LAPORAN TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA
SEMESTER II TAHUN 2021/2022

PENYELESAIAN WORD SEARCH PUZZLE DENGAN ALGORITMA BRUTE FORCE

Disusun oleh :

STEVEN 13520131

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

Merancang algoritma *brute force* untuk menemukan semua kata di dalam *word search puzzle*.

1. Tulislah program kecil (sederhana) dalam Bahasa C/C++/Java yang mengimplementasikan algoritma Brute Force untuk mencari solusi word search puzzle.
2. Input: file teks yang berisi matriks huruf di dalam puzzle (antar huruf dipisahkan oleh spasi), diikuti satu baris kosong, dan daftar kata-kata yang akan dicari di dalam puzzle. Satu baris satu kata. Puzzle boleh dalam Bahasa Inggris atau Bahasa Indonesia.
3. Output: Tampilan di layar yang memuat:
    a. Semua kata yang ditemukan
    b. Waktu eksekusi program (tidak termasuk waktu pembacaan file input).
    c. Jumlah total perbandingan huruf yang dilakukan untuk menemukan kata di dalam puzzle.

BAB II
PENJELASAN ALGORITMA *BRUTEFORCE* YANG DIGUNAKAN

Pertama-tama, akan dilakukan pembacaan file yang diinput menjadi dua buah vector of string, yaitu satu untuk menampung huruf di dalam puzzle (word_matrix) dan satu lagi untuk menampung daftar kata-kata yang akan dicari di dalam *puzzle* (word_list). Selanjutnya, akan dijelaskan algoritma untuk menyelesaikan permasalahan ini. Kita ambil contoh sebagai berikut.

```
J S O L U T I S
S U N A R U U A
N E P T U N E T
S O N I E I S U
R C E V T R E R
A H T R A E S N
M M E R C U R Y
```

```
EARTH
JUPITER
MARS
MERCURY
NEPTUNE
SATURN
URANUS
VENUS
```

Pertama-tama, dilakukan pencarian lokasi dari seluruh abjad pertama dari kata-kata yang akan dicari, lalu catat kordinatnya dan simpan dalam array of array of tuple. Setelah diperoleh koordinat-koordinat dari karakter tersebut, barulah kita memulai pencarian. Pencarian akan dimulai dari koordinat yang telah diperoleh sebelumnya dengan cara mencari pada 8 arah. Pencarian 8 arah ini merupakan algoritma *brute force*.

Misalnya kita ingin mencari EARTH, maka kita perlu mendata seluruh koordinat dari abjad pertamanya. Selanjutnya, akan dilakukan pencarian pada 8 arah dengan melakukan pencocokan string dengan algoritma *brufe force*. Untuk karakter-karakter yang cocok, akan disimpan koordinatnya lalu selanjutnya akan dioutputkan hasilnya.

BAB III

KODE PROGRAM DALAM BAHASA C++

main.cpp

```cpp
#include <fstream>
#include <string>
#include <vector>
#include <iostream>
#include <cstdint>
#include <chrono>
#include <map>
#include <tuple>
#include <algorithm>
using std::int64_t;

// Debugging Helper
void print(std::vector<std::tuple<char, int64_t, int64_t>>& arg) {
    for(int i = 0; i < arg.size(); ++i) {
        std::cout << std::get<0>(arg[i]) << ' ';
    }
    std::cout << '\n';
}
// End of Debugging Helper

int main()
{
    std::ifstream file_input;
    std::string file_name;
    std::cout << "Masukkan nama file \n";
    std::cout << "Contoh: ../test/tc1.txt \n\n";
    std::cout << ">>> ";
    std::cin >> file_name;
    std::cout << "\n";
    file_input.open(file_name);
    std::vector<std::string> word_matrix;
    std::vector<std::string> word_list;
    std::map<char, std::vector<std::tuple<int64_t, int64_t>>> word_pos;

    int64_t compare_count = 0;

    for(std::string line_in; std::getline(file_input, line_in);)
    {
        if(line_in == "") continue;

        if(line_in[1] == ' ')
        {
            std::string temp;
            for(int64_t i = 0; i < line_in.size(); i += 2)
            {
                temp.push_back(line_in[i]);
            }
            word_matrix.emplace_back(std::move(temp));
        }
        else
        {
            word_list.emplace_back(std::move(line_in));
            auto iter = word_pos.find(word_list.back()[0]);
            if(iter == word_pos.end())
            {
                word_pos[word_list.back()[0]] = std::vector<std::tuple<int64_t, int64_t>>{};
            }
        }
    }

    std::vector<std::vector<std::tuple<char, int64_t, int64_t>>> output;

    auto time_start = std::chrono::steady_clock::now();

    for(int64_t i = 0; i < word_matrix.size(); ++i)
    {
        for(int64_t j = 0; j < word_matrix[i].size(); ++j)
        {
            compare_count++;
            auto iter = word_pos.find(word_matrix[i][j]);
            if(iter != word_pos.end())
            {
                word_pos[word_matrix[i][j]].emplace_back(std::make_tuple(i, j));
            }
        }
    }

    for(int i = 0; i < word_list.size(); ++i)
    {
        std::string word = word_list[i];

        for(int j = 0; j < word_pos[word[0]].size(); ++j)
        {
            int64_t row_num;
            int64_t col_num;
            std::tie(row_num, col_num) = word_pos[word[0]][j];

            bool found = true;
            // Check to the top
            std::vector<std::tuple<char, int64_t, int64_t>> output_temp;
            for(int64_t index = 0; index < word.size(); ++index)
            {
                if(row_num - index < 0)
                {
                    found = false;
                    break;
                }
                compare_count++;
                if(word[index] == word_matrix[row_num-index][col_num])
                {
                    output_temp.emplace_back(std::make_tuple(word[index], row_num - index, col_num));
                }
                else
                {
                    found = false;
                    break;
                }
            }
```

```
            if(found)
            {
                output.emplace_back(std::move(output_temp));
                continue;
            }

            output_temp.clear();

            found = true;
            // Check to the left
            for(int64_t index = 0; index < word.size(); ++index)
            {
                if(col_num - index < 0)
                {
                    found = false;
                    break;
                }
                compare_count++;
                if(word[index] == word_matrix[row_num][col_num-index])
                {
                    output_temp.emplace_back(std::make_tuple(word[index], row_num, col_num-index));
                }
                else
                {
                    found = false;
                    break;
                }
            }
            if(found)
            {
                output.emplace_back(std::move(output_temp));
                continue;
            }

            output_temp.clear();

            found = true;
            // Check to the bottom
            for(int64_t index = 0; index < word.size(); ++index)
            {
                if(row_num + index >= word_matrix.size())
                {
                    found = false;
                    break;
                }
                compare_count++;
                if(word[index] == word_matrix[row_num+index][col_num])
                {
                    output_temp.emplace_back(std::make_tuple(word[index], row_num + index, col_num));
                }
                else
                {
                    found = false;
                    break;
                }
            }
            if(found)
            {
                output.emplace_back(std::move(output_temp));
                continue;
            }

            output_temp.clear();

            found = true;
            // Check to the right
            for(int64_t index = 0; index < word.size(); ++index)
            {
                if(col_num + index >= word_matrix[0].size())
                {
                    found = false;
                    break;
                }
                compare_count++;
                if(word[index] == word_matrix[row_num][col_num+index])
                {
                    output_temp.emplace_back(std::make_tuple(word[index], row_num, col_num+index));
                }
                else
                {
                    found = false;
                    break;
                }
            }
            if(found)
            {
                output.emplace_back(std::move(output_temp));
                continue;
            }

            output_temp.clear();

            found = true;
            // Check to top-left
            for(int64_t index = 0; index < word.size(); ++index)
            {
                if(col_num - index < 0 || row_num - index < 0)
                {
                    found = false;
                    break;
                }
                compare_count++;
                if(word[index] == word_matrix[row_num-index][col_num-index])
                {
                    output_temp.emplace_back(std::make_tuple(word[index], row_num-index, col_num-index));
                }
                else
                {
```

```
                    found = false;
                    break;
                }
            }
            if(found)
            {
                output.emplace_back(std::move(output_temp));
                continue;
            }

            output_temp.clear();

            found = true;
            // Check to top-right
            for(int64_t index = 0; index < word.size(); ++index)
            {
                if(col_num + index >= word_matrix[0].size() || row_num - index < 0)
                {
                    found = false;
                    break;
                }
                compare_count++;
                if(word[index] == word_matrix[row_num-index][col_num+index])
                {
                    output_temp.emplace_back(std::make_tuple(word[index], row_num-index, col_num+index));
                }
                else
                {
                    found = false;
                    break;
                }
            }
            if(found)
            {
                output.emplace_back(std::move(output_temp));
                continue;
            }

            output_temp.clear();

            found = true;
            // Check to bottom-left
            for(int64_t index = 0; index < word.size(); ++index)
            {
                if(col_num - index < 0 || row_num + index >= word_pos.size())
                {
                    found = false;
                    break;
                }
                compare_count++;
                if(word[index] == word_matrix[row_num+index][col_num-index])
                {
                    output_temp.emplace_back(std::make_tuple(word[index], row_num+index, col_num-index));
                }
                else
                {
                    found = false;
                    break;
                }
            }
            if(found)
            {
                output.emplace_back(std::move(output_temp));
                continue;
            }

            output_temp.clear();

            found = true;
            // Check to bottom-right
            for(int64_t index = 0; index < word.size(); ++index)
            {
                if(col_num + index >= word_matrix[0].size() || row_num + index >= word_matrix.size())
                {
                    found = false;
                    break;
                }
                compare_count++;
                if(word[index] == word_matrix[row_num+index][col_num+index])
                {
                    output_temp.emplace_back(std::make_tuple(word[index], row_num+index, col_num+index));
                }
                else
                {
                    found = false;
                    break;
                }
            }
            if(found)
            {
                output.emplace_back(std::move(output_temp));
                continue;
            }
            output_temp.clear();
        }
    }

    // Return hasil
    for(auto& result : output)
    {
        for(int i = 0; i < word_matrix.size(); ++i)
        {
            for(int j = 0; j < word_matrix[i].size(); ++j)
            {
                auto iter = std::find_if(result.begin(), result.end(), [&i, &j](std::tuple<char, int64_t, int64_t> el)
                {
                    return std::get<1>(el) == i && std::get<2>(el) == j;
                });
```

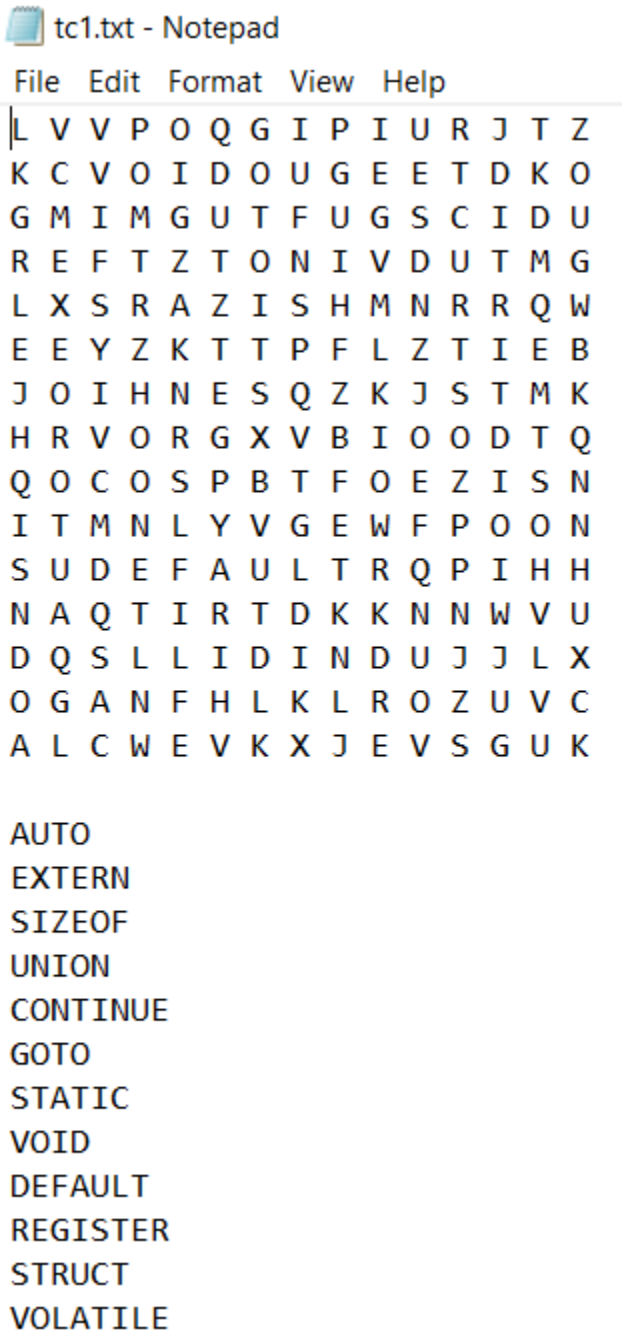
```
                if(iter != result.end())
                {
                    std::cout << std::get<0>(*iter);
                }
                else
                {
                    std::cout << '-';
                }
                std::cout << ' ';
            }
            std::cout << '\n';
        }
        std::cout <<"-----------------------------------------------------------------\n\n";
    }
    auto time_end = std::chrono::steady_clock::now();
    std::cout << "Waktu yang diperlukan untuk mengeksekusi program adalah " << std::chrono::duration_cast<std::chrono::duration<double>>(time_end - time_start).count() << " detik\n";
    std::cout << "Banyak operasi perbandingan kata yang dilakukan program ini adalah " << compare_count << " perbandingan \n\n";
}

