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

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

Disusun oleh:
Ahmad Alfani Handoyo
13520023
K-02

Program Studi S1 Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

A. Algoritma *Brute Force* untuk menyelesaikan *Word Search Puzzle*

Algoritma yang digunakan untuk menyelesaikan *word search puzzle* adalah metode *brute force* dengan teknik heuristik untuk hanya mengecek indeks yang memuat seluruh panjang kata yang diterima (pada masing-masing arah mata angin) sehingga tidak ada masalah *out-of-bounds checking*.

Langkah-langkah algoritma yang digunakan dalam menyelesaikan masalah tersebut adalah sebagai berikut:

1. Telusuri seluruh indeks berelemen karakter pada matriks (mulai dari pojok kiri atas hingga pojok kiri kanan)
2. Bandingkan karakter pada indeks matriks dengan huruf pertama kata yang dicari. Bila sama, maka cek huruf-huruf kata berikutnya pada semua 8 mata angin. Bila tidak, geser ke indeks berikutnya.
3. Pada pengecekan setiap indeks mata angin, bila huruf sama geser ke posisi berikutnya pada matriks sesuai dengan arah mata angin. Bila huruf tidak sama cek pada arah mata angin yang lain.
4. Bila ditemukan satu mata angin yang mana seluruh huruf pada kata yang dicari sesuai, maka solusi untuk kata tersebut ditemukan untuk indeks tersebut. Bila tidak, geser pencarian ke indeks berikutnya.
5. Bila hingga indeks terakhir tidak ditemukan indeks yang memuat solusi kata, maka solusi tidak ditemukan.

B. *Source Code* Program

Program ditulis pada bahasa C dengan semua program diletakkan pada satu *file* bernama *main.c* pada folder */src*. Terlampir di bawah *source code* program seperti pada *file main.c*:

```c
/*  NAMA  : Ahmad Alfani Handoyo
    NIM   : 13520023
    Kelas : K02
    Tugas Kecil 1 IF2211 Strategi Algoritma Semester II tahun 2021/2022 */

#include <stdio.h>
#include <string.h>
#include <time.h>

typedef struct {
   char word[1000];
   int length;
```

```
} Word;

typedef struct {
    char matrix[100][100];
    int row;
    int col;
} Matrix;

// Print matrix
void PrintMatrix (Matrix matrix) {
    int i, j;
    for (i=0; i < matrix.row; i++) {
        for (j=0; j < matrix.col; j++) {
                printf("%c ", matrix.matrix[i][j]);
            }
            printf("\n");
        }
}

// Make a copy of a matrix with '-' in every index
void CopyEmptyMatrix (Matrix original, Matrix *copy) {
    Matrix temp = original;
    int i, j;
    for (i=0; i < temp.row; i++) {
        for (j=0; j < temp.col; j++) {
            temp.matrix[i][j] = '-';
        }
    }
    *copy = temp;
}

// Check first letter of word
void solveFirst (Matrix Puzzle, int pRow, int pCol, Word word, int *compare, int *firstFound) {
    int bool;
    if (Puzzle.matrix[pRow][pCol] != word.word[0]) {
        bool = 0;
    } else {
        bool = 1;
    }
    *compare = *compare + 1;
    *firstFound = bool;
}

// Check for direction East
void solveE (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);
```

```
    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if (pCol < (Puzzle.col - word.length + 1)) {
            int i = pRow, j = pCol+1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
                j++; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }

            *foundBool = allFound;
            *compare = counter;
        }
    }
}

// Check for direction South-East
void solveSE (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if ((pCol < (Puzzle.col - word.length + 1)) && (pRow < (Puzzle.row - word.length + 1))) {
            int i = pRow+1, j = pCol+1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
```

```
                }
                i++; j++; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }

            *foundBool = allFound;
            *compare = counter;
        }
    }
}

// Check for direction South
void solveS (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if (pRow < (Puzzle.row - word.length + 1)) {
            int i = pRow+1, j = pCol, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
                i++; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }

            *foundBool = allFound;
            *compare = counter;
        }
    }
}
```

```
// Check for direction South-West
void solveSW (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if ((pCol >= (word.length - 1)) && (pRow < (Puzzle.row - word.length + 1))) {
            int i = pRow+1, j = pCol-1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
                i++; j--; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }

            *foundBool = allFound;
            *compare = counter;
        }
    }
}

// Check for direction West
void solveW (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if (pCol >= (word.length - 1)) {
            int i = pRow, j = pCol-1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
```

```
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
                j--; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }

            *foundBool = allFound;
            *compare = counter;
        }
    }
}

// Check for direction North-West
void solveNW (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if ((pCol >= (word.length - 1)) && (pRow >= (word.length - 1))) {
            int i = pRow-1, j = pCol-1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
                i--; j--; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }
```

```
            *foundBool = allFound;
            *compare = counter;
        }
    }
}

// Check for direction North
void solveN (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if (pRow >= (word.length - 1)) {
            int i = pRow-1, j = pCol, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
                i--; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }

            *foundBool = allFound;
            *compare = counter;
        }
    }
}

// Check for direction North-East
void solveNE (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
```

```
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if ((pCol < (Puzzle.col - word.length + 1)) && (pRow >= (word.length - 1))) {
            int i = pRow-1, j = pCol+1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
                i--; j++; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }

            *foundBool = allFound;
            *compare = counter;
        }
    }
}

int main() {
    char repeatprogram = 'Y';
    while (repeatprogram == 'Y') {
        clock_t time;
        int i = 0, j = 0, k = 0, l = 0;

        printf("====================================\n");
        printf("WELCOME TO WORD SEARCH PUZZLE SOLVER\n");
        printf("by 13520023 - Ahmad Alfani Handoyo\n");
        printf("====================================\n\n");


        // Read filename of puzzle
        char directory[207] = "../test/";
        char filename[200];
        printf("Insert filename of puzzle (including the file format): ");
        scanf("%s", filename);
        strcat(directory,filename);


        // Read file
        char c, cPrev;
```

```
    int Row = 0, Col = 0, foundRow = 0;
    FILE *fp = fopen(directory, "r");

    if (fp == NULL) {
        printf("No such file with name %s is found!\n\n", filename);
    } else {
        printf("----------------------------\n");
        printf("     WORD SEARCH PUZZLE\n");
        printf("----------------------------\n");

        // Read rows and columns of puzzle matrix
        c = cPrev = getc(fp);
        while ((c != '\n') || (cPrev != '\n')) {
            if (c == '\n') {
                foundRow = 1;
                Row++;
            }
            if (!foundRow && c != ' ') {
                Col++;
            }
            cPrev = c;
            c = getc(fp);
        }
        fclose(fp);
        printf("%d ROWS and %d COLUMNS\n\n", Row, Col);

        // Read puzzle matrix
        Matrix Puzzle;
        Puzzle.row = Row; Puzzle.col = Col;

        fp = fopen(directory, "r");
        c = cPrev = getc(fp);
        while ((c != '\n') || (cPrev != '\n')) {
            if (c != '\n' && c != ' ') {
                Puzzle.matrix[i][j] = c;
                j++;
                if (j == Col) {
                    j = 0;
                    i++;
                }
            }
            cPrev = c;
            c = getc(fp);
        }

        // Print puzzle
        PrintMatrix(Puzzle);
```

```
        // Read words
        int wordcount = 0, wordlength = 0;
        Word words[30];
        c = getc(fp);
        while (c != EOF) {
            if (c == '\n') {
                words[wordcount].length = wordlength;
                wordcount++;
                wordlength = 0;
            } else {
                words[wordcount].word[wordlength] = c;
                wordlength++;
            }
            c = getc(fp);
        }
        words[wordcount].length = wordlength;
        wordcount++;
        fclose(fp);

        // Print words
        printf("\nKEYWORDS:\n");
        for (i=0; i < wordcount; i++) {
            printf("%d. ", i+1);
            for (j = 0; j < words[i].length; j++) {
                printf("%c",words[i].word[j]);
            }
            printf("\n");
        }
        printf("\n");

        // Brute-Force Algorithm
        printf("----------------------------\n");
        printf("         SOLUTIONS\n");
        printf("----------------------------\n");
        time = clock();
        int found, firstFound, compare = 0;
        for (i=0; i < wordcount; i++) {
            printf("%d. ", i+1);
            for (l = 0; l < words[i].length; l++) {
                printf("%c",words[i].word[l]);
            }
            printf("\n");

            found = 0;
            j = 0;
            while (j < Row && !found) {
                k = 0;
                while (k < Col && !found) {
```

```
                    solveFirst(Puzzle, j, k, words[i], &compare, &firstFound);
                    if (firstFound) {
                        solveE(Puzzle, j, k, words[i], &found, &compare);
                        solveSE(Puzzle, j, k, words[i], &found, &compare);
                        solveS(Puzzle, j, k, words[i], &found, &compare);
                        solveSW(Puzzle, j, k, words[i], &found, &compare);
                        solveW(Puzzle, j, k, words[i], &found, &compare);
                        solveNW(Puzzle, j, k, words[i], &found, &compare);
                        solveN(Puzzle, j, k, words[i], &found, &compare);
                        solveNE(Puzzle, j, k, words[i], &found, &compare);
                    }
                    k++;
                }
                j++;
            }
            if (!found) {
                printf("Word is not found in the puzzle\n\n");
            }
        }
        time = clock() - time;
        int time_consumed = (((double) time)/CLOCKS_PER_SEC)*1000;
        printf("----------------------------\n");
        printf("         STATISTICS\n");
        printf("----------------------------\n");
        printf("TOTAL LETTER COMPARISON: %d LETTERS\n", compare);
        printf("TOTAL TIME TAKEN: %d miliseconds\n\n", time_consumed);
    }

    printf("Do you want to solve another puzzle? (Y/N): ");
    scanf(" %c", &repeatprogram);
  }
  return 0;
}
```

C. *Screenshot* Hasil *Input* dan *Output* Program

Mesin yang digunakan untuk menguji coba program berjalan pada Windows 11 build 22000.434. Mesin juga mempunyai spesifikasi prosesor AMD Ryzen 7 4800H dengan cache L1 512KB, L2 4MB, dan L3 8MB dan RAM teralokasi 23,4GB.

1. Ukuran *small*

   small1.txt (16x15, 11 kata)

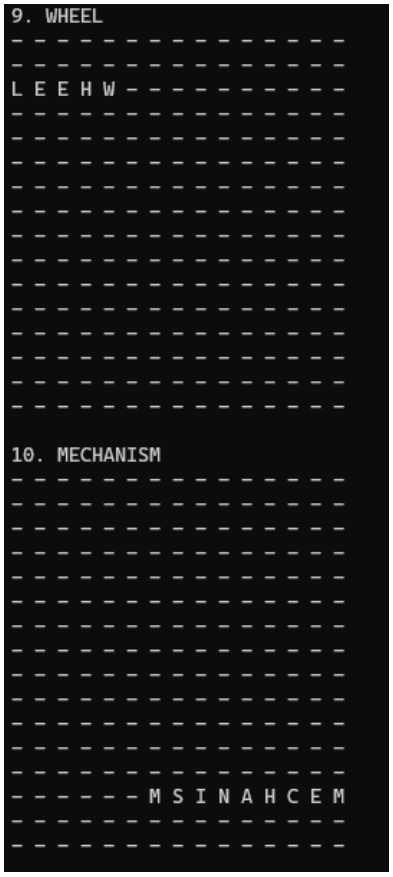
```
==================================
WELCOME TO WORD SEARCH PUZZLE SOLVER
by 13520023 - Ahmad Alfani Handoyo
==================================

Insert filename of puzzle (including the file format): small1.txt
----------------------------
     WORD SEARCH PUZZLE
----------------------------
16 ROWS and 15 COLUMNS

V F C H A S E N J J W D W I T
O G P G D C L O D V E H F D H
L E E H W S Z W E Z F S D Z S
T M D U J U F K S Z P A I R S
N I T E L L U B K W N E Z C I
S I M P L I C I T Y C L J N U
M K F T O L E R A N T D T W R
Y C S O B J E K Z D C E M V A
A G S N P C R F C K L U Q P B
T Z B O J N O L F L T C B J L
R N O N W R N O I T U L O V E
R L A N G U A G E W X N F X L
H D I R J G E B G S A S J O L
N C A Z R N M S I N A H C E M
E O B Z C A P F F T G V S Q A
O K N E P I W R Z L E V U J H

KEYWORDS:
1. SIMPLICITY
2. LANGUAGE
3. CHASE
4. WARRANT
5. EVOLUTION
6. LEASH
7. BULLETIN
8. TOLERANT
9. WHEEL
10. MECHANISM
11. INTELLIGENCE
```

