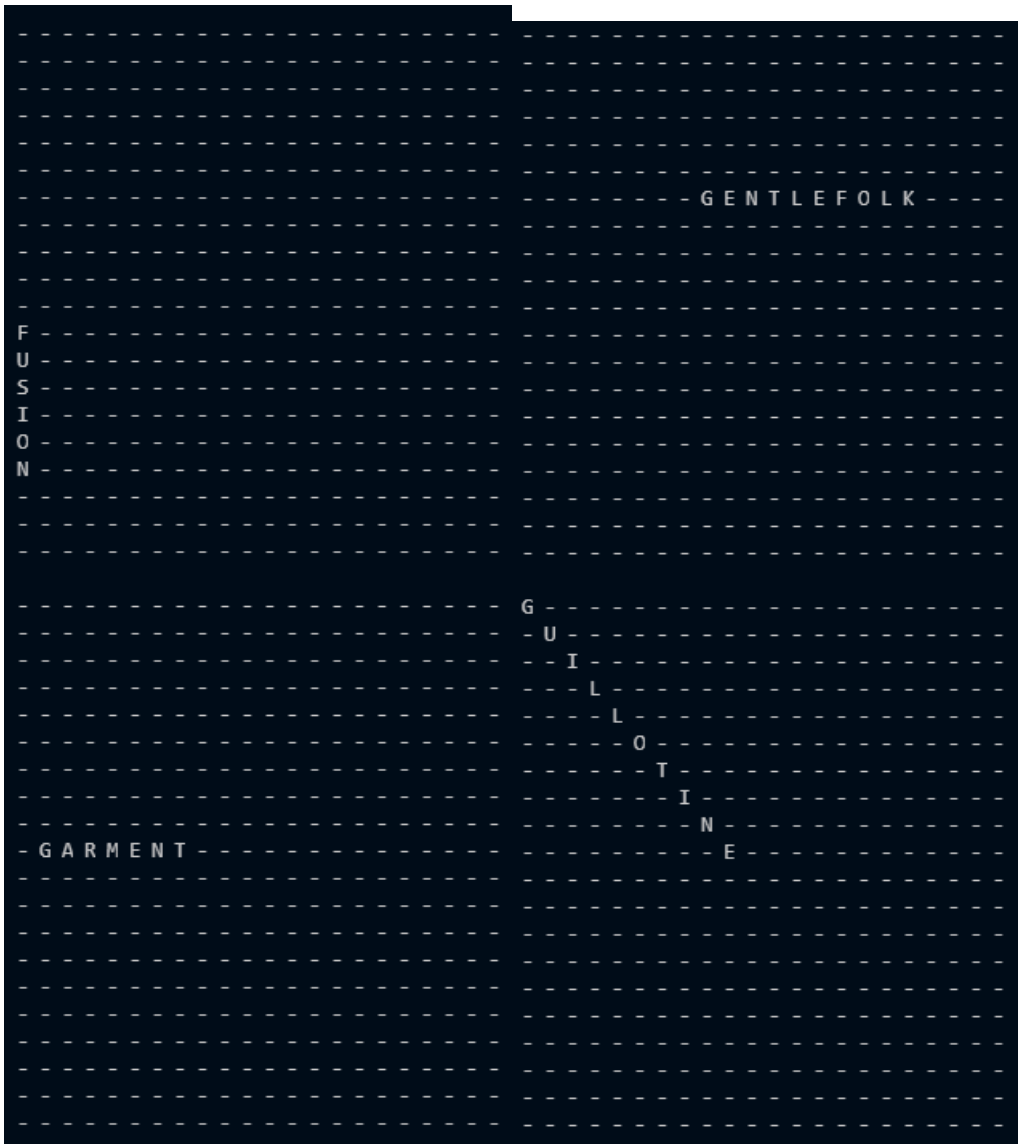
8. Input & Output Hard 2

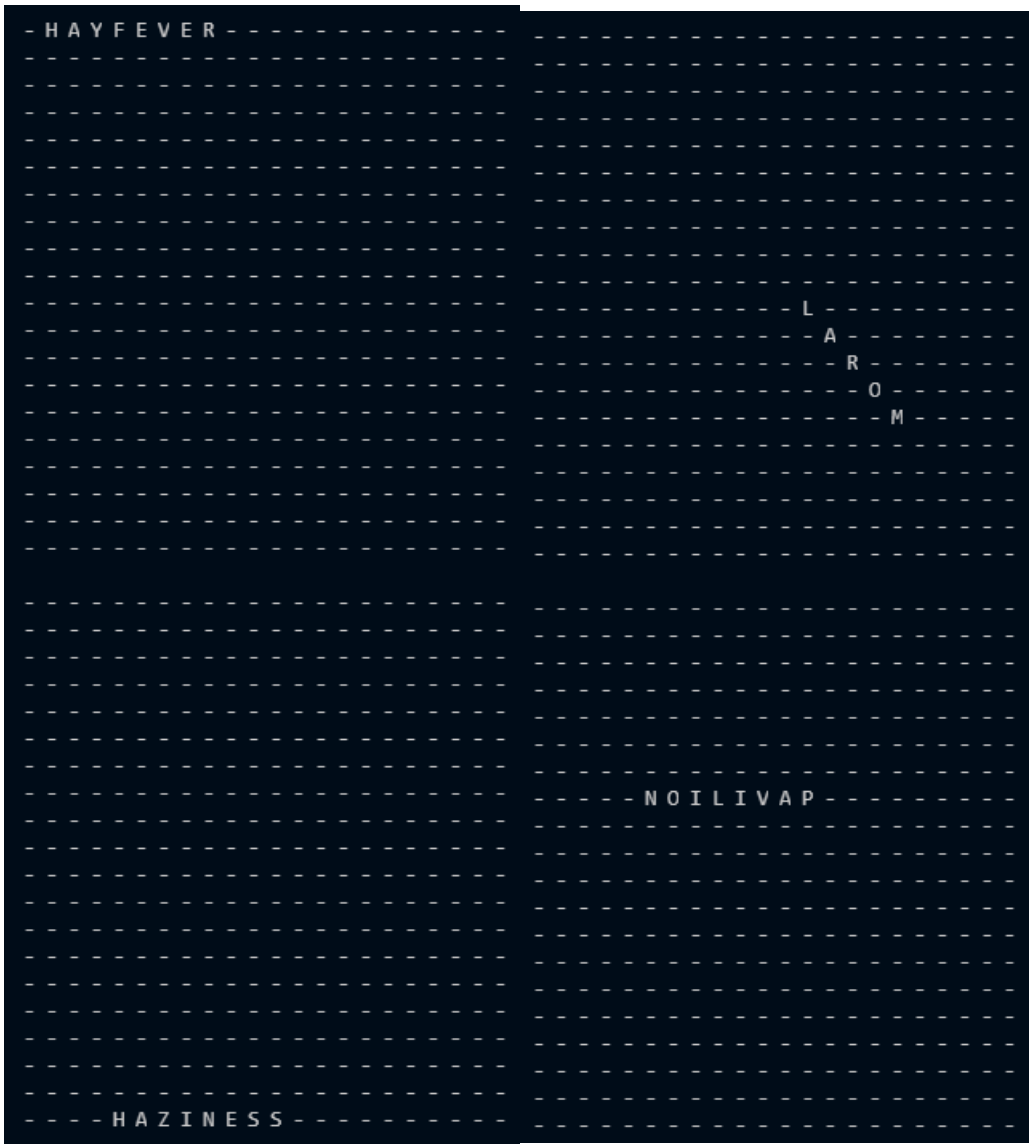


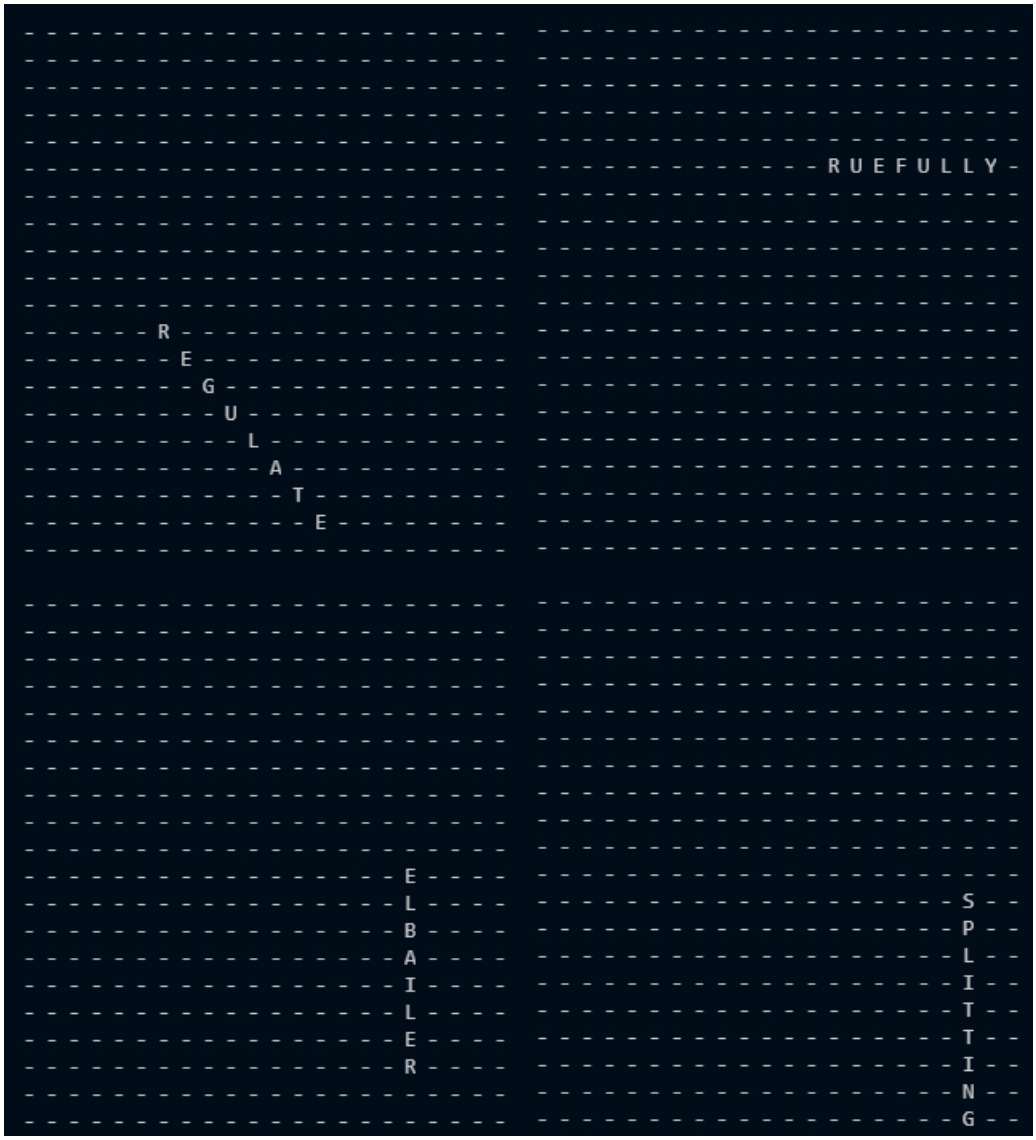
```
Tucil 1 > hard2.txt
 1  G H A Y F E V E R U L Y R E L T U C A F K E
 2  X U N S P R I N G I E S T G T X F A U F O U
 3  W I I O Z W Q F V J J A O K D I X F Y D W X
 4  L F Q L O Q L A W D G H R O Y P U K V T D N
 5  A A T B L A S P H E M E R I X M T B Q A R Z
 6  N S G V Q O P G V L T J A R U E F U L L Y M
 7  Q K J A H P T W G E N T L E F O L K L Y J N
 8  A Z W L S N O I L I V A P I P M O B J G Y W
 9  Z P P B K H M C N W H U Y P F H O W G H S W
10  G G A R M E N T E E L X O A Z B Q G E D G M
11  Y B C I H N M K Y P H M L K Y O V E U U P W
12  F I Y G Y S R D U M E A D A B L W L D S A F
13  U V N O I T C E L F E D E U R U R B H P U X
14  S V G M A B A J G S K E P H B O E A T L C D
15  I B N J U E X D A U E V O G E O M I J I I O
16  O R W Y Y C D O U X L B S W E V K L O T B R
17  N L H Y Z S P H D F P A I M P J F E M T U R