```

BAB IV

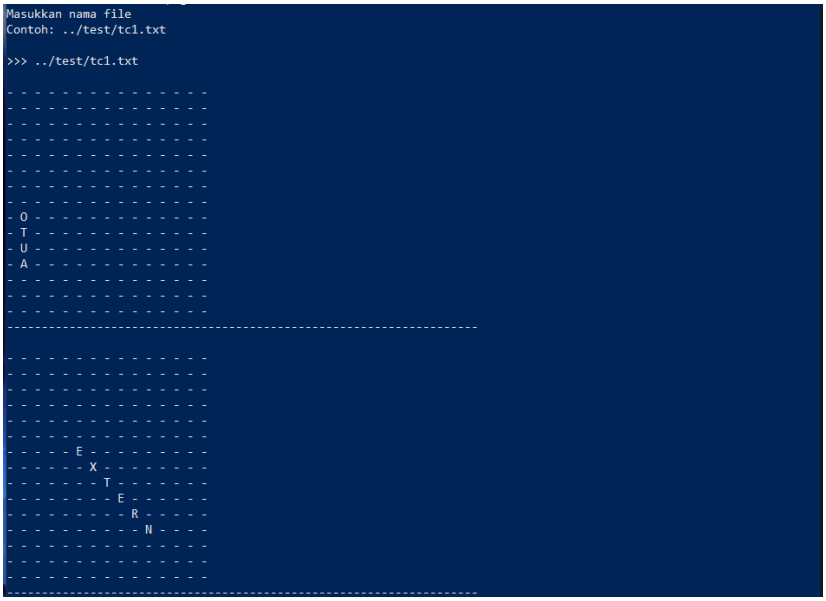
INPUT/OUTPUT PROGRAM

Test Case 1 (Small)

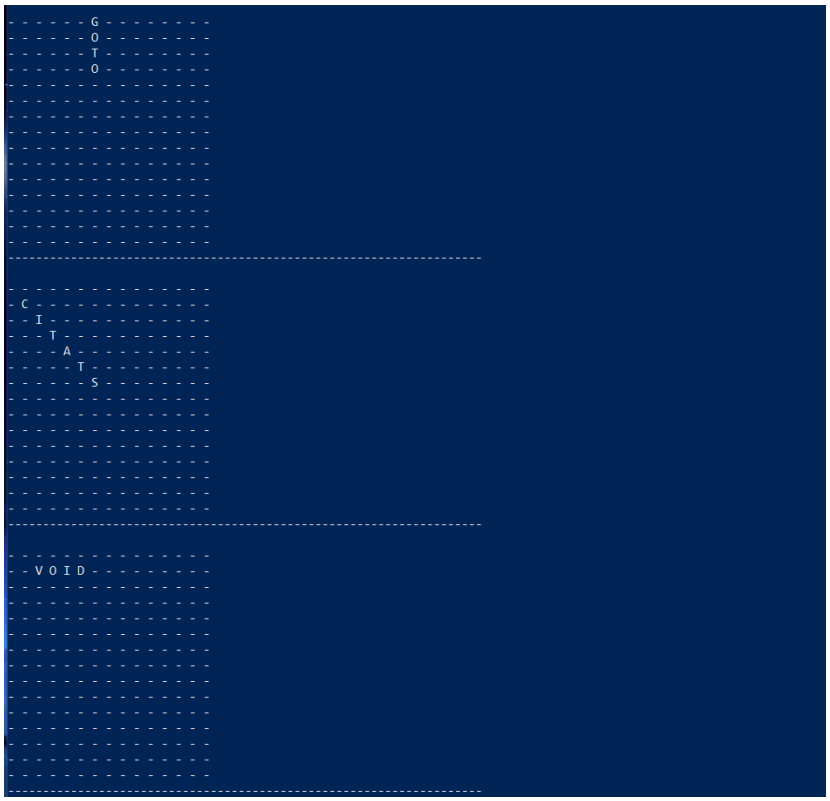
Input



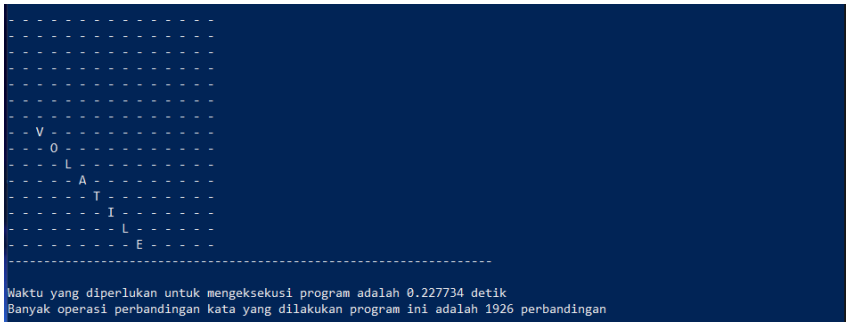
Output


Masukkan nama file
Contoh: ../test/tc1.txt
>>> ../test/tc1.txt


F O E Z I S
N
O
I
N
U
E
U
N
I
T
N
O
C


GOTO
CIT
AT
S
VOID


DEFAULT
RETURN
GIST
TCURT
S



Test Case 2 (Small)

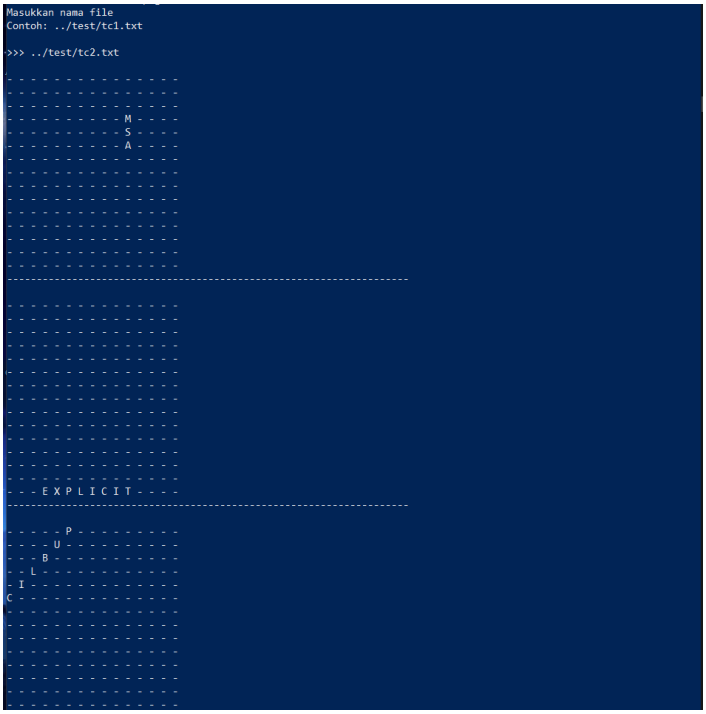
Input

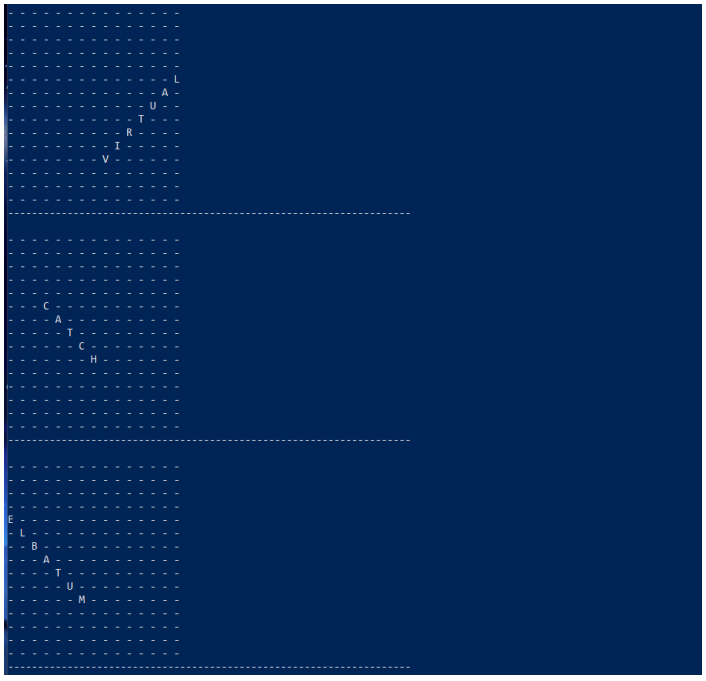
tc2.txt - Notepad

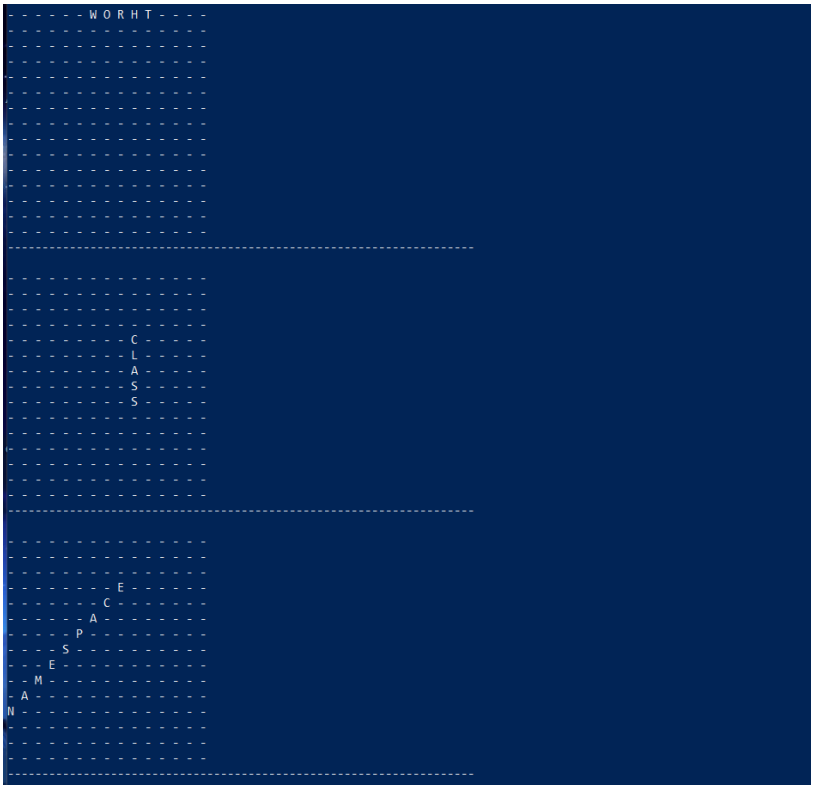
File Edit Format View Help

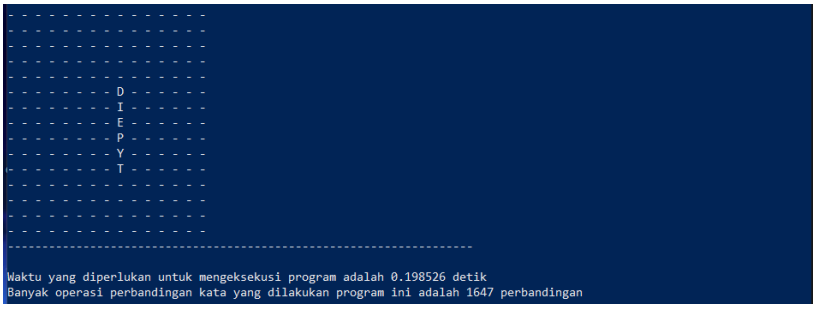
```
N A X C W P W O R H T X W A R
V I O U U I Q Y R P L K I W T
Y G P B J G F Y J B W L N C I
D B L Q U F D D E J M J C O K
E I S N G O S C V C S D J P P
C L P C Y E A K D L A V Z Z L
C U B K A P Y S I A Q W O A X
T R L A S T Z G E S F Y U U T
O K W E T V C X P S B T B Y W
A B M G Q U B H Y P R X G K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

ASM
EXPLICIT
PUBLIC
VIRTUAL
CATCH
MUTABLE
THROW
CLASS
NAMESPACE
TYPEID
```

Output


Masukkan nama file
Contoh: ../test/tc1.txt
>>> ../test/tc2.txt
EXPLICIT



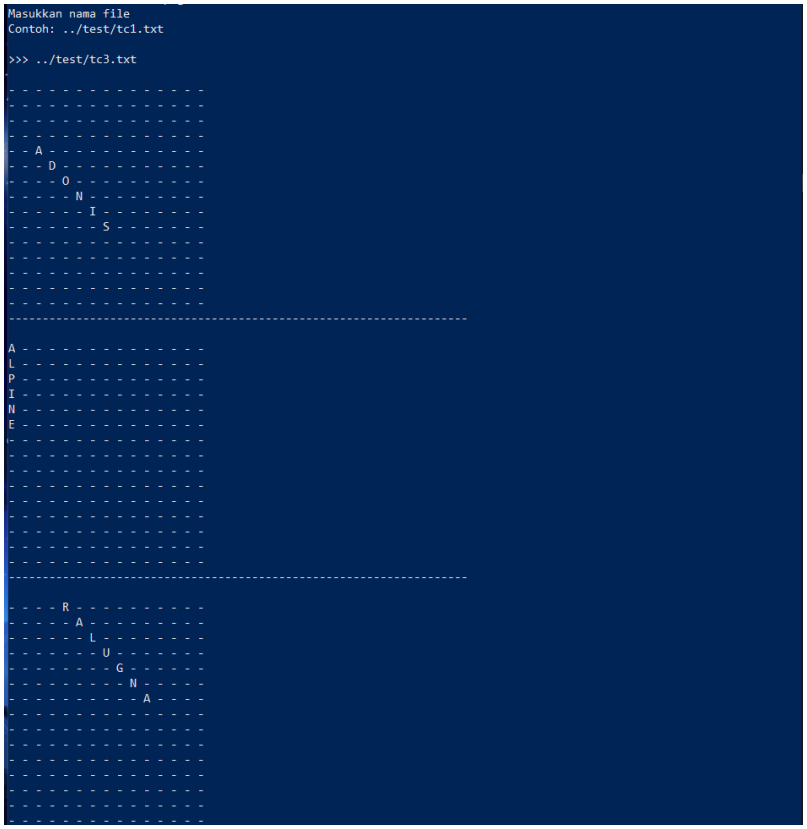

W O R H T
C
L
A
S
S
E
C
A
P
S
E
M
A
N

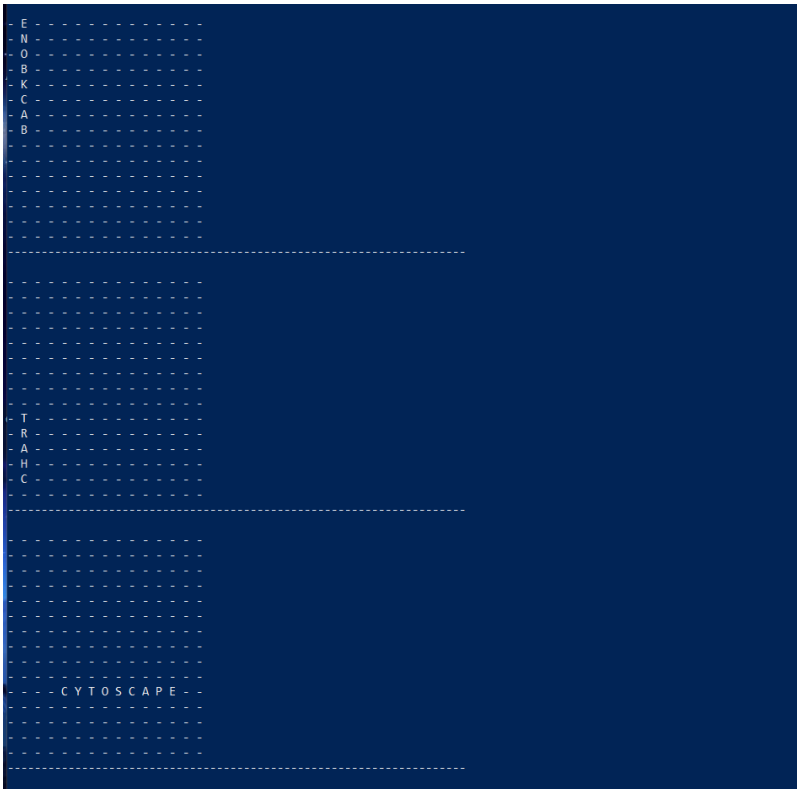

D
I
E
P
Y
T
Waktu yang diperlukan untuk mengeksekusi program adalah 0.198526 detik
Banyak operasi perbandingan kata yang dilakukan program ini adalah 1647 perbandingan