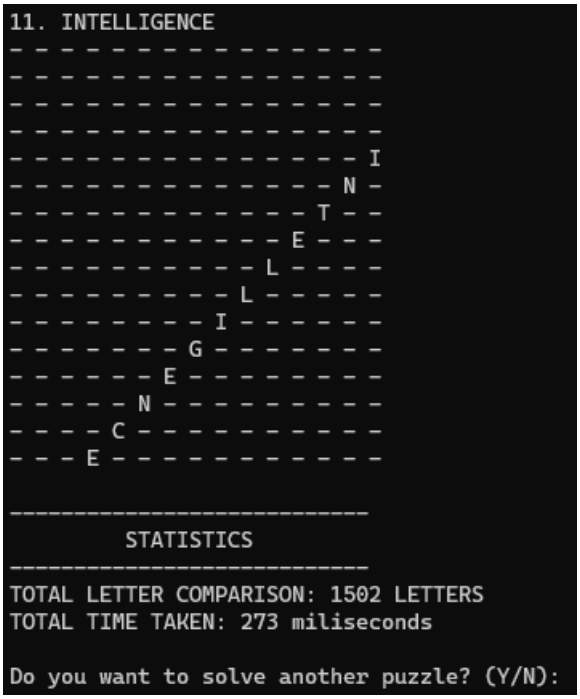
```
----------------------------
         SOLUTIONS
----------------------------
1. SIMPLICITY
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
S I M P L I C I T Y - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -

2. LANGUAGE
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- L A N G U A G E - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
```

```
3. CHASE
- - C H A S E - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -

4. WARRANT
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
T - - - - - - - - - - - - - -
- N - - - - - - - - - - - - -
- - A - - - - - - - - - - - -
- - - R - - - - - - - - - - -
- - - - R - - - - - - - - - -
- - - - - A - - - - - - - - -
- - - - - - W - - - - - - - -
```

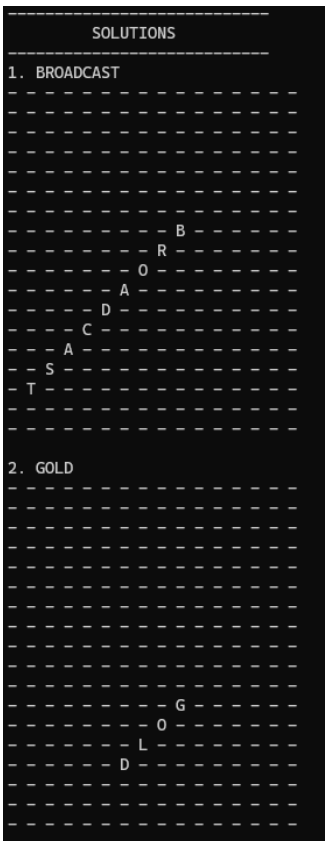
```
5. EVOLUTION
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - N O I T U L O V E
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -

6. LEASH
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - H - -
- - - - - - - - - - - - - S - -
- - - - - - - - - - - - - A - -
- - - - - - - - - - - - - E - -
- - - - - - - - - - - - - L - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
```

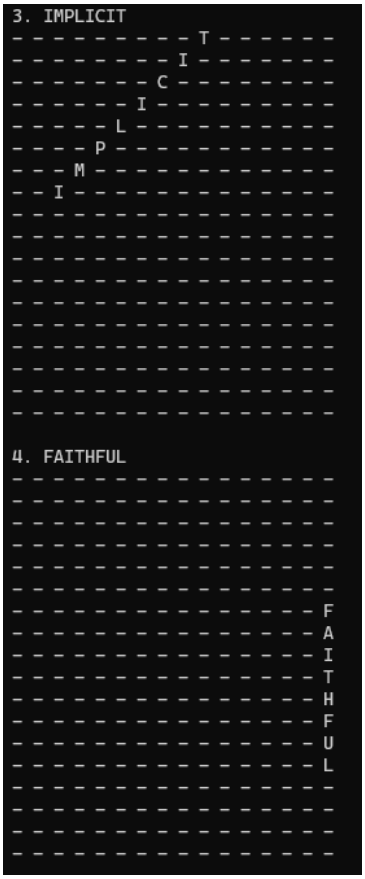
```
7. BULLETIN
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
N I T E L L U B - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -

8. TOLERANT
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - T O L E R A N T - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
```

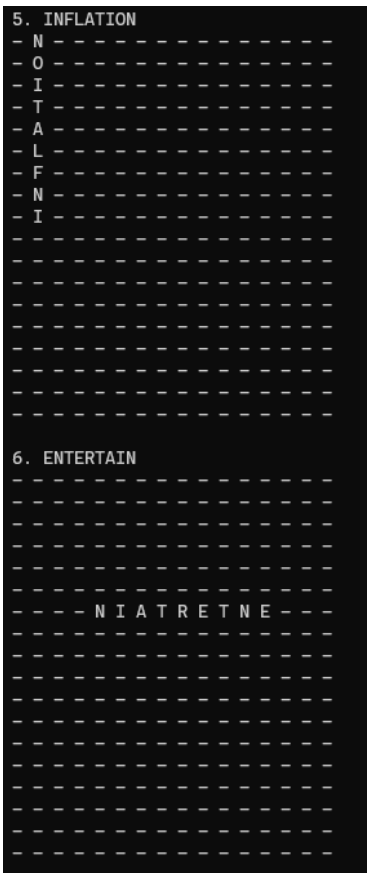


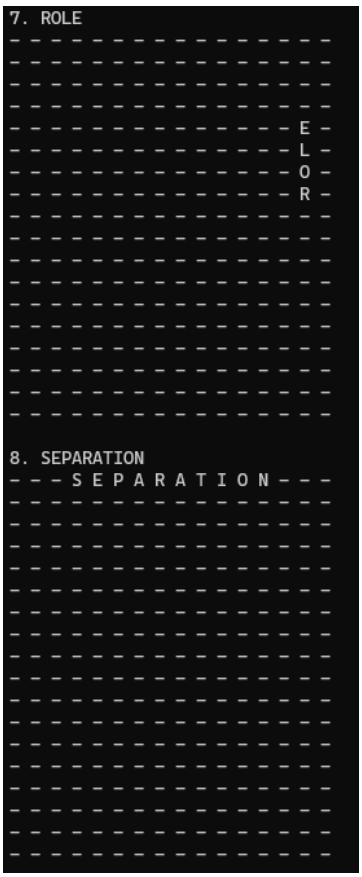


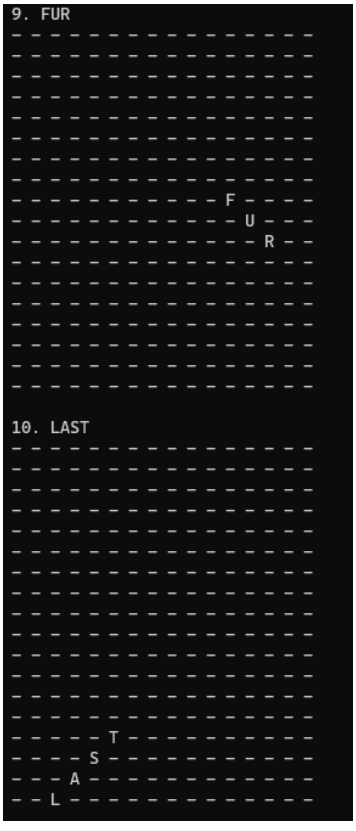
small2.txt (18x16, 13 kata)

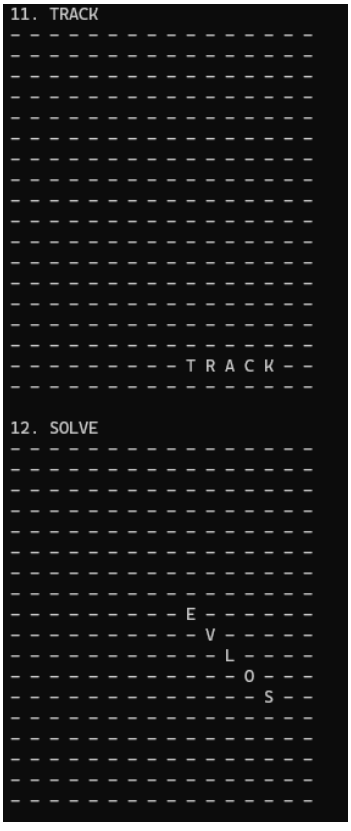


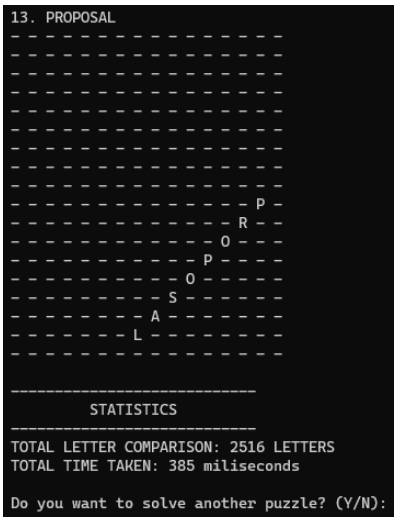

3. IMPLICIT
4. FAITHFUL


5. INFLATION
6. ENTERTAIN


7. ROLE
8. SEPARATION


9. FUR
10. LAST


11. TRACK
12. SOLVE


13. PROPOSAL
STATISTICS
TOTAL LETTER COMPARISON: 2516 LETTERS
TOTAL TIME TAKEN: 385 miliseconds
Do you want to solve another puzzle? (Y/N):

small3.txt (15x15, 10 kata)

```
==================================
WELCOME TO WORD SEARCH PUZZLE SOLVER
by 13520023 - Ahmad Alfani Handoyo
==================================

Insert filename of puzzle (including the file format): small3.txt
---------------------------
     WORD SEARCH PUZZLE
---------------------------
15 ROWS and 15 COLUMNS

H E H P M C H S D G V K J D E
U F C K T U H K H V W E H E X
V H N E D Q E O N Y K K N G P
H Q C M U R H A C C P C N R E
T C E P X E D T T O E V N A R
K Q R Q I V W E A M L L B H I
I X D M O K U B T W Z A Y C M
T R N W Z I P U D E V Z T R E
N O I T A L U M U C C A X E N
Y E X A J D K L Q Q X T E V T
I E C N E R E F R E T N I O D
D X T N E M T N I O P P A V H
G V B A A D V E R T I S E S E
D S Y D J E C N A W O L L A Q
C A A S P V T E K G A M P D R

KEYWORDS:
1. ALLOWANCE
2. DETECTIVE
3. OVERCHARGE
4. ADVERTISE
5. APPOINTMENT
6. EXPERIMENT
7. CHOCOLATE
8. EXPECT
9. INTERFERENCE
10. ACCUMULATION
```

```
---------------------------
        SOLUTIONS
---------------------------
1. ALLOWANCE
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - E C N A W O L L A -
- - - - - - - - - - - - - - -

2. DETECTIVE
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - D - - - - - - -
- - - - - - - E - - - - - - -
- - - - - - - - T - - - - - -
- - - - - - - - - E - - - - -
- - - - - - - - - - C - - - -
- - - - - - - - - - - T - - -
- - - - - - - - - - - - I - -
- - - - - - - - - - - - - V -
- - - - - - - - - - - - - - E
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
```

```
3. OVERCHARGE
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - E -
- - - - - - - - - - - - - G -
- - - - - - - - - - - - - R -
- - - - - - - - - - - - - A -
- - - - - - - - - - - - - H -
- - - - - - - - - - - - - C -
- - - - - - - - - - - - - R -
- - - - - - - - - - - - - E -
- - - - - - - - - - - - - V -
- - - - - - - - - - - - - O -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -

4. ADVERTISE
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - A D V E R T I S E - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
```

```
5. APPOINTMENT
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - T N E M T N I O P P A - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -

6. EXPERIMENT
- - - - - - - - - - - - - - E
- - - - - - - - - - - - - - X
- - - - - - - - - - - - - - P
- - - - - - - - - - - - - - E
- - - - - - - - - - - - - - R
- - - - - - - - - - - - - - I
- - - - - - - - - - - - - - M
- - - - - - - - - - - - - - E
- - - - - - - - - - - - - - N
- - - - - - - - - - - - - - T
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
```

```
7. CHOCOLATE
- - - - - - C - - - - - - - -
- - - - - - H - - - - - - - -
- - - - - - - O - - - - - - -
- - - - - - - - C - - - - - -
- - - - - - - - - O - - - - -
- - - - - - - - - - L - - - -
- - - - - - - - - - - A - - -
- - - - - - - - - - - - T - -
- - - - - - - - - - - - - E -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -

8. EXPECT
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
T C E P X E - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
```