18  Y R Z G N G X M I N O I T J E E H R F I C X
19  T U Q R D E C N U O R T E E R H T B S N W I
20  B M A V H A Z I N E S S D Q O P O U Q G S T
21
22  BEEPER
23  BLASPHEMER
24  CUBIC
25  CUTLERY
26  DEFLECTION
27  DEPOSITED
28  FUSION
29  GARMENT
30  GENTLEFOLK
31  GUILLOTINE
32  HAYFEVER
33  HAZINESS
34  MORAL
35  PAVILION
36  REGULATE
37  RELIABLE
38  RUEFULLY
39  SPLITTING
40  SPRINGIEST
41  TROUNCED
```

```
Welcome to Word Search!
Please enter an input file: hard2.txt
CrossWord Puzzle solution for hard2.txt:

- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - B - - - - - -
- - - - - - - - - - - - - - - - E - - - - - -
- - - - - - - - - - - - - - - - E - - - - - -
- - - - - - - - - - - - - - - - P - - - - - -
- - - - - - - - - - - - - - - - E - - - - - -
- - - - - - - - - - - - - - - - R - - - - - -
- - - - - - - - - - - - - - - - - - - - - -

- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - B L A S P H E M E R - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
```


NOITCELFED
CIBUC
YRELTUC
DEPOSITED


GENTLEFOLK
FUSION
GUILLOTINE
GARMENT


HAYFEVER
L
A
R
O
M
NOILIVAP
HAZINESS


RUEFULLY
REGULATE
ELBAILER
SPLITTING

```
- - - - - - - - - - - - - - - - - - - - - -
- - - S P R I N G I E S T - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -

- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
- - - - D E C N U O R T - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -

Number of Checks: 11744
Execution time: 0.937000 second(s)
```

V. Checklist Fitur Program

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

VI. Link Drive Kode Program

https://drive.google.com/drive/folders/1miRwRQNcH1bkfWazhA18YJcyrsQjmdU0?usp=sharing

*** File yang terdapat di drive diatas sudah saya bagi menjadi empat folder demi kerapihan, akan tetapi untuk mencoba/testing dengan .txt file, file yang akan di test harus dipindahkan terlebih dahulu ke directory yang sama dengan directory executable. ***