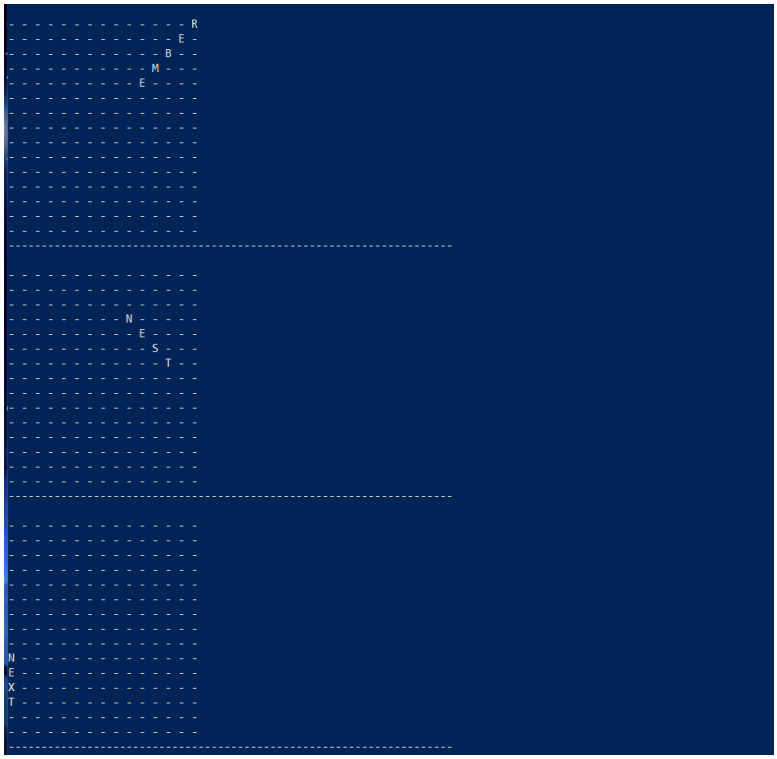
Test Case 3 (Small)

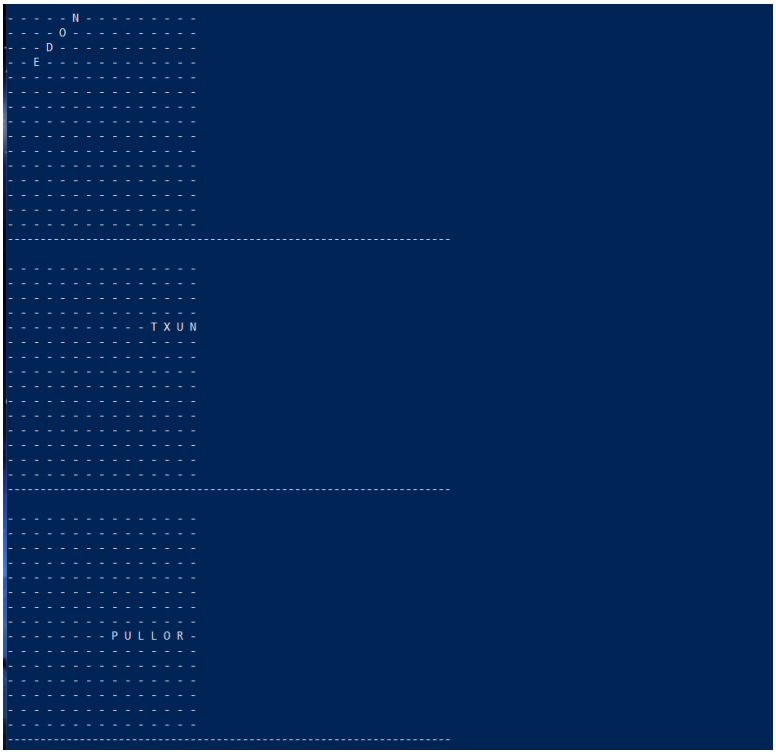
Input

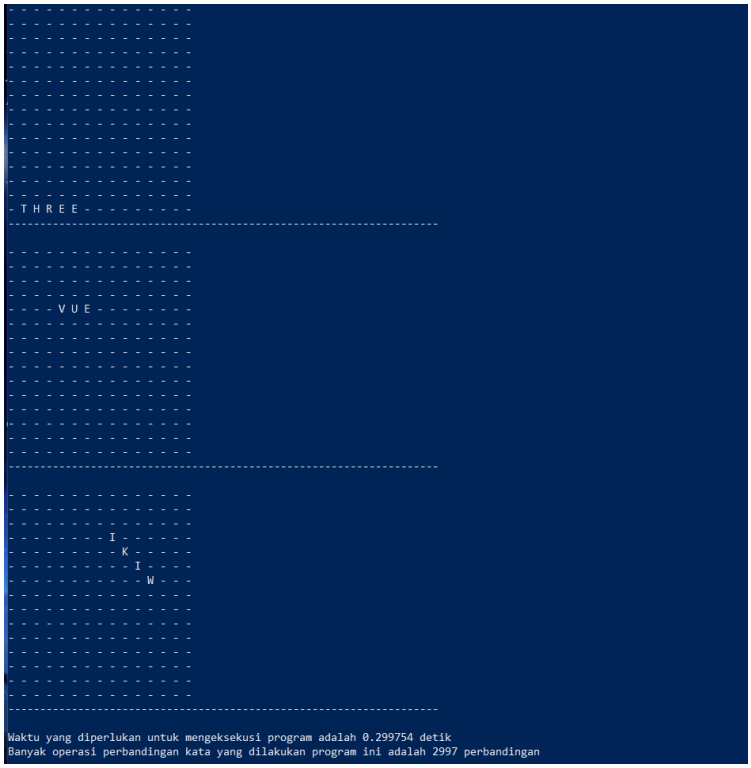
Output


Masukkan nama file
Contoh: ../test/tc1.txt
>>> ../test/tc3.txt
A L P I N E
R A L U G N A


E N O B K C A B
T R A H C
C Y T O S C A P E







Test Case 4 (Medium)