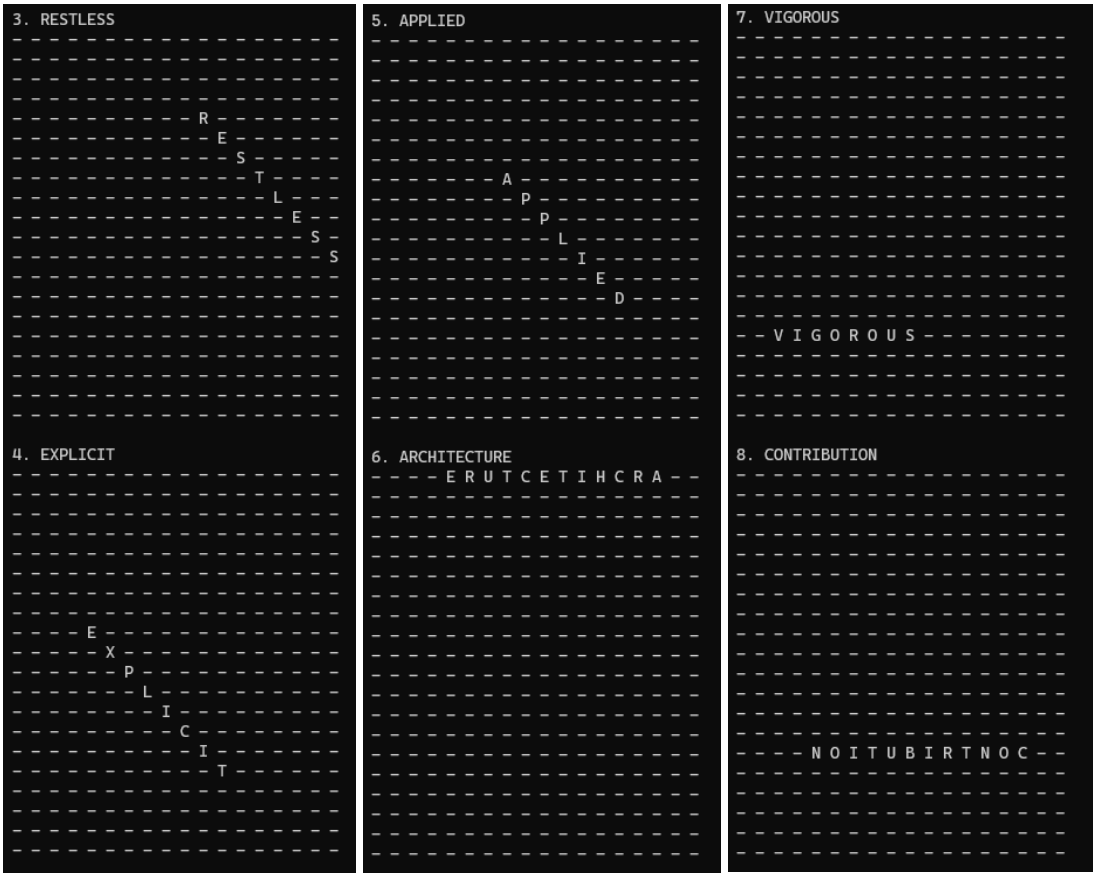
```
9. INTERFERENCE
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- E C N E R E F R E T N I - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -

10. ACCUMULATION
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
N O I T A L U M U C C A - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - -

----------------------------
         STATISTICS
----------------------------
TOTAL LETTER COMPARISON: 1401 LETTERS
TOTAL TIME TAKEN: 233 miliseconds

Do you want to solve another puzzle? (Y/N):
```

2. Ukuran *medium*

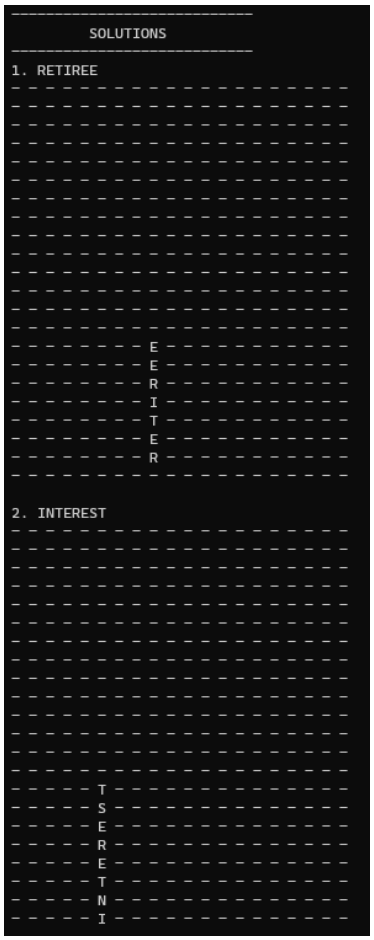
   medium1.txt (20x18, 12 kata)

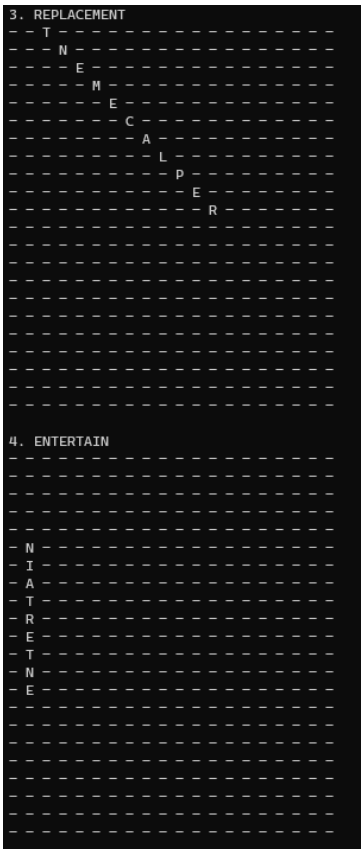
```
==================================
WELCOME TO WORD SEARCH PUZZLE SOLVER
by 13520023 - Ahmad Alfani Handoyo
==================================

Insert filename of puzzle (including the file format): medium1.txt
----------------------------
     WORD SEARCH PUZZLE
----------------------------
20 ROWS and 18 COLUMNS

Q R K B E R U T C E T I H C R A T F
C N M Y M C H Z L K F J A B J V I X
Q K M W U N N D D A Q O N M R X T V
N Y B Y A E A Z N G W J O T X J I O
H O A N T R E N P A R A L Y Z E D K
H Q O P T G I Y N Q T E M E C A F R
X R O V T S F W G O T S S W D C G L
L V O H Z U Q A C A U V R T X C Y B
F T X Z E Z B Z P O B N U E L M P F
Z B H U V X V P L P Q Y C K D E X N
E Q T G N O P D O R L U L E E N S O
R W L U U F G L O T E I H L M G U S
V H I X O O H Q I X E B E K T E G E
H N F I U H H K Q C Q O L D V K N X
F F W R N O I T U B I R T N O C N T
I C V I G O R O U S O T D S X P N V
Z N T G D U J Q G H G V B N H K M H
F H G H Q H H I P R K X Y E Q Y K H
E E T T E R A G I C Q X E Y X M N Y
G F U Y F O L K L O R E H H O M G I

KEYWORDS:
1. FOLKLORE
2. CIGARETTE
3. RESTLESS
4. EXPLICIT
5. APPLIED
6. ARCHITECTURE
7. VIGOROUS
8. CONTRIBUTION
9. ANNOUNCEMENT
10. THOUGHT
11. PARALYZED
12. UNDERSTAND
```

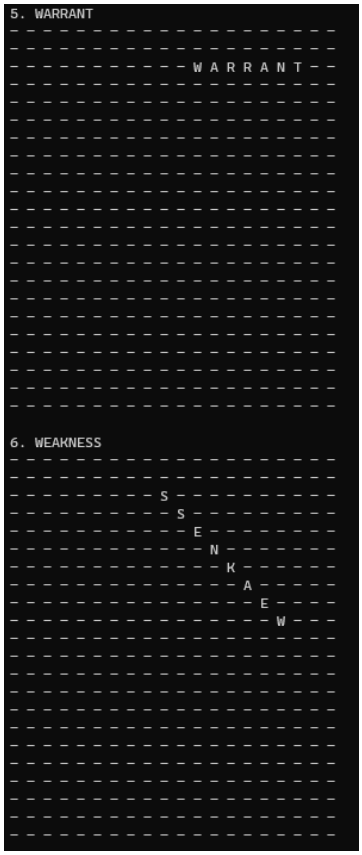
```
----------------------------
         SOLUTIONS
----------------------------
1. FOLKLORE
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - F O L K L O R E - - - - - -

2. CIGARETTE
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - -
- E T T E R A G I C - - - - - - - -
- - - - - - - - - - - - - - - - - - -
```

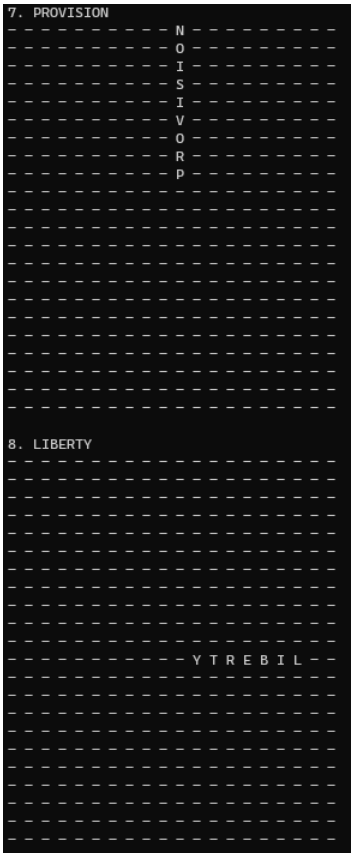

3. RESTLESS
4. EXPLICIT
5. APPLIED
6. ARCHITECTURE
E R U T C E T I H C R A
7. VIGOROUS
V I G O R O U S
8. CONTRIBUTION
N O I T U B I R T N O C
9. ANNOUNCEMENT
10. THOUGHT
11. PARALYZED
P A R A L Y Z E D
12. UNDERSTAND
STATISTICS
TOTAL LETTER COMPARISON: 2678 LETTERS
TOTAL TIME TAKEN: 446 miliseconds
Do you want to solve another puzzle? (Y/N):

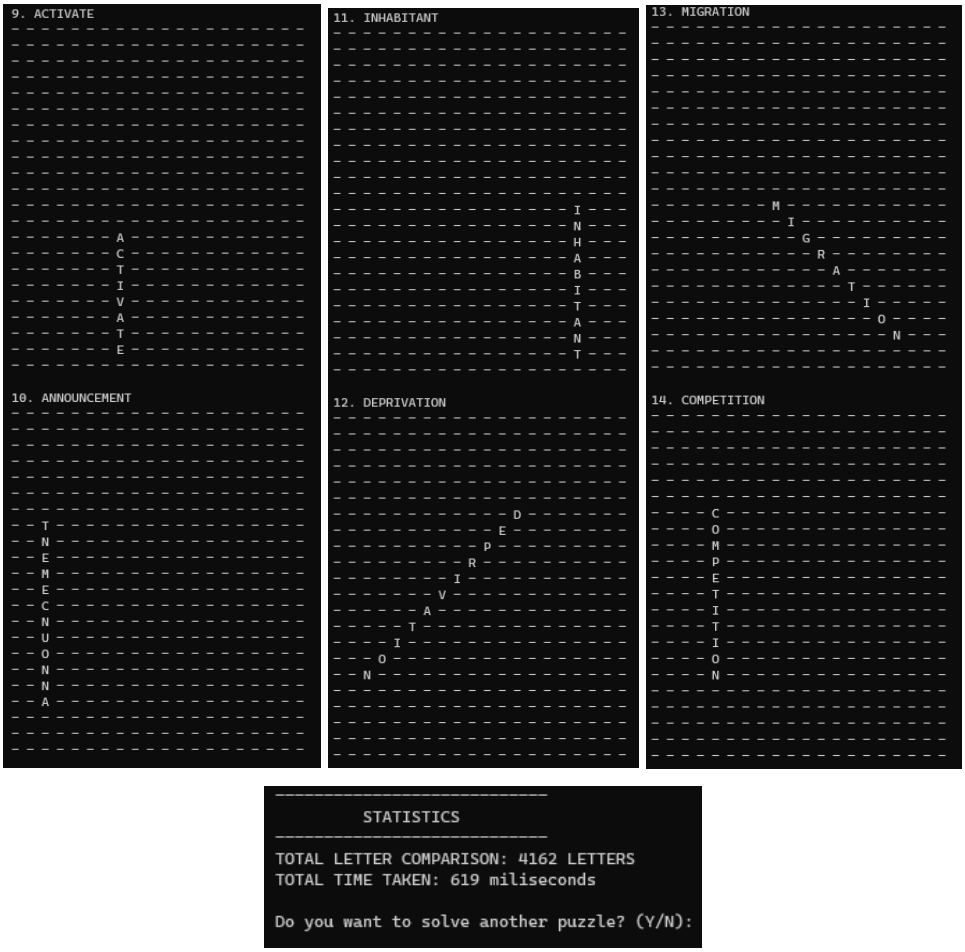
medium2.txt (22x20, 14 kata)