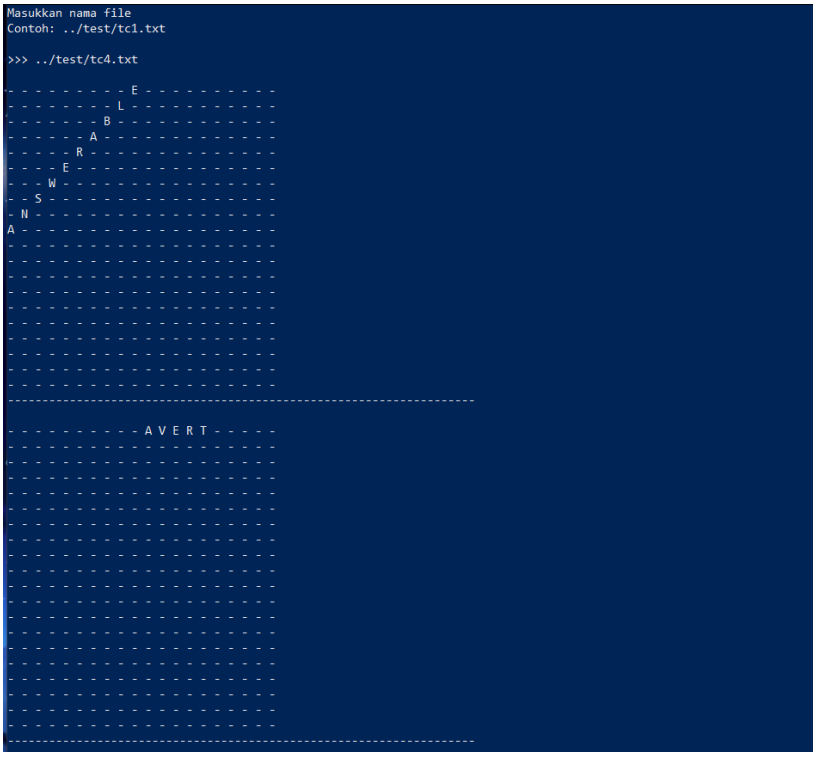
Input

tc4.txt - Notepad

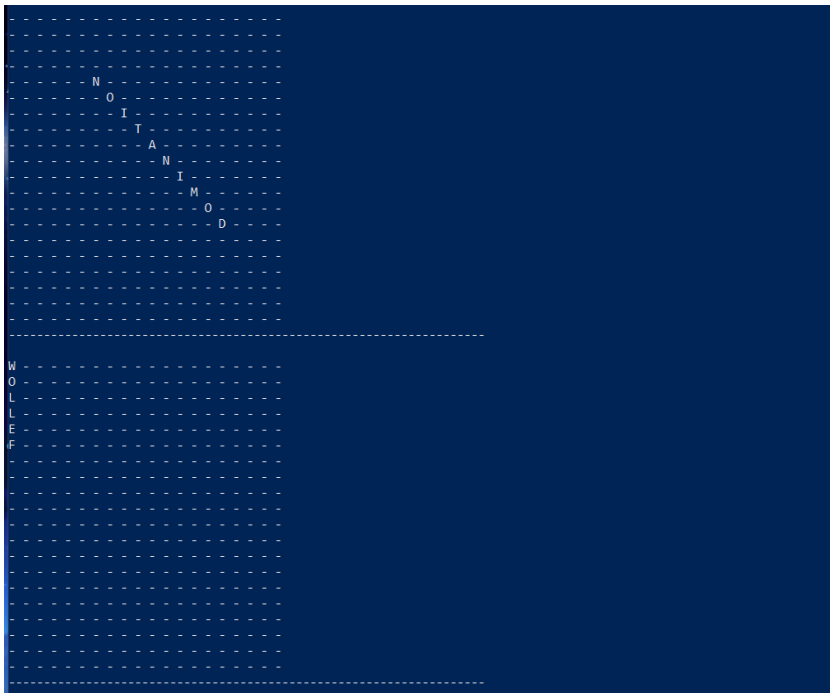
File Edit Format View Help

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

ANSWERABLE
AVERT
BEDAZZLE
CREDULITY
CUTBACK
DELIVERING
DOMINATION
FELLOW
GRAVITAS
ICICLE
PALM
REACTIVATE
SAVORY
SEVER
SIMPLEST
SKETCHY
SPHERE
SQUARING
STUFFIEST
SWIMMER
```

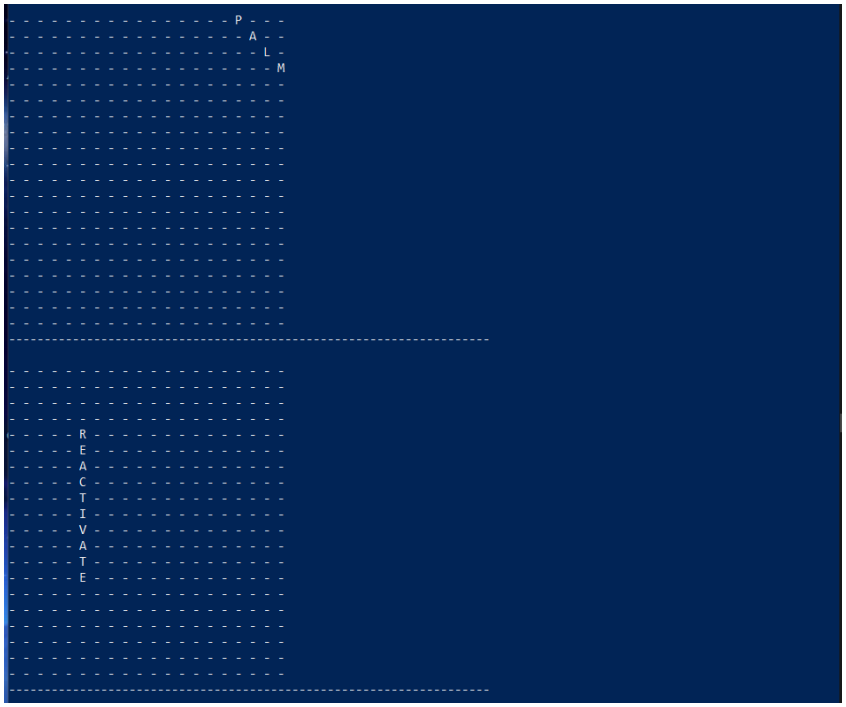
Output



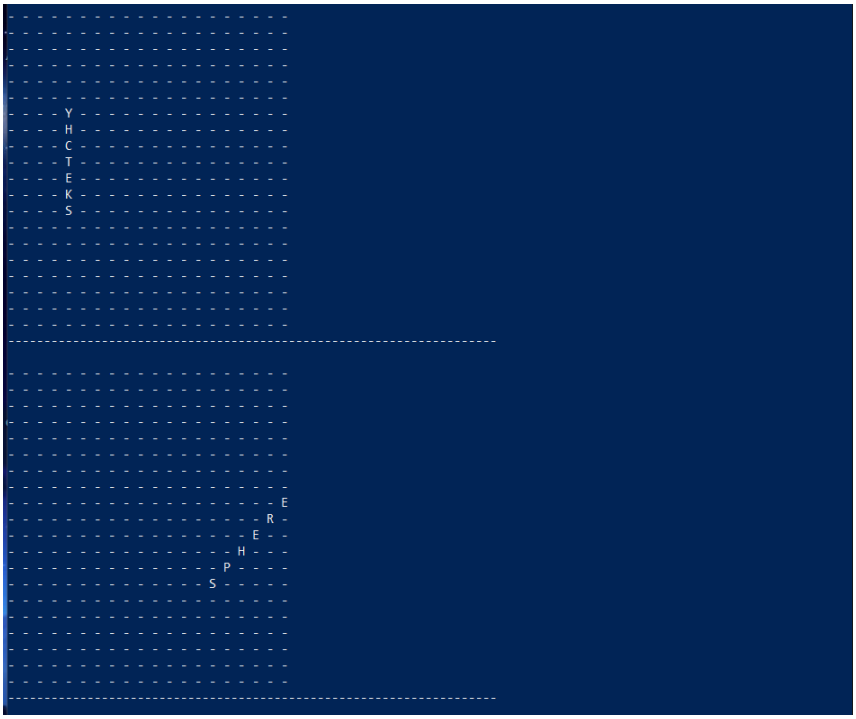

C
U
T
B
A
C
K
GNIREVILED


N
O
I
T
A
N
I
M
O
D
W
O
L
L
E
F

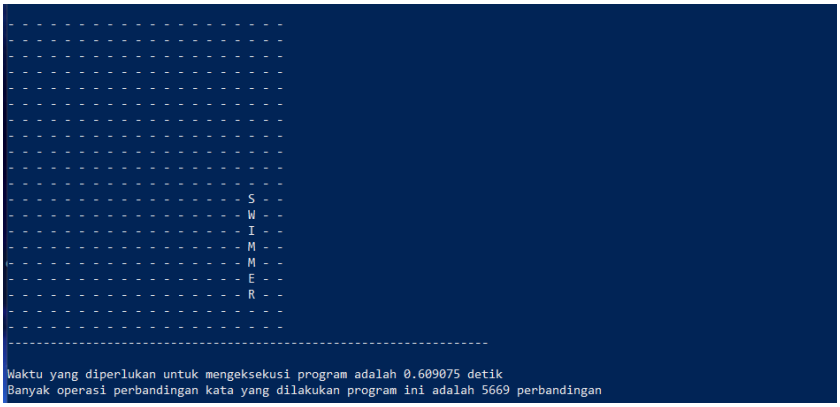

S
A
T
I
V
A
R
G
ICICLE


P
A
L
M
REACTIVATE


REVES
T
S
E
L
P
M
I
S


Y
H
C
T
E
K
S
E
R
E
H
P
S

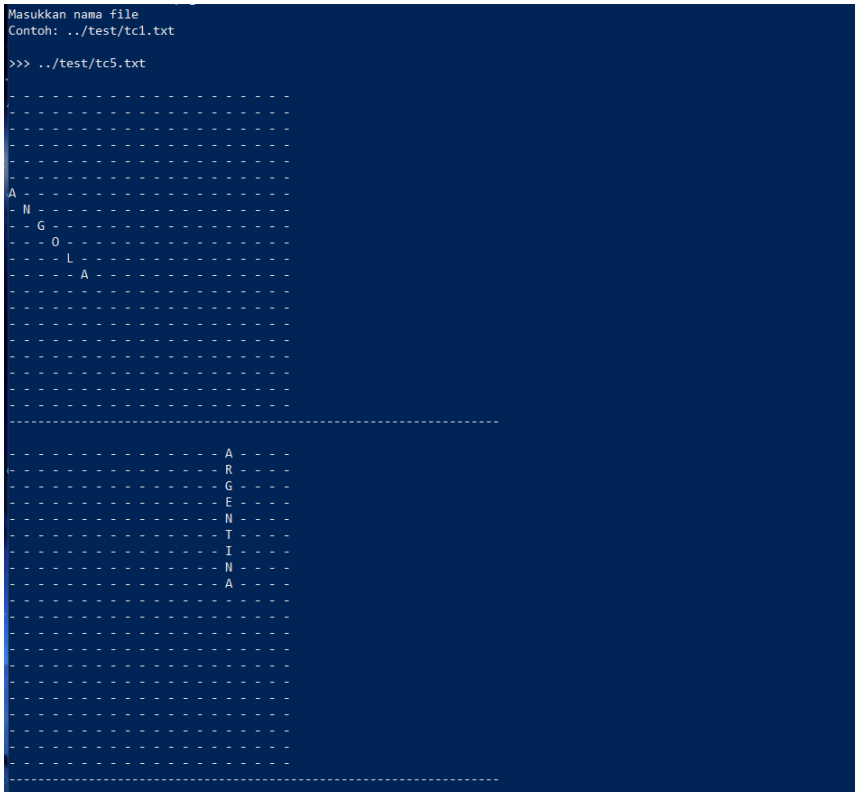

SQUARING
S
T
U
F
F
I
E
S
T

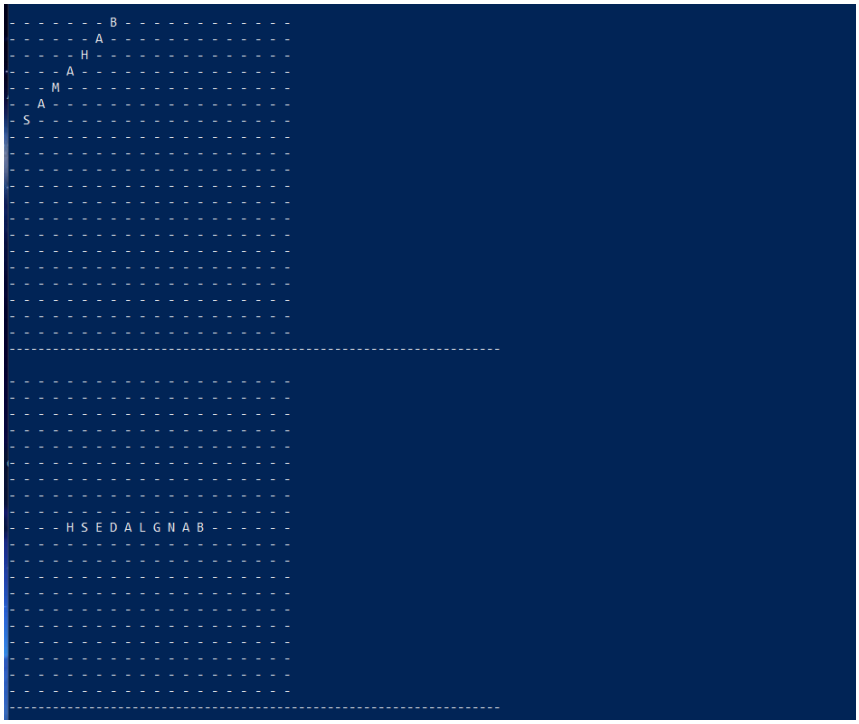

S
W
I
M
M
E
R
Waktu yang diperlukan untuk mengeksekusi program adalah 0.609075 detik
Banyak operasi perbandingan kata yang dilakukan program ini adalah 5669 perbandingan

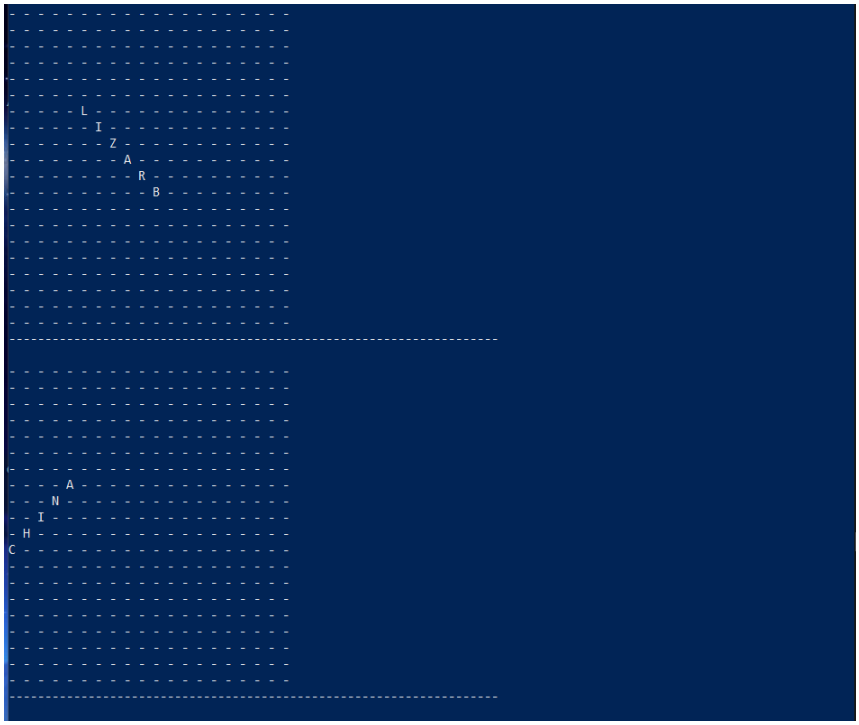
Test Case 5 (Medium)

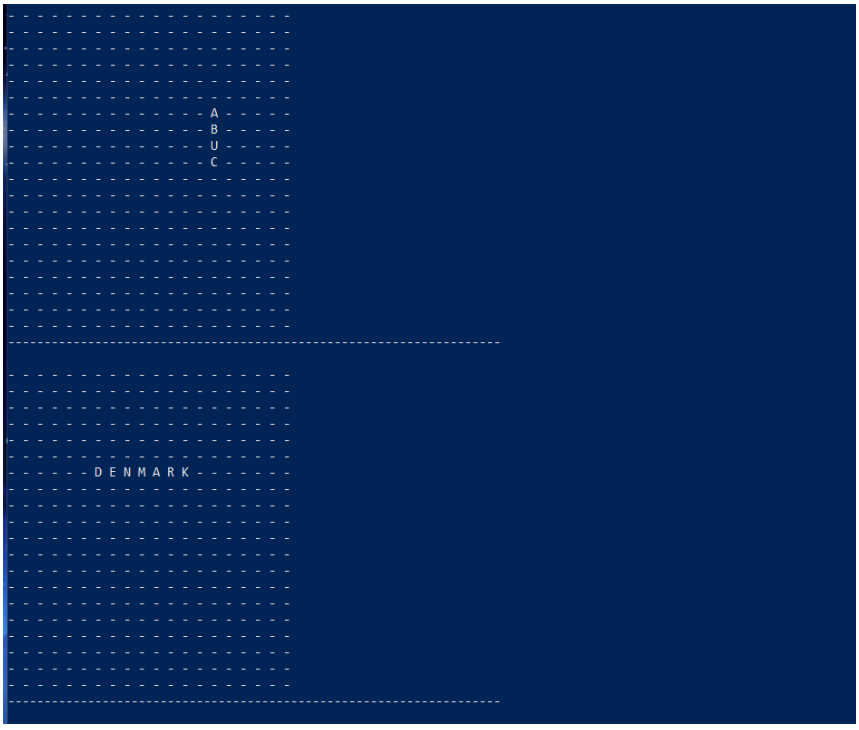
Input

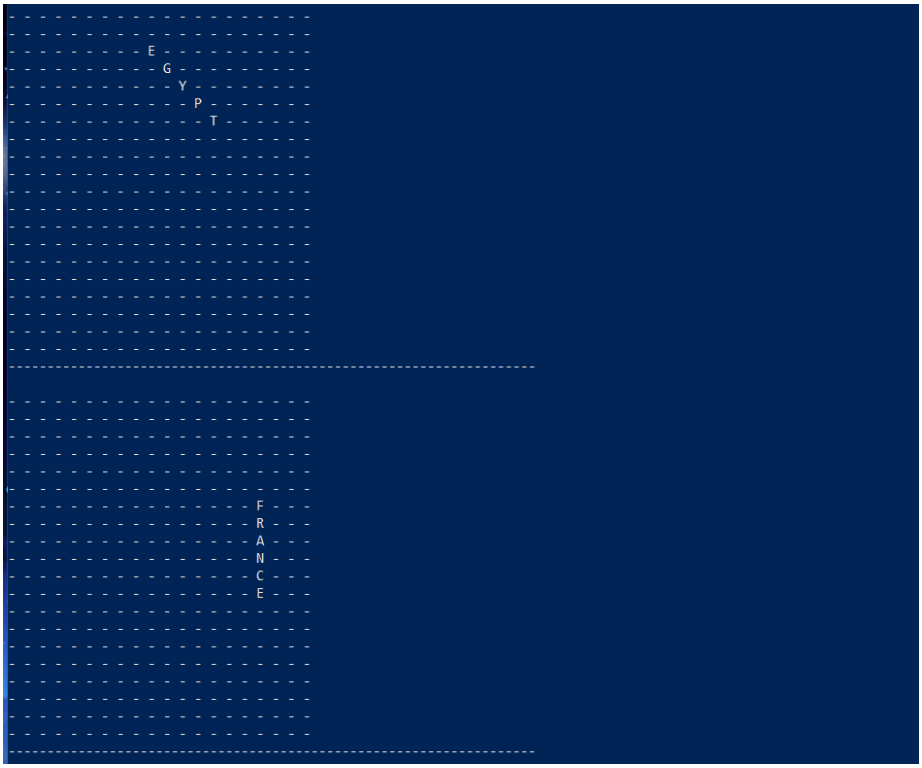
Output



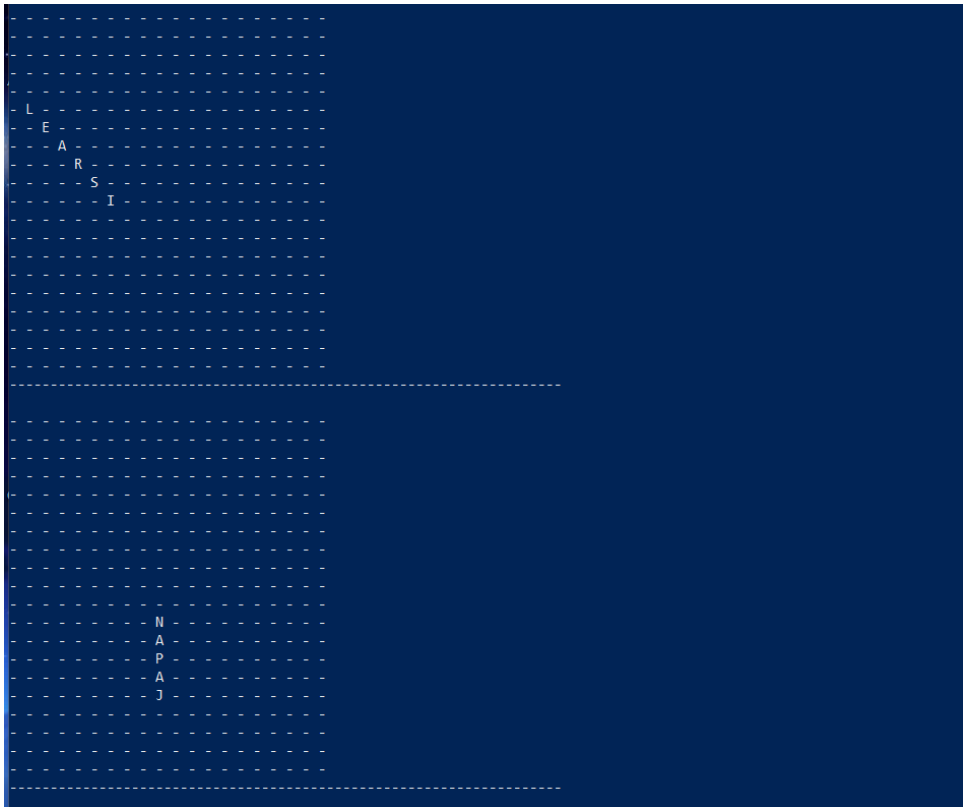


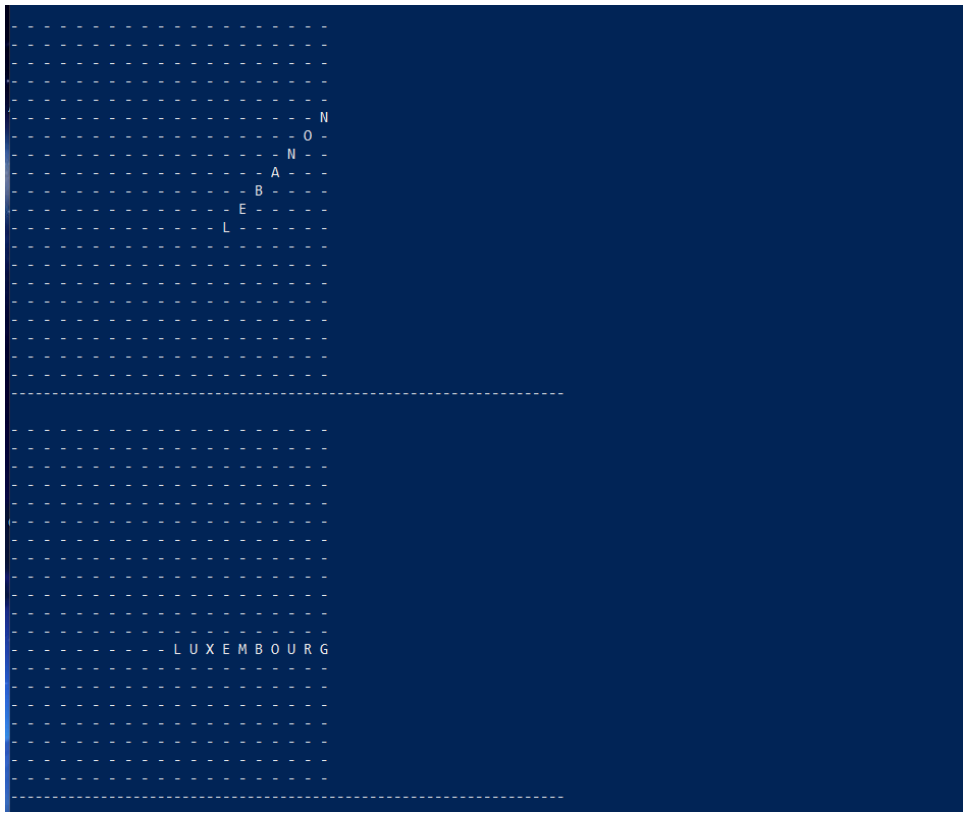

L
I
Z
A
R
B
A
N
I
H
C

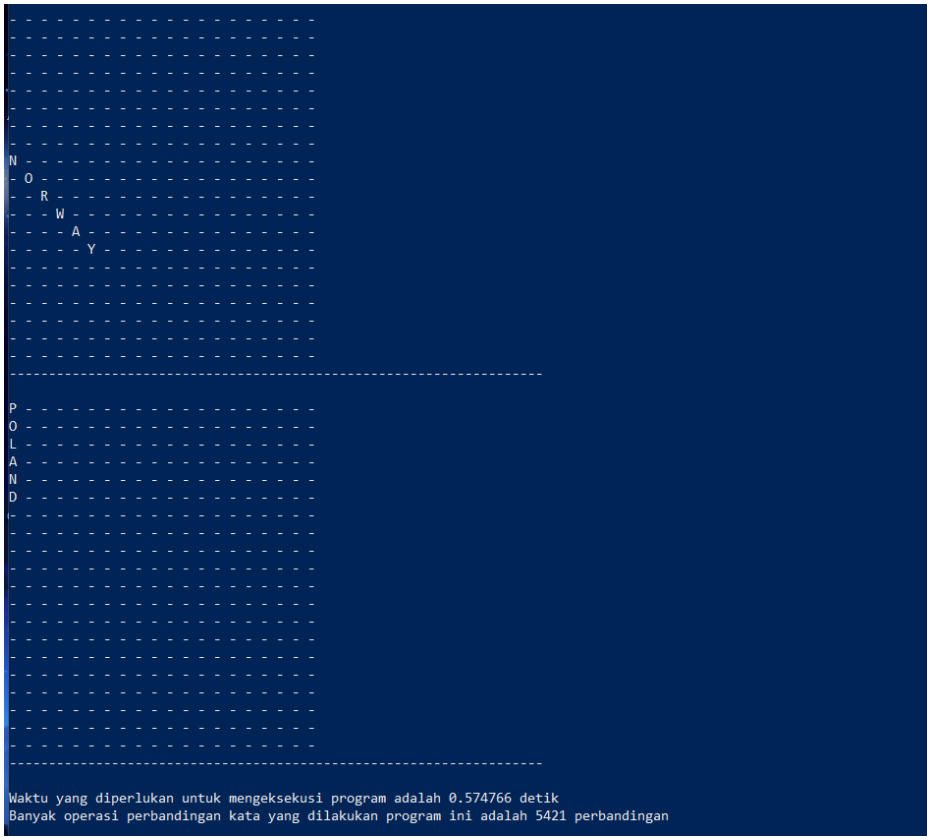

A
B
U
C
DENMARK


E
G
Y
P
T
F
R
A
N
C
E


G
U
I
N
E
A
AISENODNI


L
E
A
R
S
I
N
A
P
A
J


N
O
N
A
B
E
L
LUXEMBOURG



Test Case 6 (Medium)

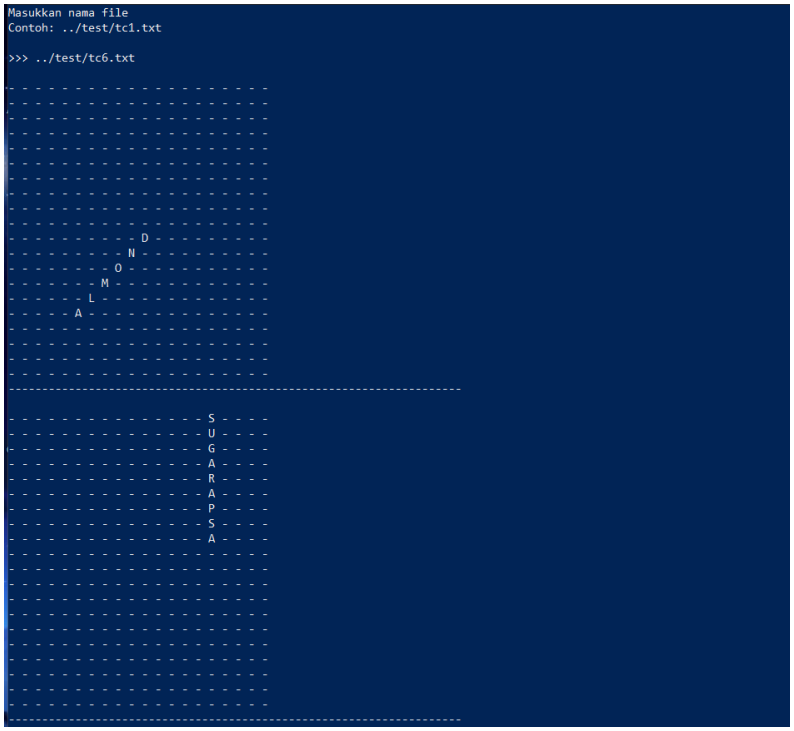
Input

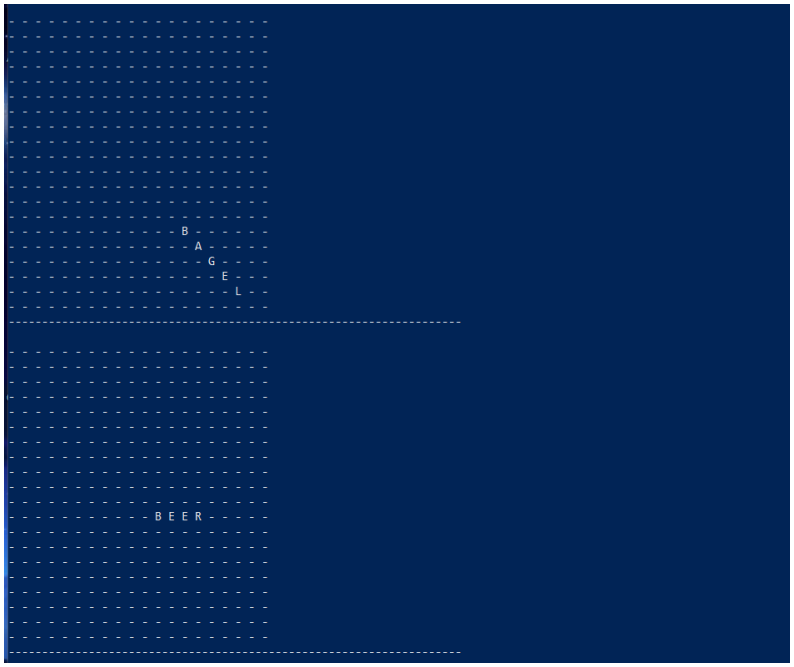
tc6.txt - Notepad

File Edit Format View Help

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

ALMOND
ASPARAGUS
BAGEL
BEER
CAKE
DUMPLINGS
FISH
GINGER
HONEY
JELLY
KIWI
LOBSTER
WAFFLES
```

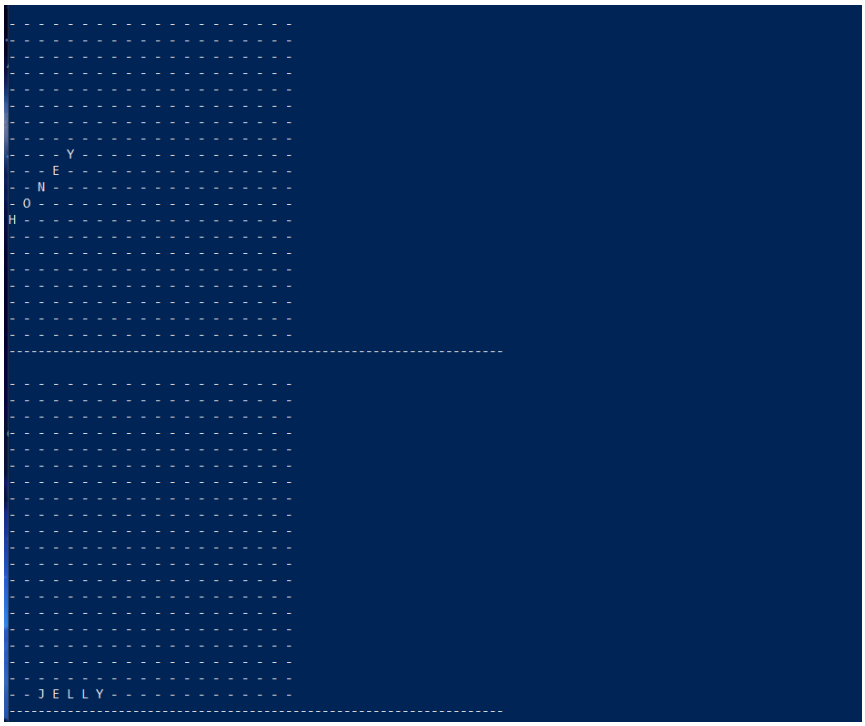
Output



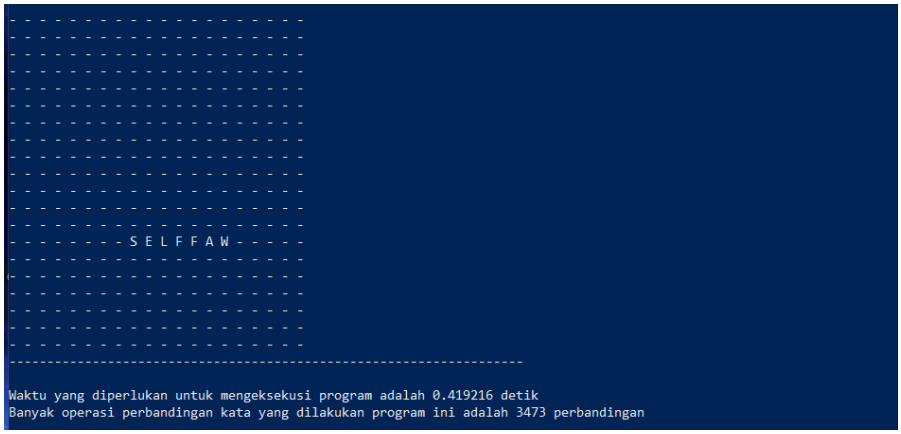



C
A
K
E
SGNILPMUD


H
S
I
F
R
E
G
N
I
G


Y
E
N
O
H
J E L L Y


K
I
W
I
L
O
B
S
T
E
R


- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - S E L F F A W - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -

Waktu yang diperlukan untuk mengeksekusi program adalah 0.419216 detik
Banyak operasi perbandingan kata yang dilakukan program ini adalah 3473 perbandingan

Test Case 7 (Large)