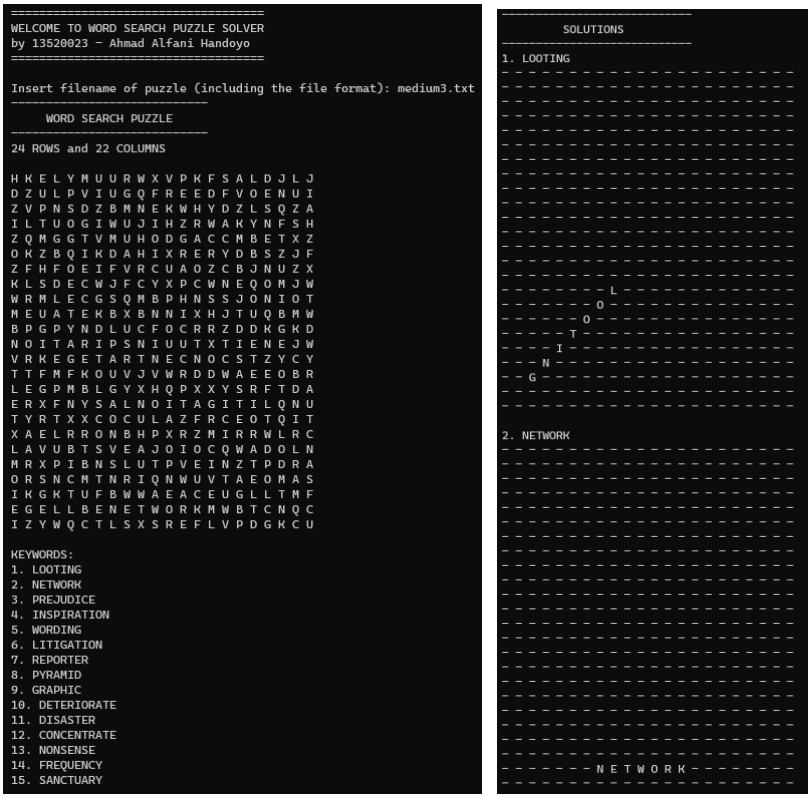


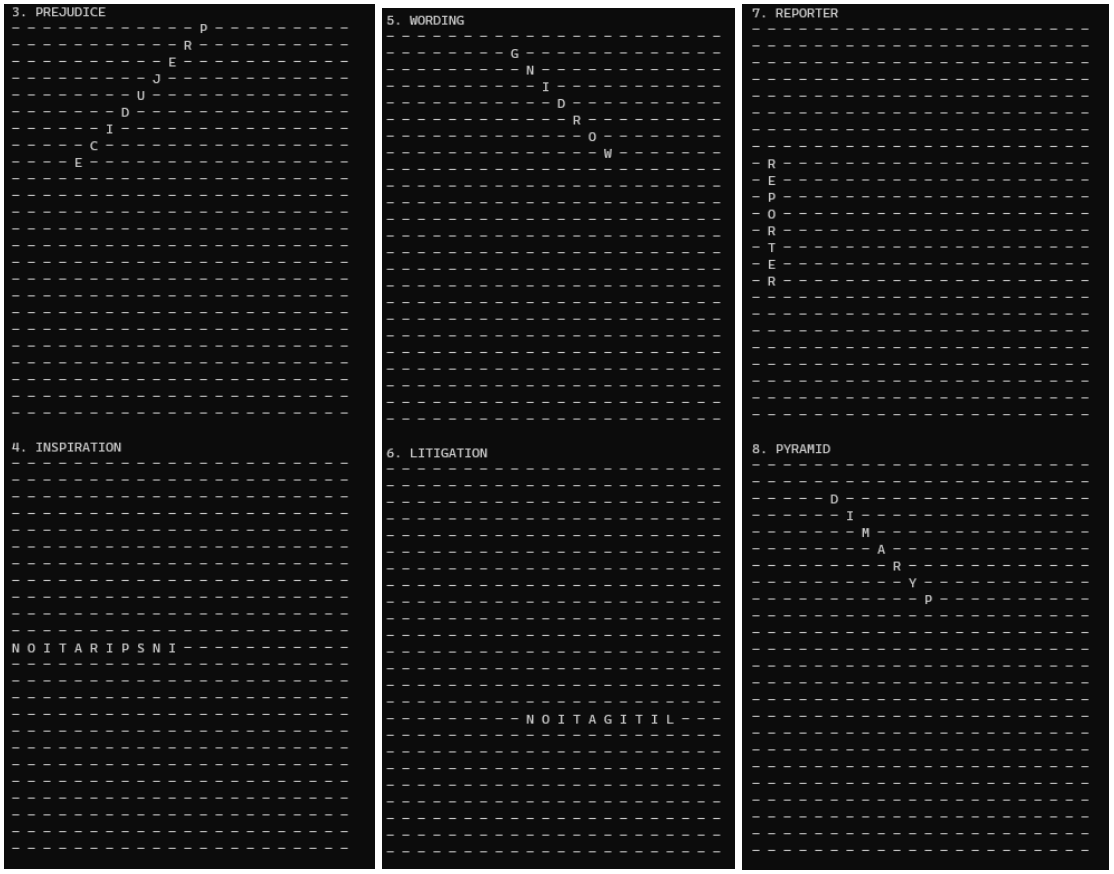


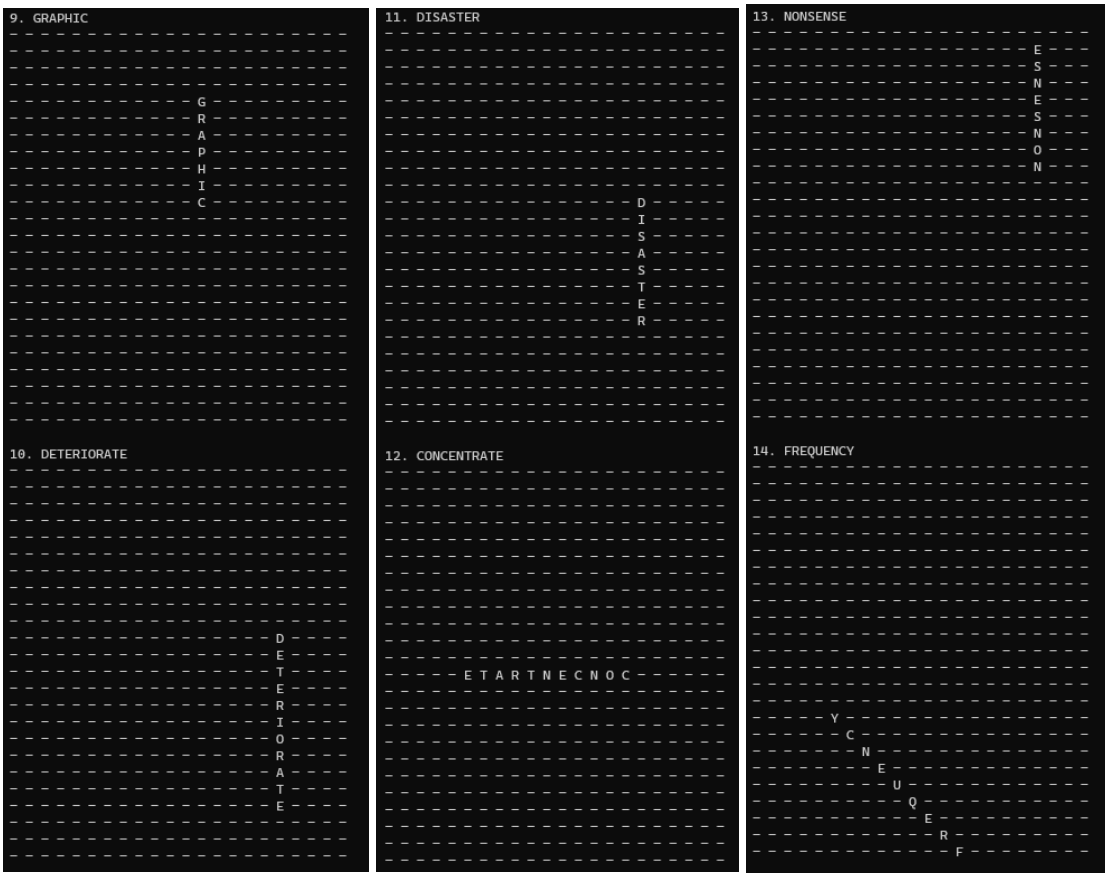


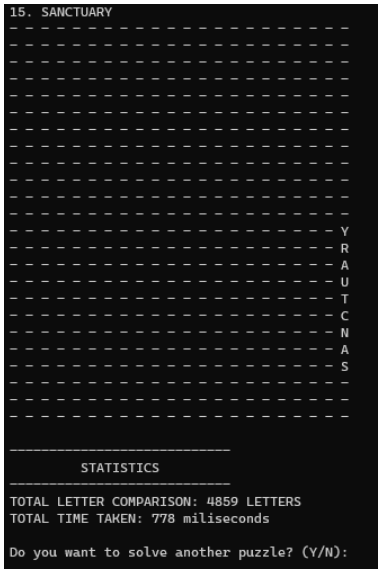


medium3.txt (24x22, 15 kata)