Input

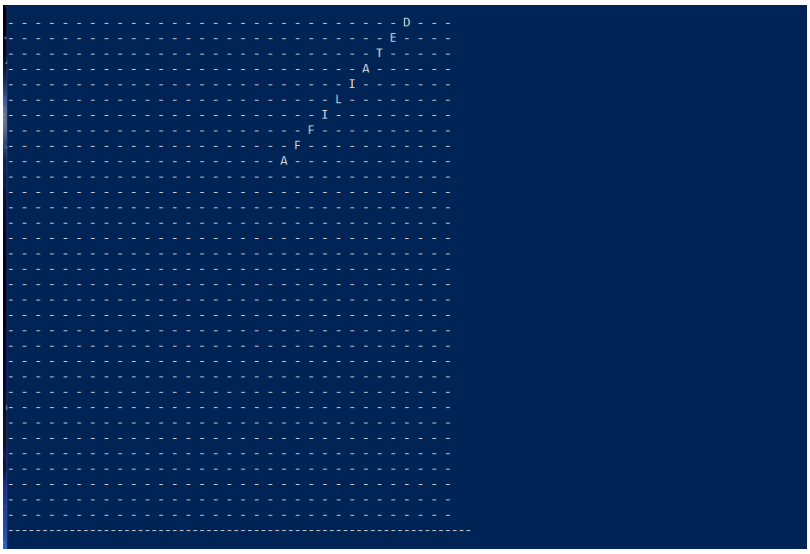

```
ACCORDED
ACUTELY
ADVANTAGE
AFFILIATED
BARE
BUSTS
CHAINED
CHIEFS
COMPUNCTION
COUNTERFEIT
CREEP
CREWMEN
DEPLETION
DIAGRAM
DRAFTING
ECONOMIC
EXPURGATED
FAVORABLE
FRESH
GUARANTOR
GUILE
IDIOCY
INSCRIBING
INSPECTION
INVESTIGATOR
MAGISTERIAL
MIGRATING
MUCUS
NEGATIVE
OBJECTING
PLATTER
POISONED
QUOTING
REPUGNANT
SALESMAN
SCABBED
SEEMED
SPARKLING
SPLATTER
SQUASH
SWARMING
SYNTHESIZE
TEXAS
THOUGHTFULLY
UNFASTENED
VIOLATED
WAFFLED
WAVERED
```

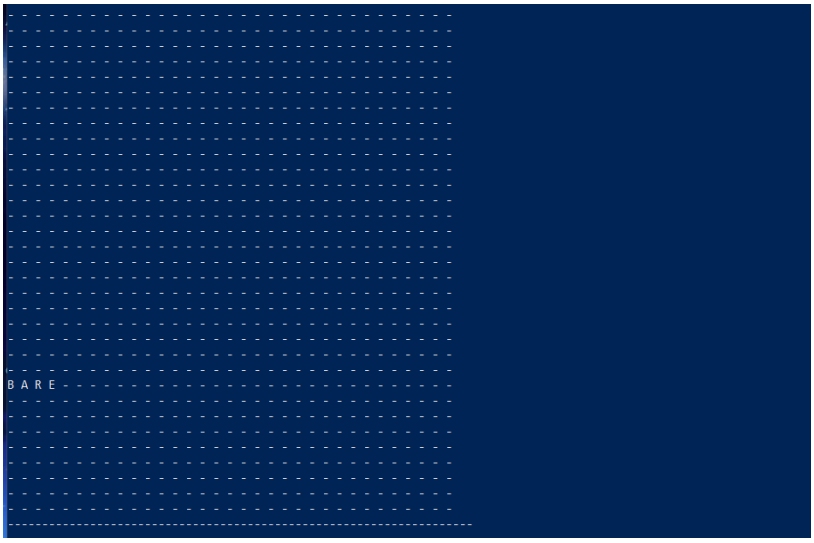
Output

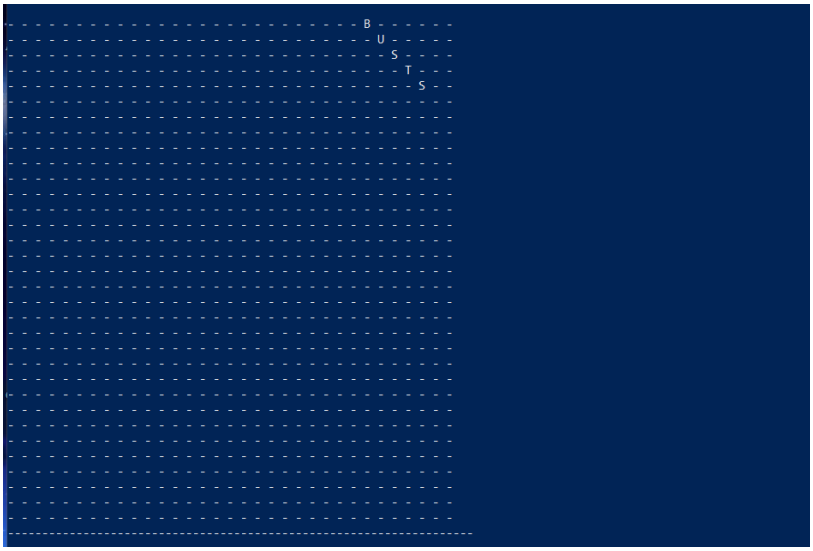

Masukkan nama file
Contoh: ../test/tc1.txt
>>> ../test/tc7.txt
D E D R O C C A

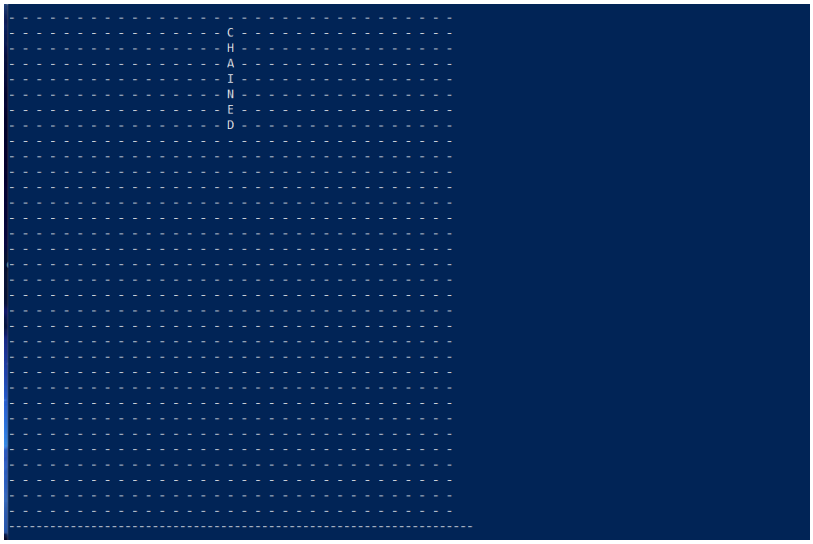

Y
L
E
T
U
C
A

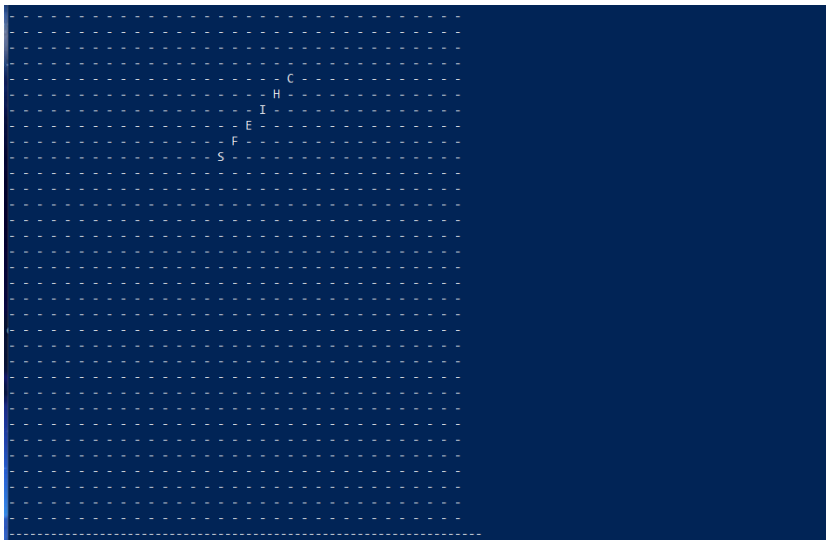

E
G
A
T
N
A
V
D
A

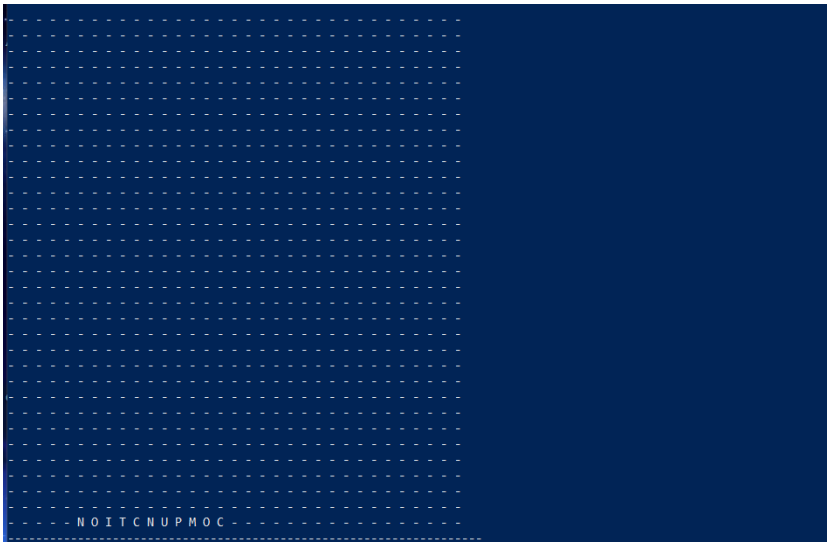

DETAILFINAL

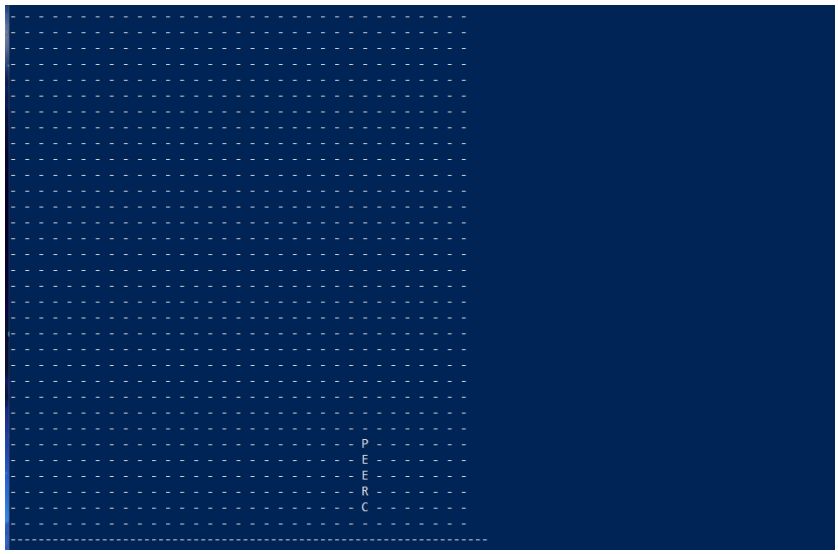

BARE

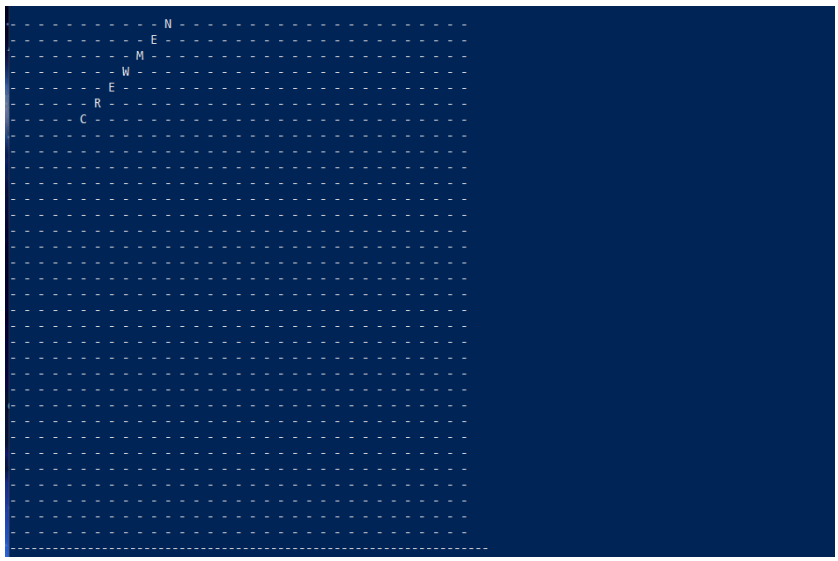

BUST


CHAINED

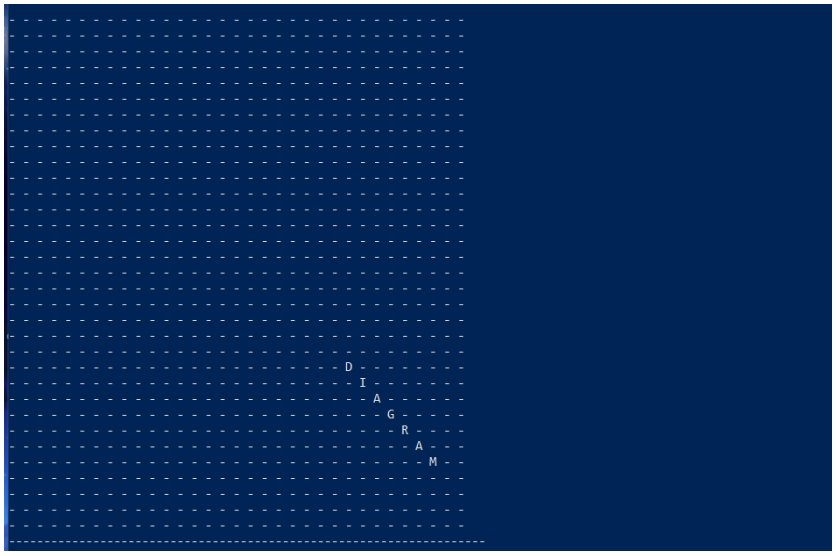

CHIEFS

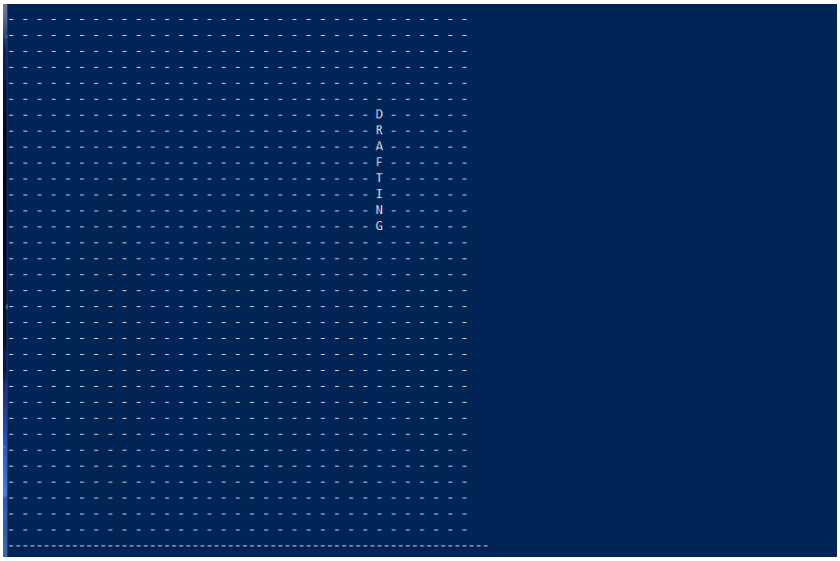

COMPUTATION

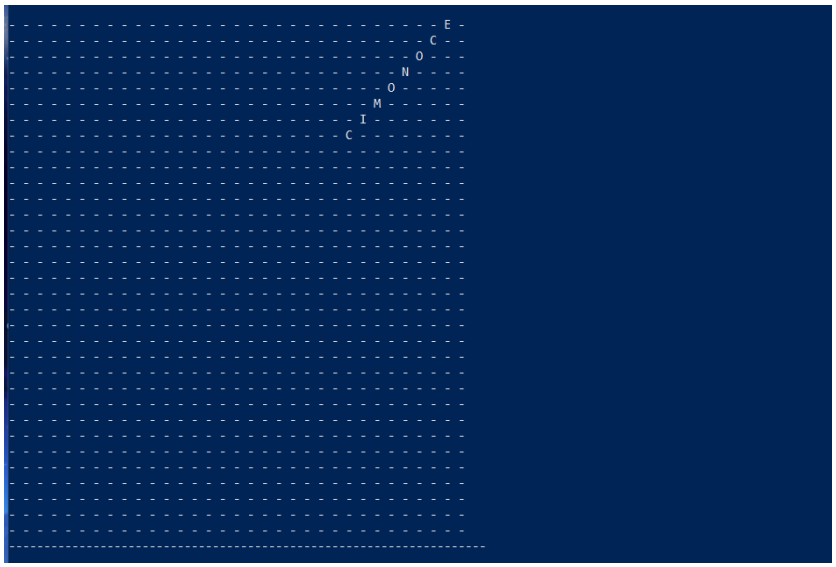


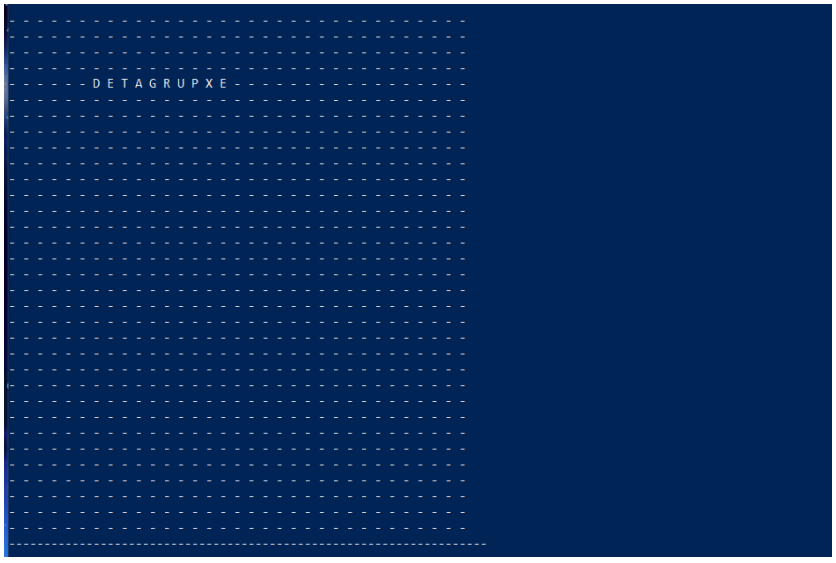



DEPLETION


D
I
A
G
R
A
M


D
R
A
F
T
I
N
G

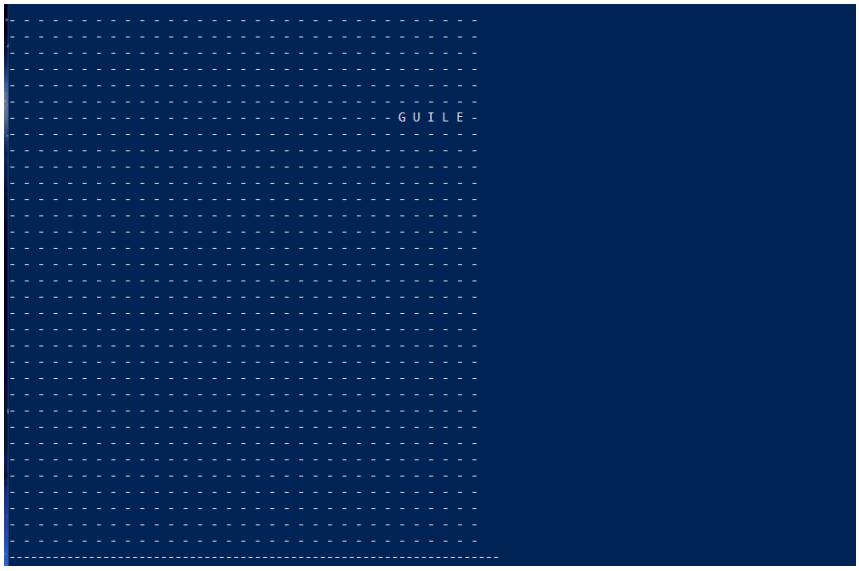

E
C
O
N
O
M
I
C

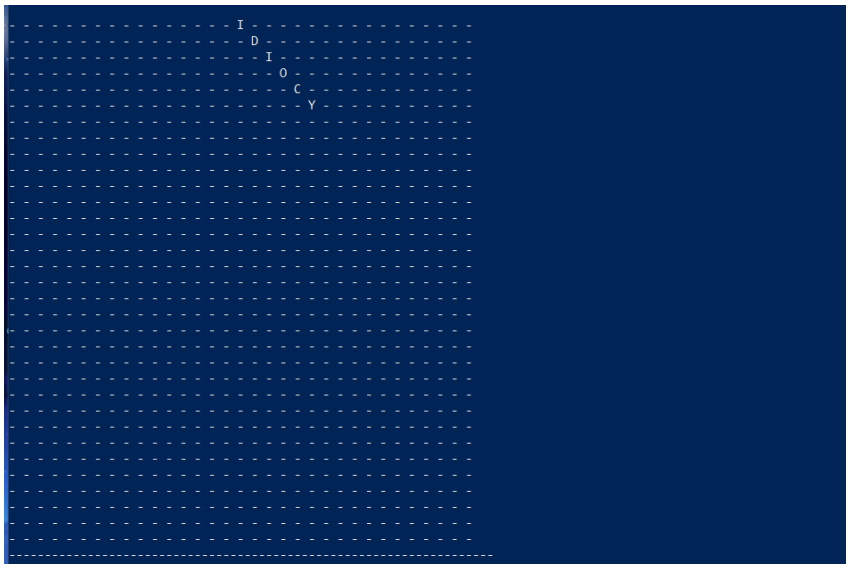

DETAGRUPXE