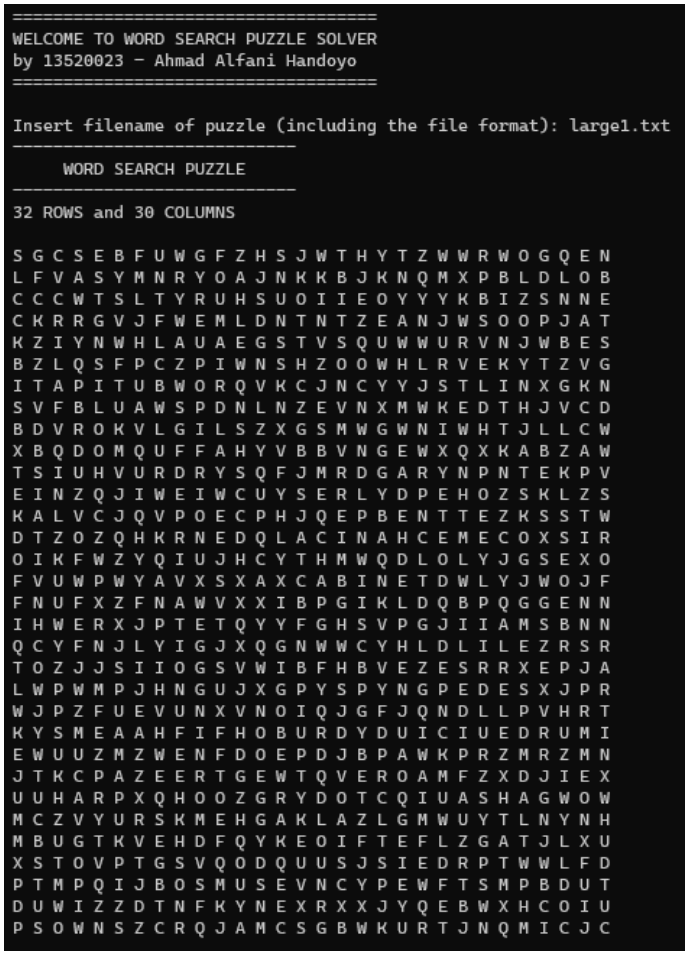

3. PREJUDICE
4. INSPIRATION
5. WORDING
6. LITIGATION
7. REPORTER
8. PYRAMID

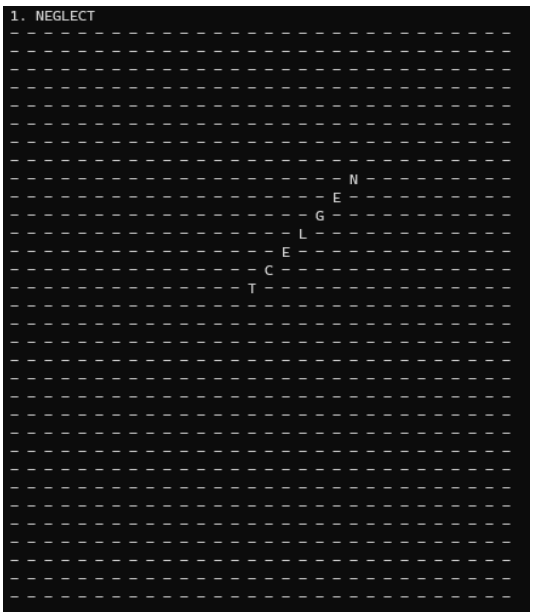

9. GRAPHIC
10. DETERIORATE
11. DISASTER
12. CONCENTRATE
13. NONSENSE
14. FREQUENCY

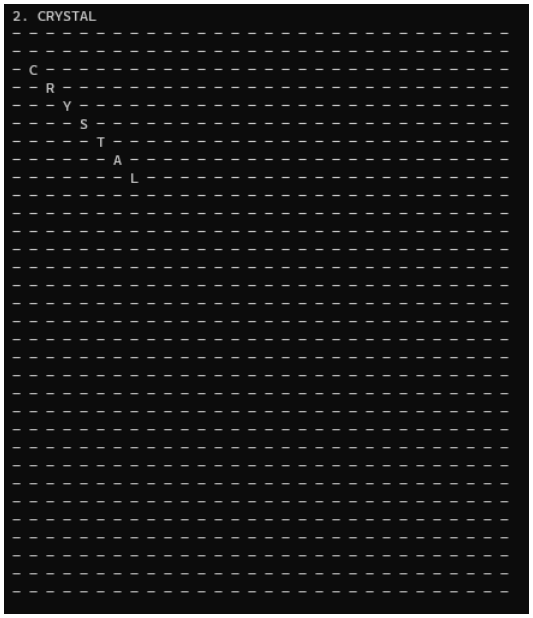


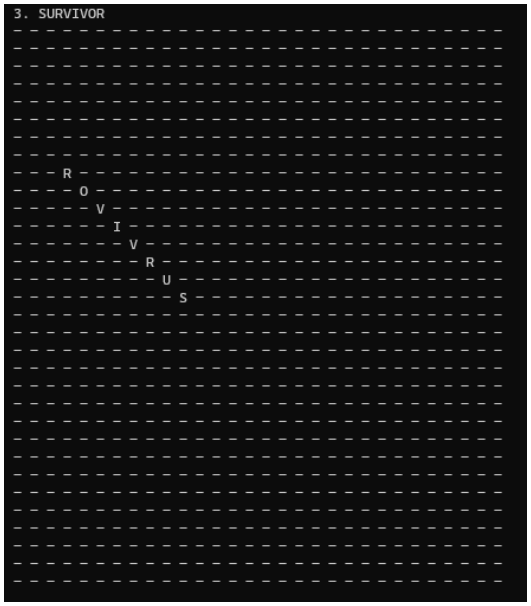
3. Ukuran *large*

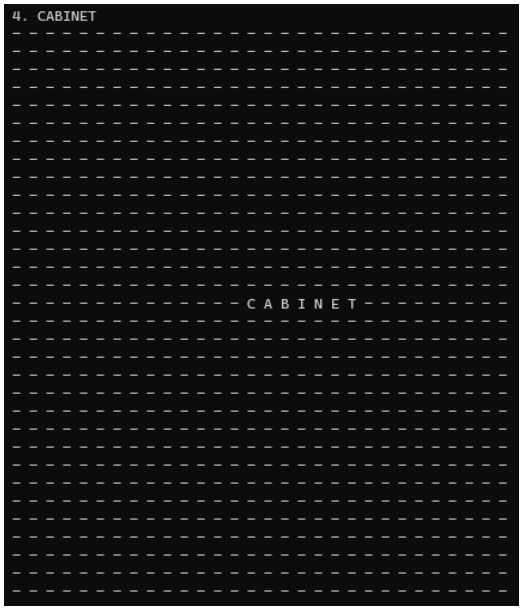
large1.txt (32x30, 13 kata)

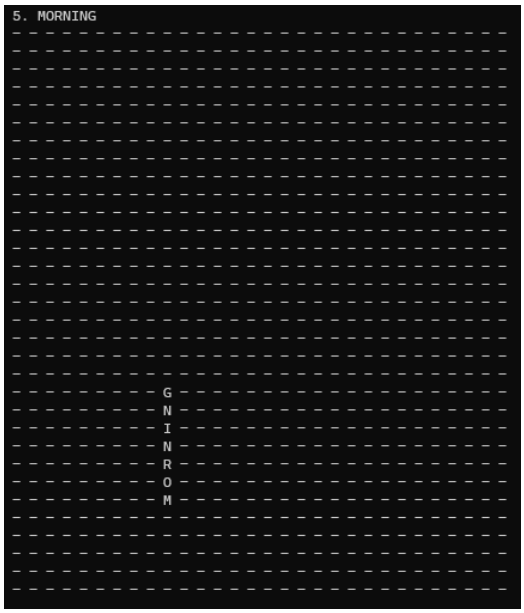


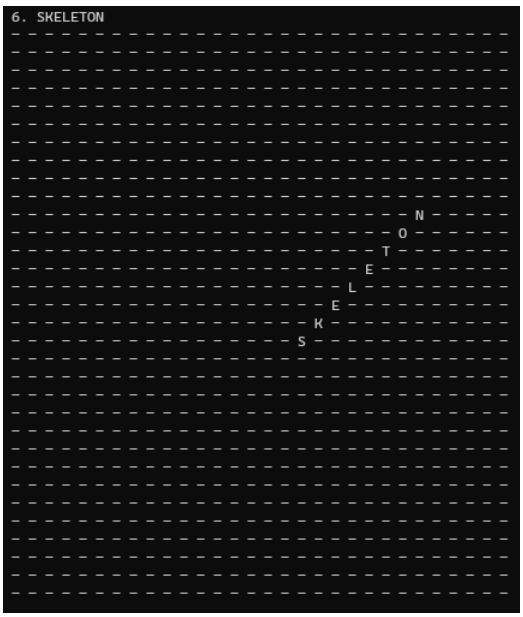

1. NEGLECT

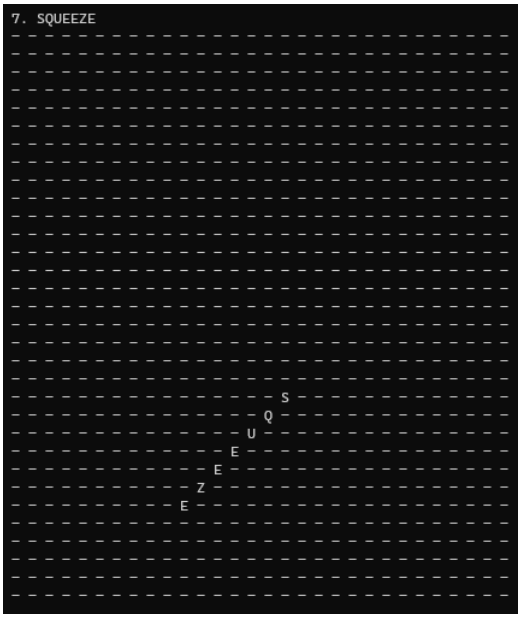

2. CRYSTAL

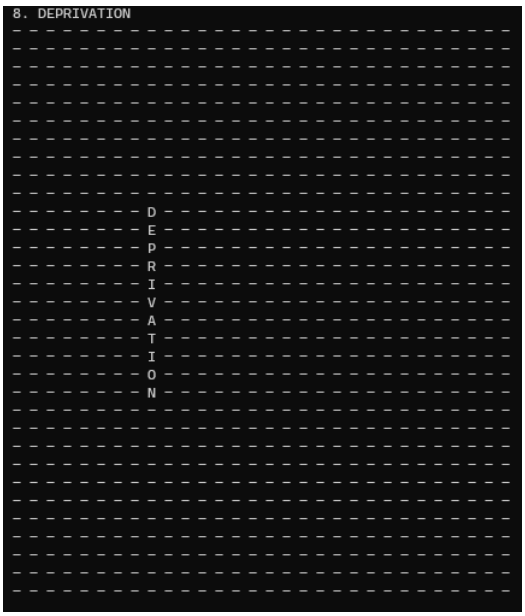

3. SURVIVOR

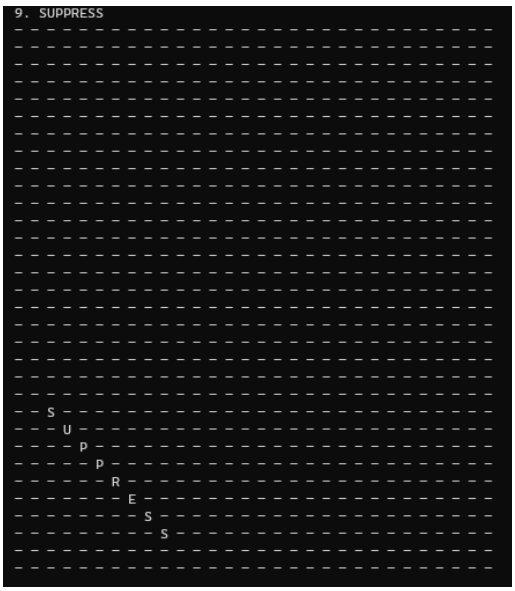

4. CABINET

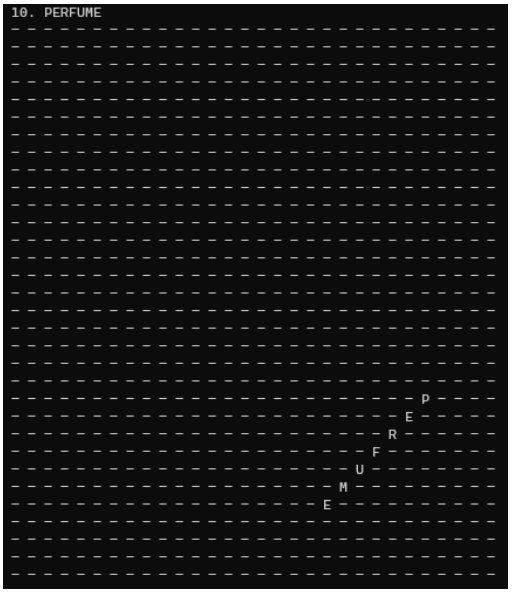

5. MORNING

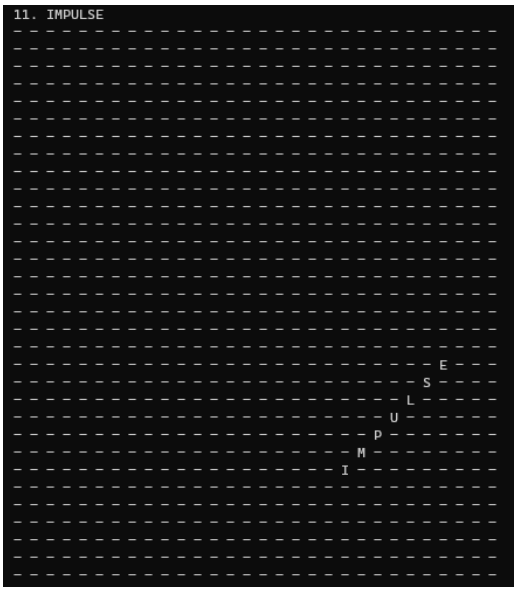

6. SKELETON

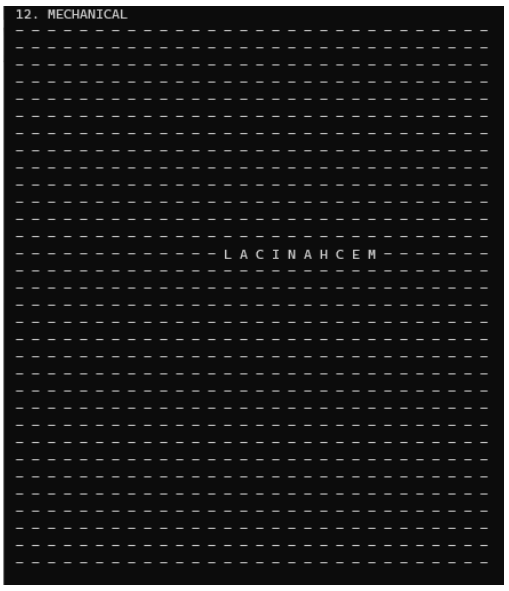

7. SQUEEZE

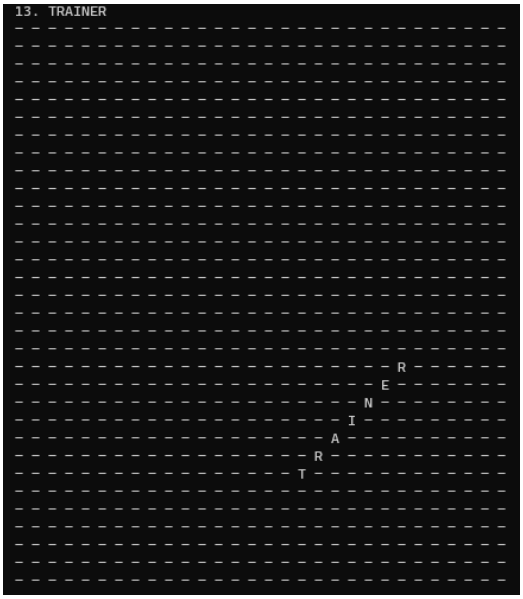

8. DEPRIVATION

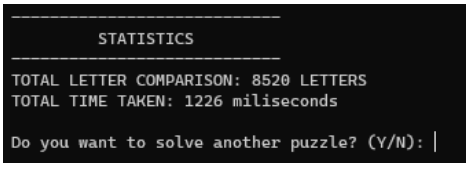

9. SUPPRESS


10. PERFUME

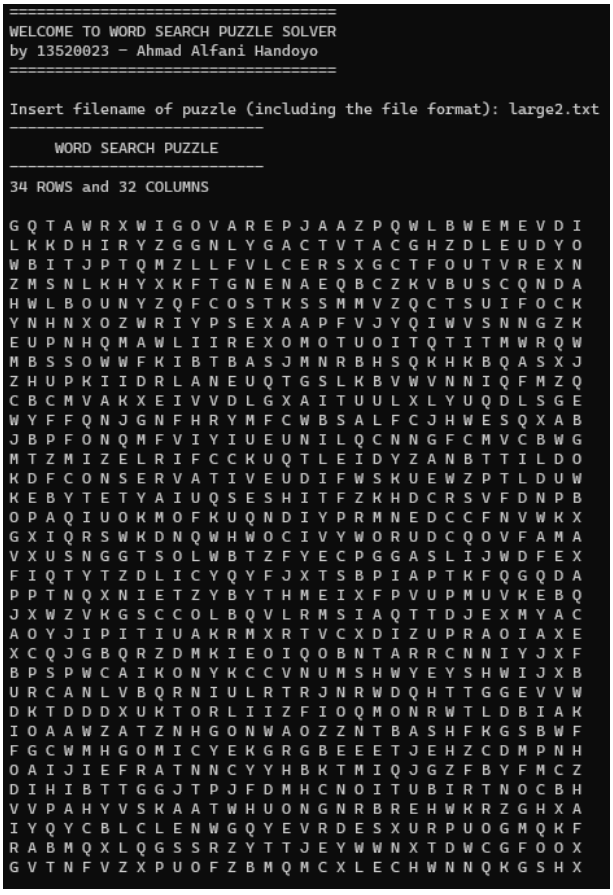

11. IMPULSE

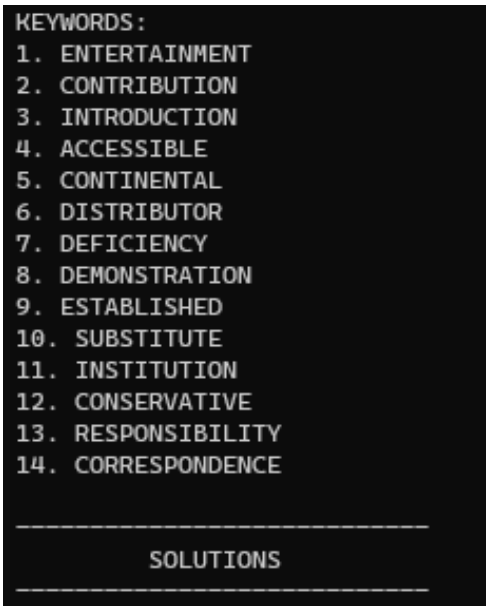

12. MECHANICAL

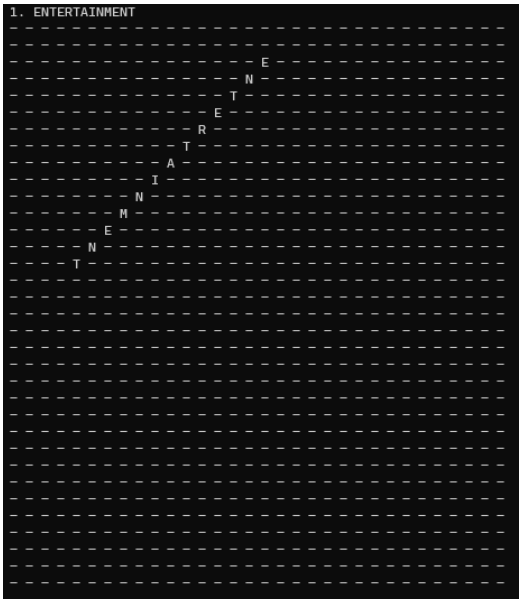


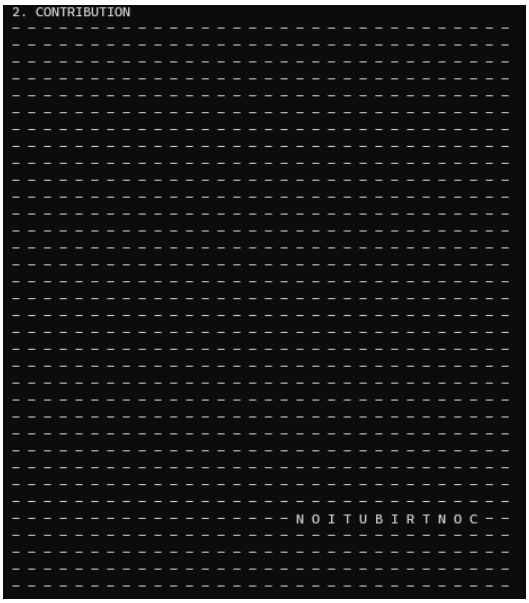


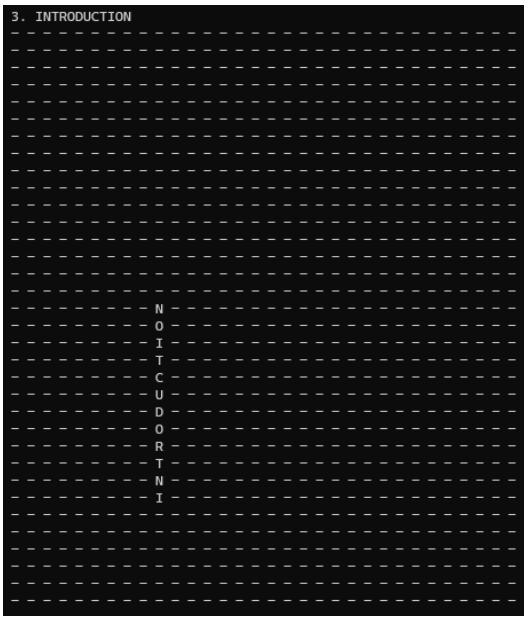
large2.txt (34x32, 14 kata)

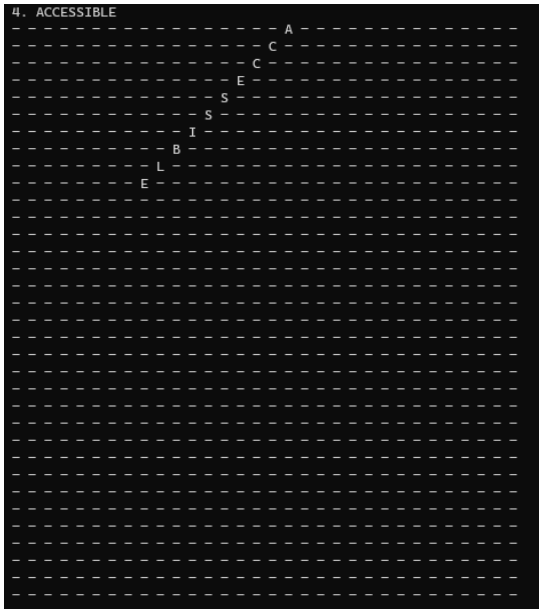


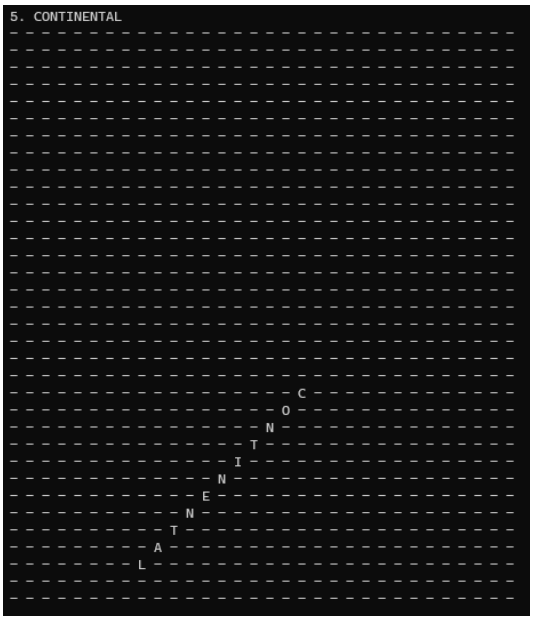


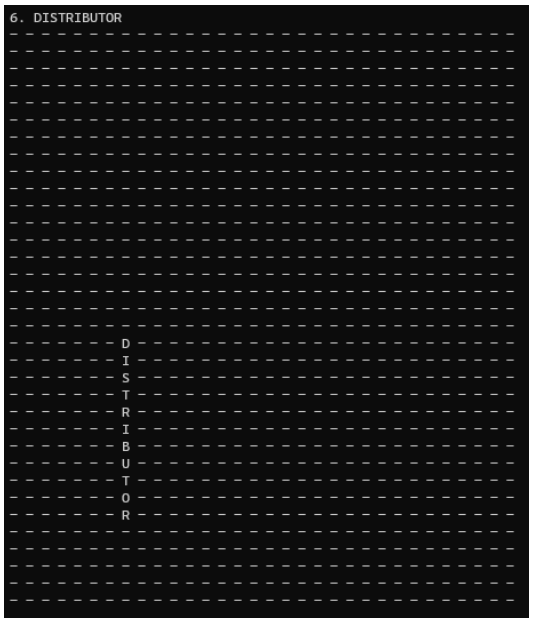

1. ENTERTAINMENT

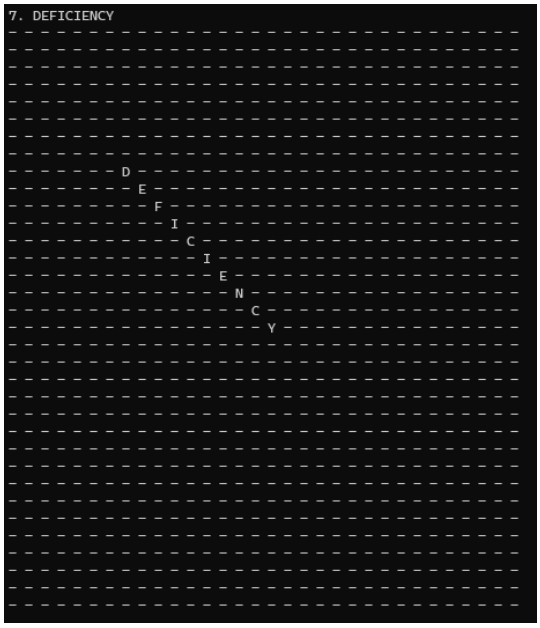

2. CONTRIBUTION
NOITUBIRTNOC

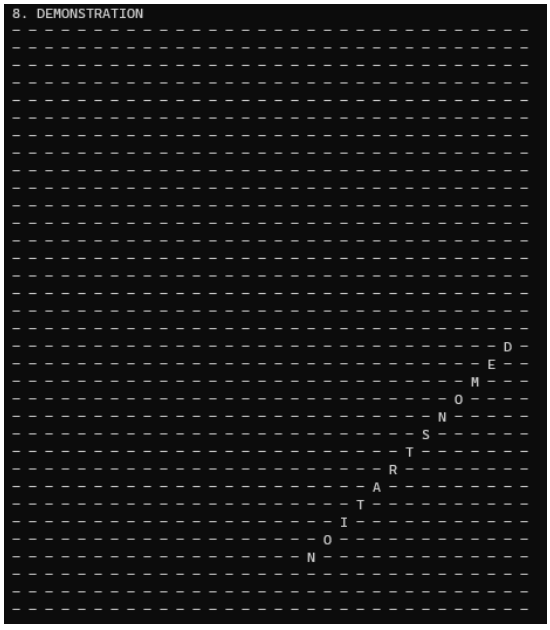

3. INTRODUCTION
NOITCUDORTNI

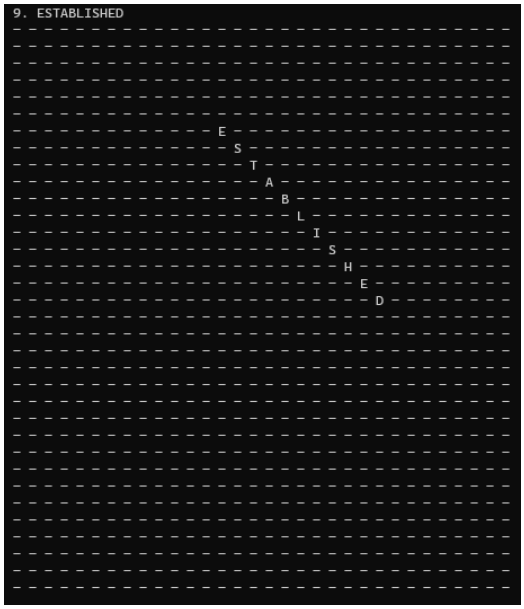

4. ACCESSIBLE
ACCESSIBLE

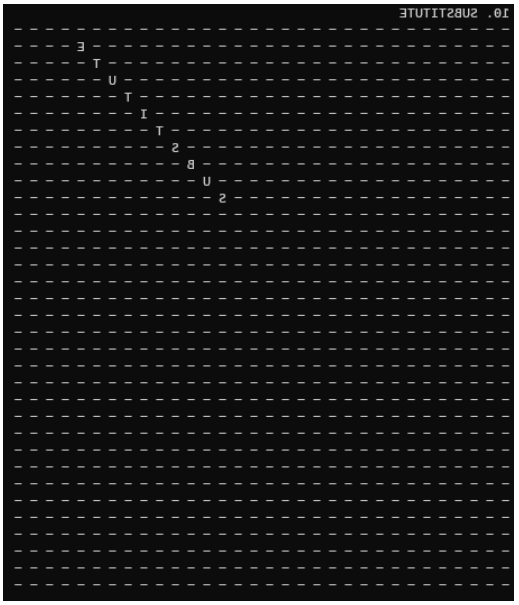

5. CONTINENTAL
CONTINENTAL

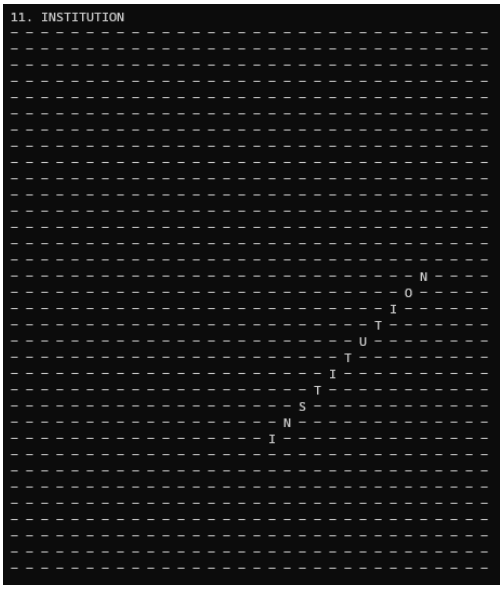

6. DISTRIBUTOR
DISTRIBUTOR

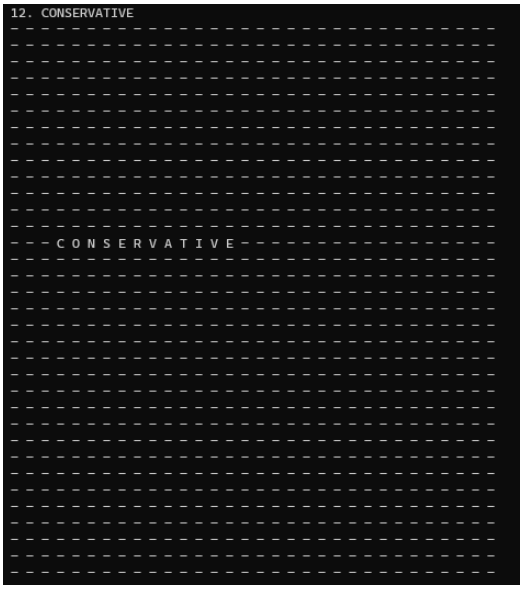

7. DEFICIENCY

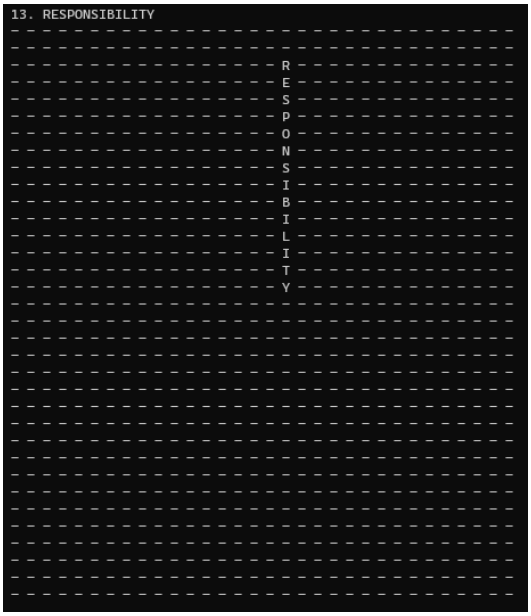

8. DEMONSTRATION

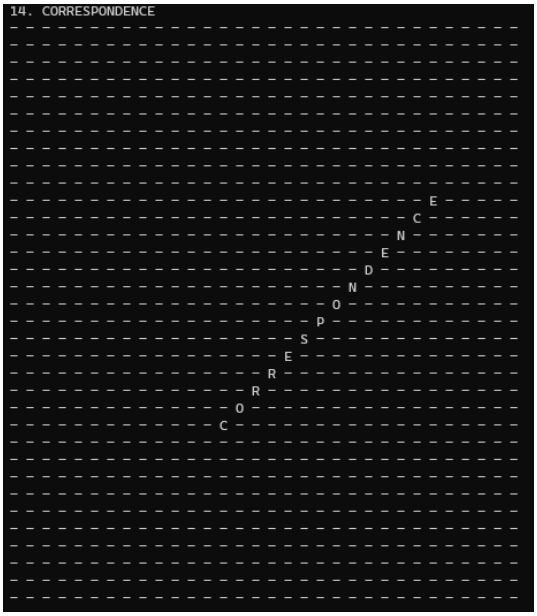

9. ESTABLISHED

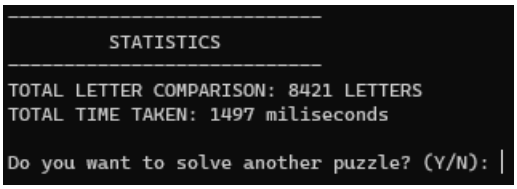

10. SUBSTITUTE


11. INSTITUTION

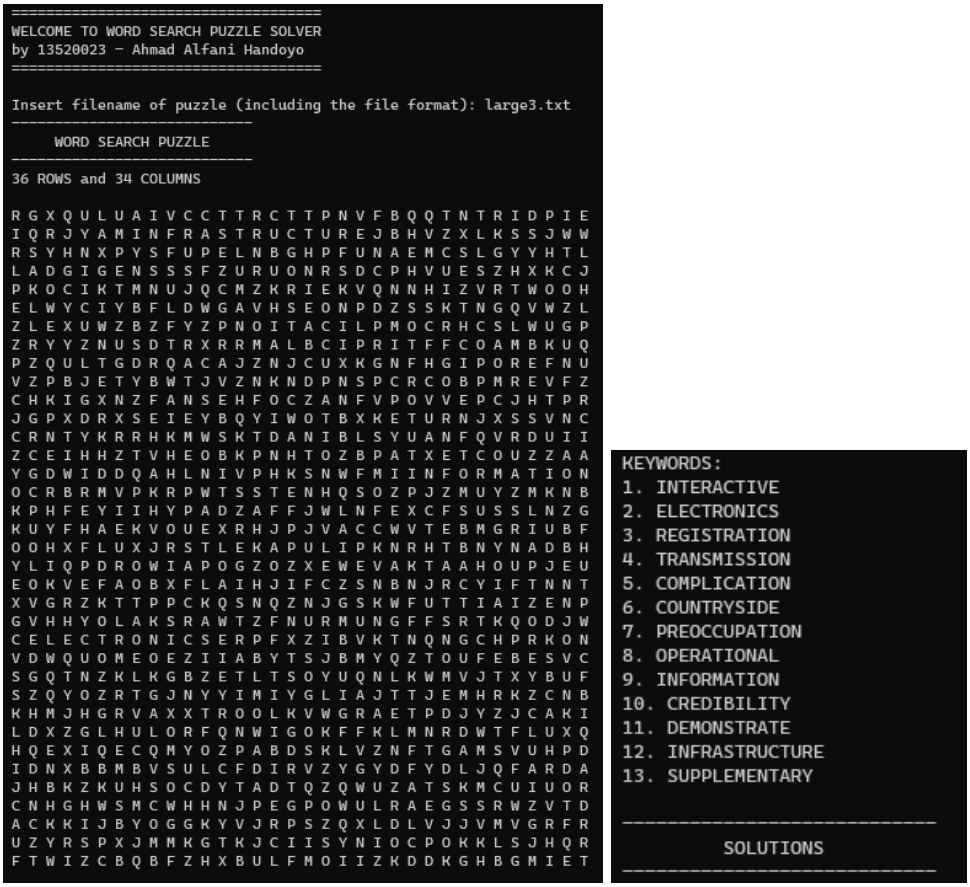

12. CONSERVATIVE
CONSERVATIVE

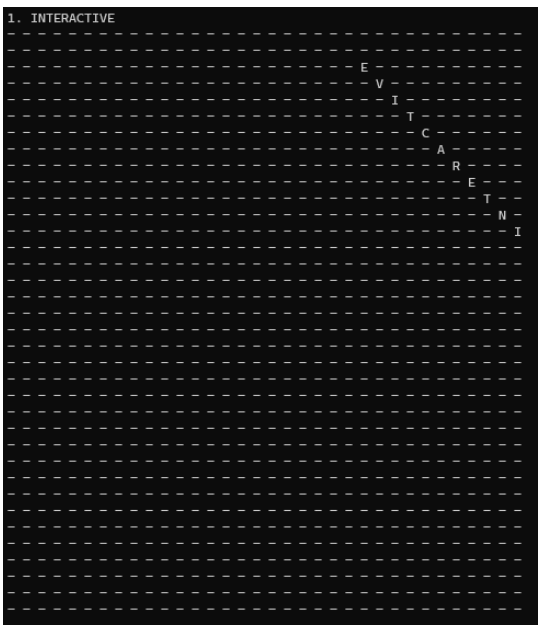


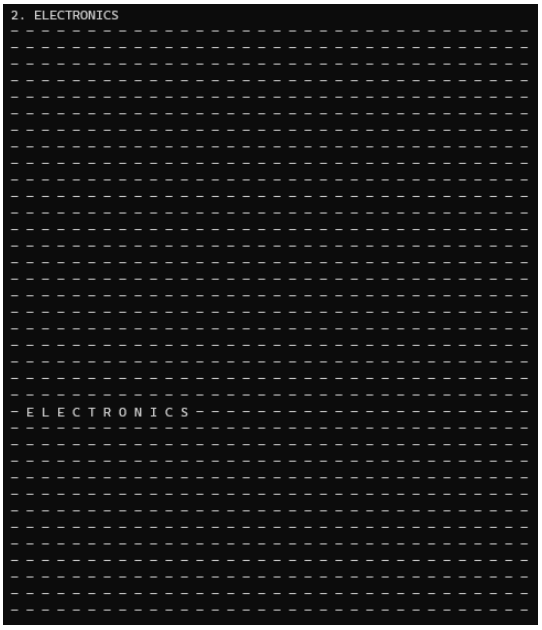


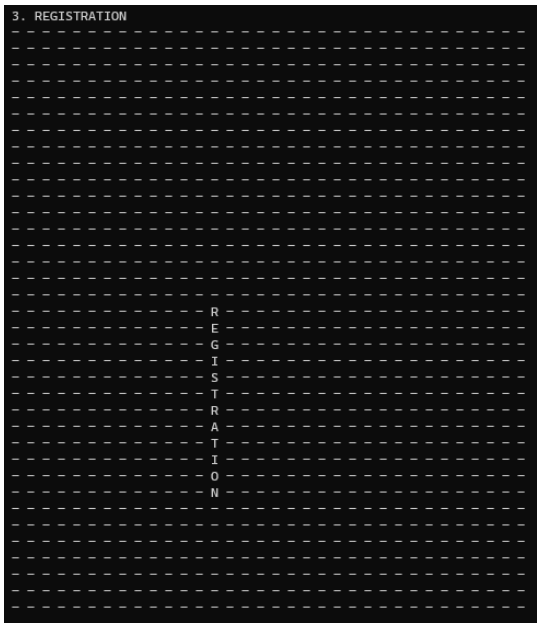


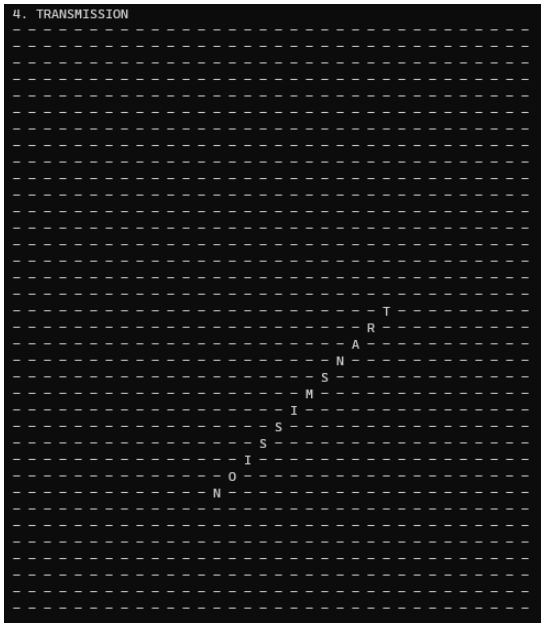
large3.txt (36x34, 13 kata)

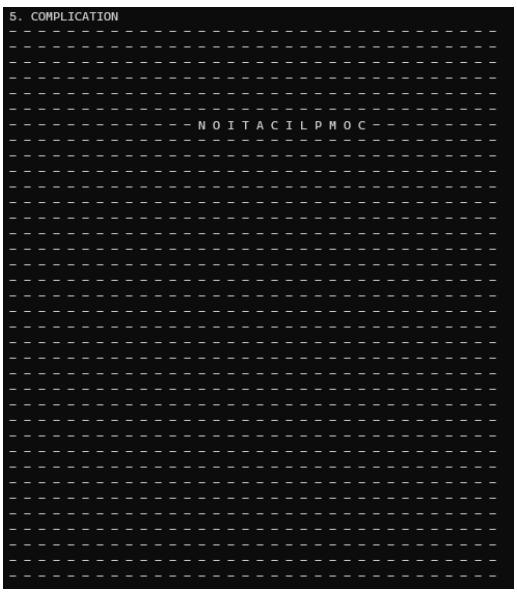


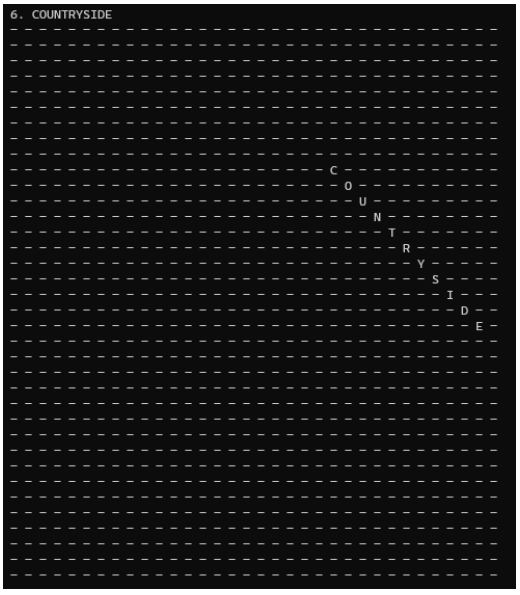

1. INTERACTIVE

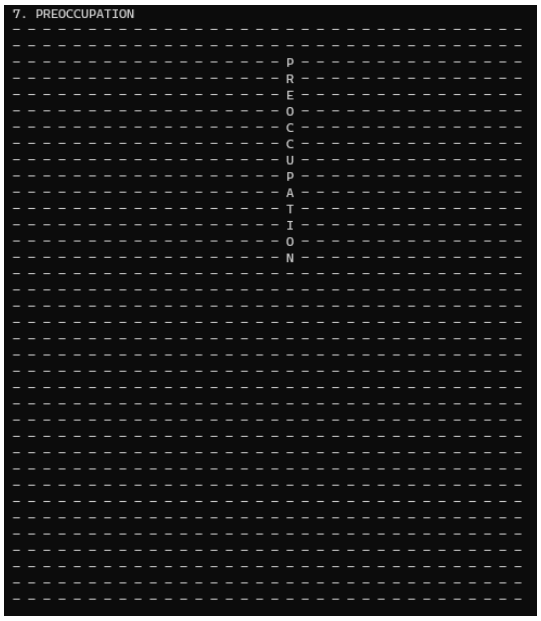

2. ELECTRONICS
ELECTRONICS

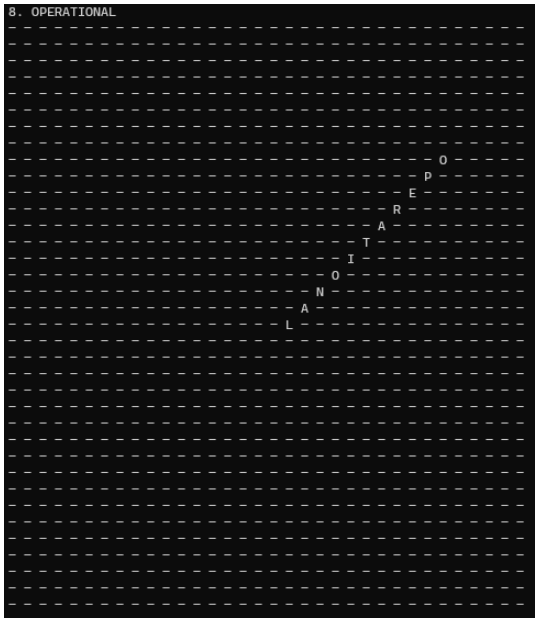

3. REGISTRATION
REGISTRATION

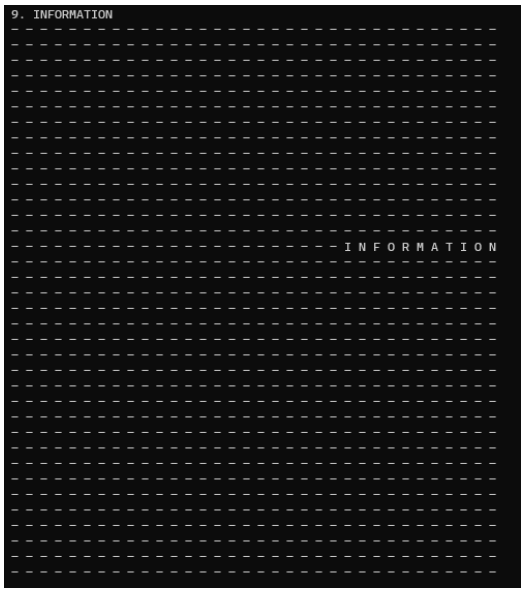

4. TRANSMISSION
TRANSMISSION

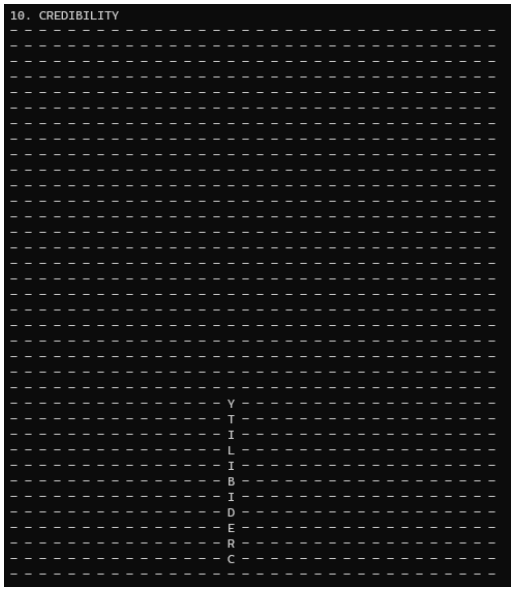

5. COMPLICATION
NOITACILPMOC

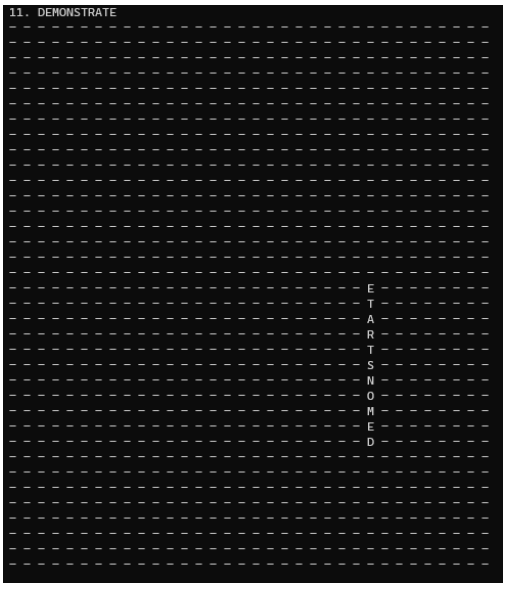

6. COUNTRYSIDE
COUNTRYSIDE

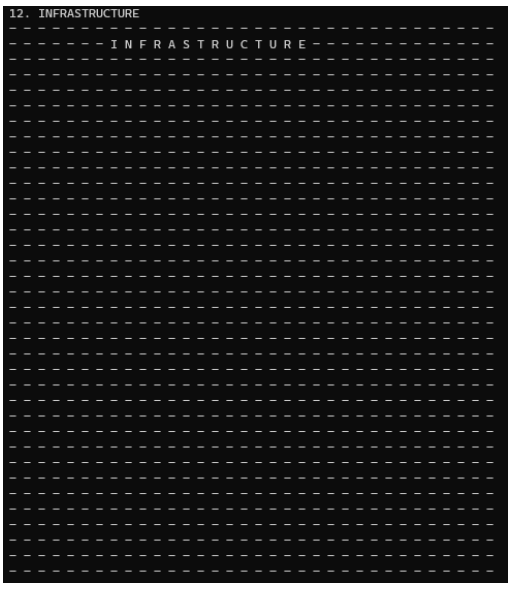

7. PREOCCUPATION
PREOCCUPATION


8. OPERATIONAL
OPERATIONAL


9. INFORMATION
INFORMATION


10. CREDIBILITY
CREDIBILITY


11. DEMONSTRATE
DEMONSTRATE


12. INFRASTRUCTURE
INFRASTRUCTURE

D. **Alamat Unduh Program**

Program dapat diunduh pada *repository* berikut:

https://github.com/blueguy42/Word-Search-Solver

E. Tabel Keberjalanan 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.	✓	