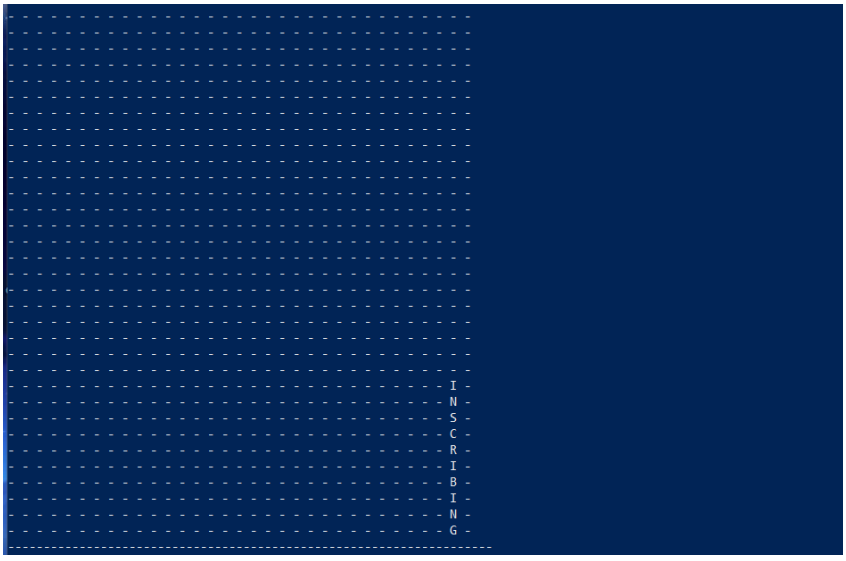

E
L
B
A
R
O
V
A
F

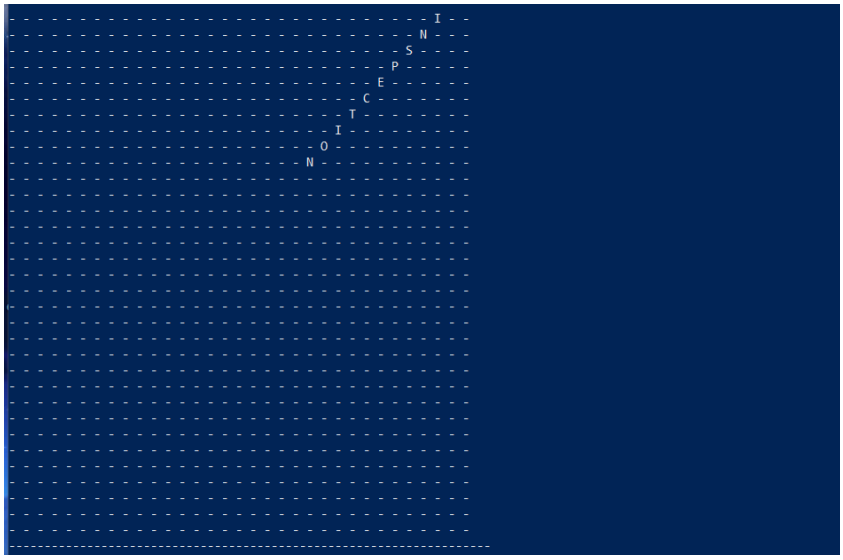

H
S
E
R
F


R
O
T
N
A
R
A
U
G

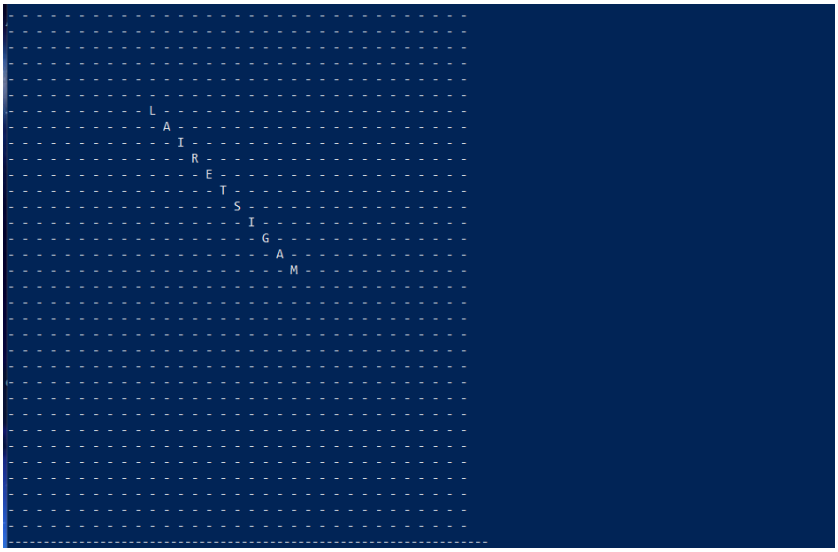

G U I L E

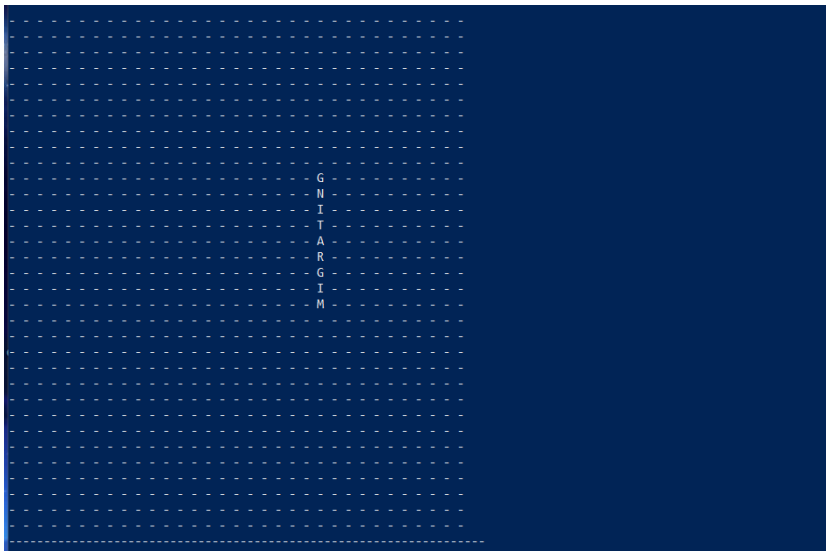

I
D
I
O
C
Y

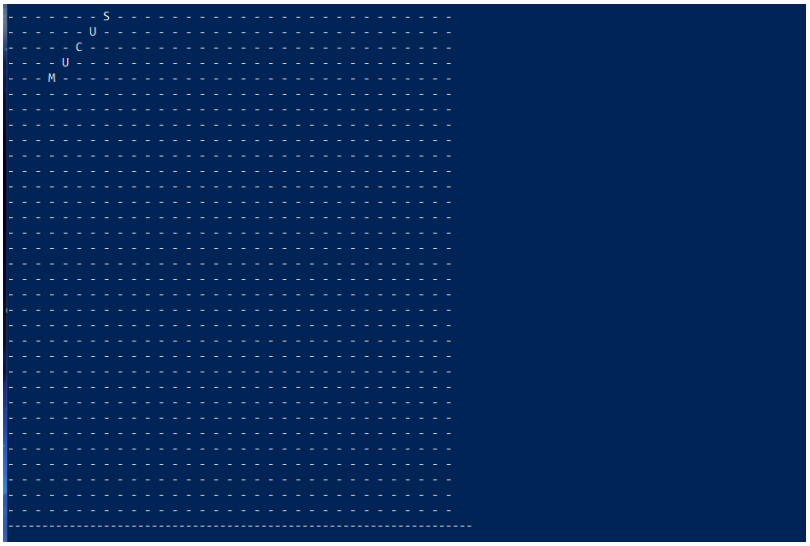

I
N
S
C
R
I
B
I
N
G


I
N
S
P
E
C
T
I
O
N

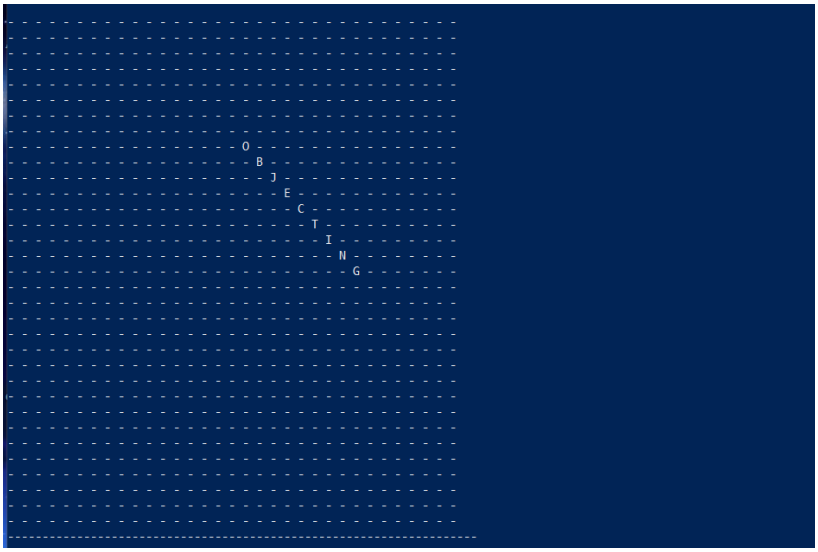

ROTAGITSEVNI

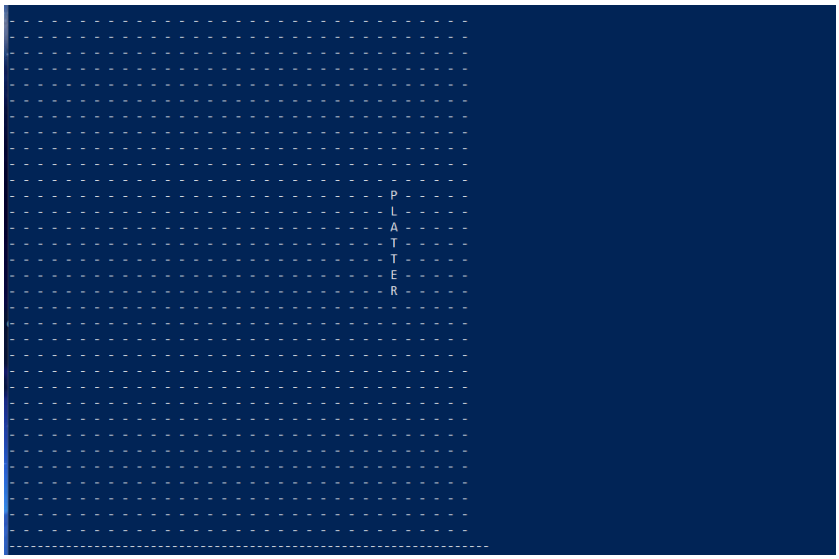

LAIRETSIGAM

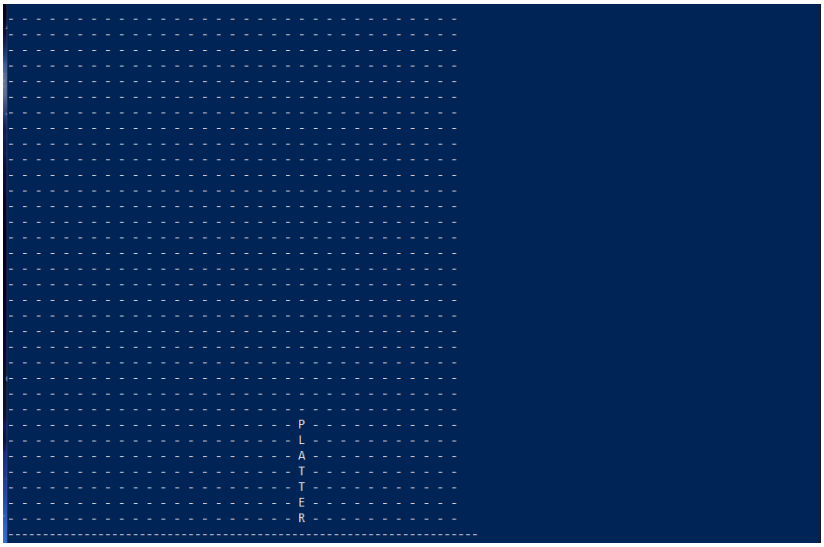

GNITARGIM

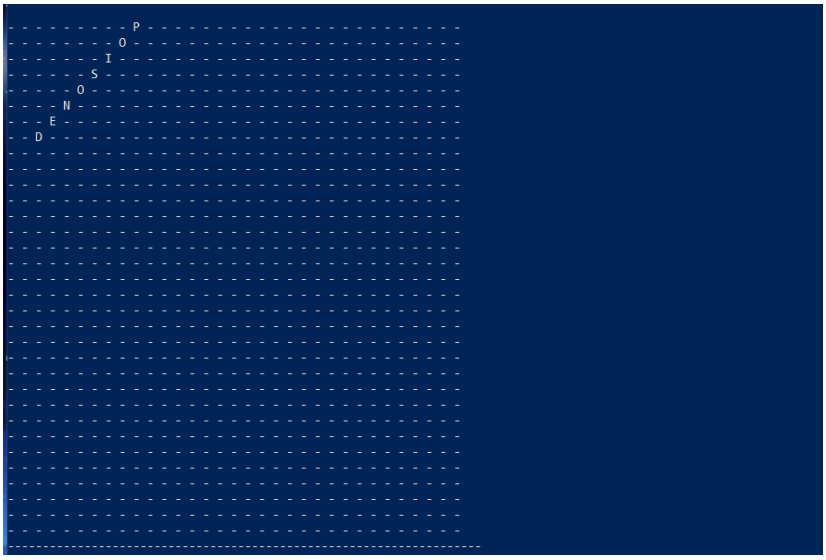

S
U
C
U
M

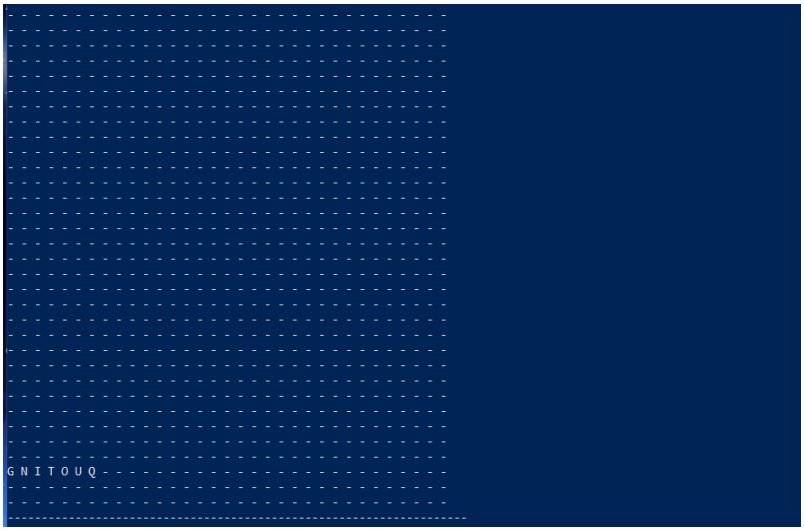

NEGATIVE

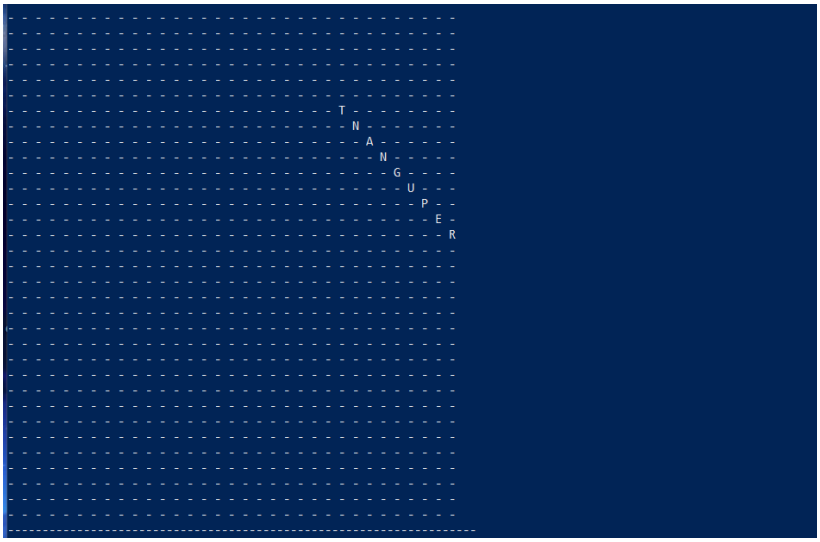

O
B
J
E
C
T
I
N
G

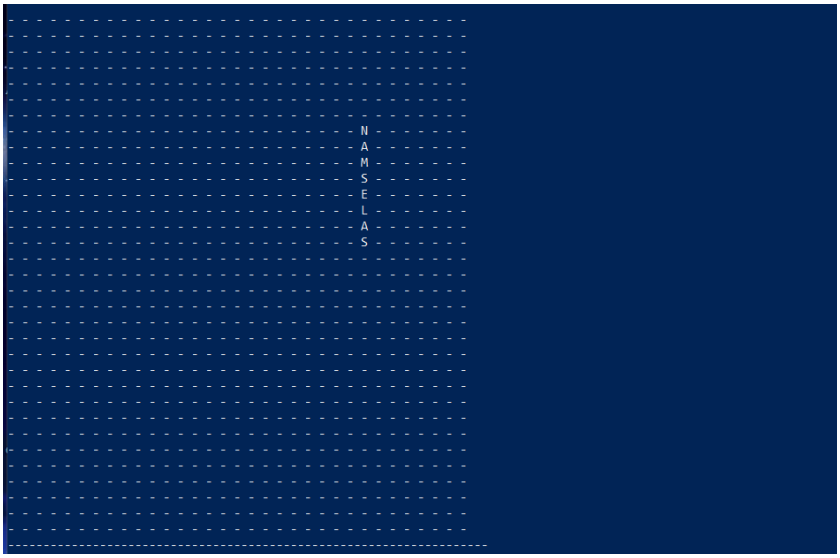

PLATTER

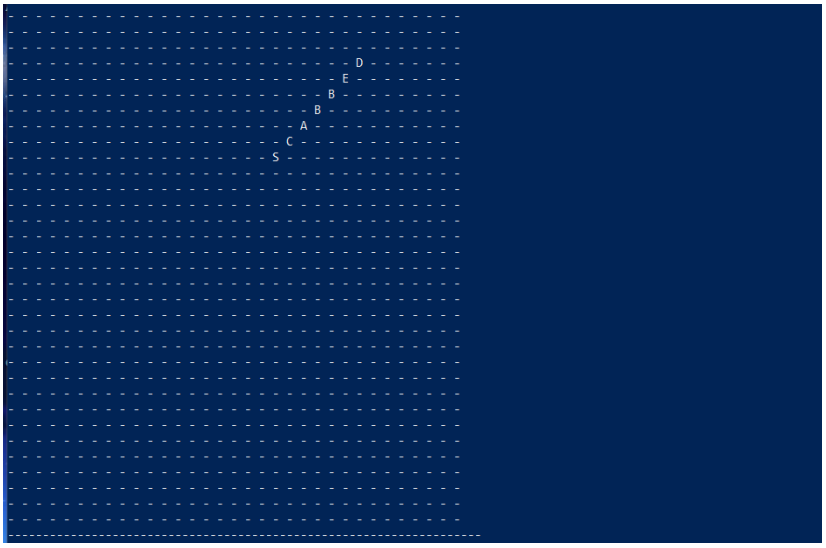

PLATTER


POISON
DE

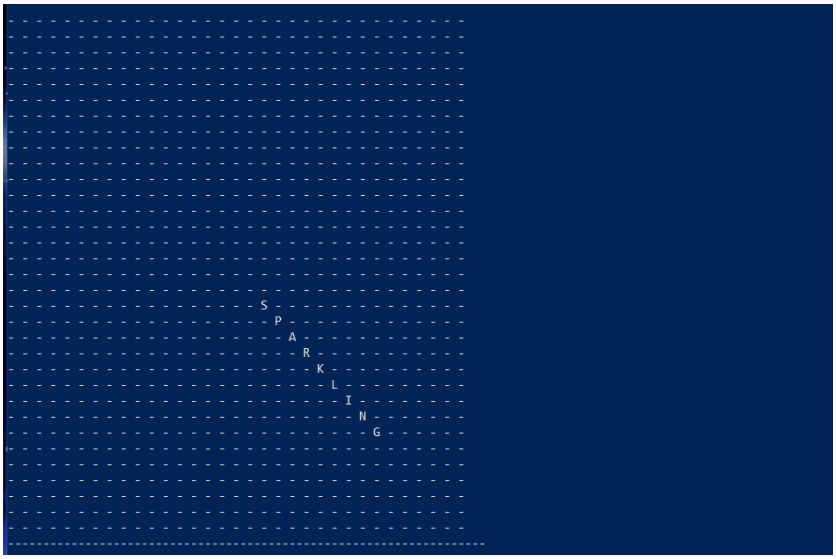

GNITOUQ

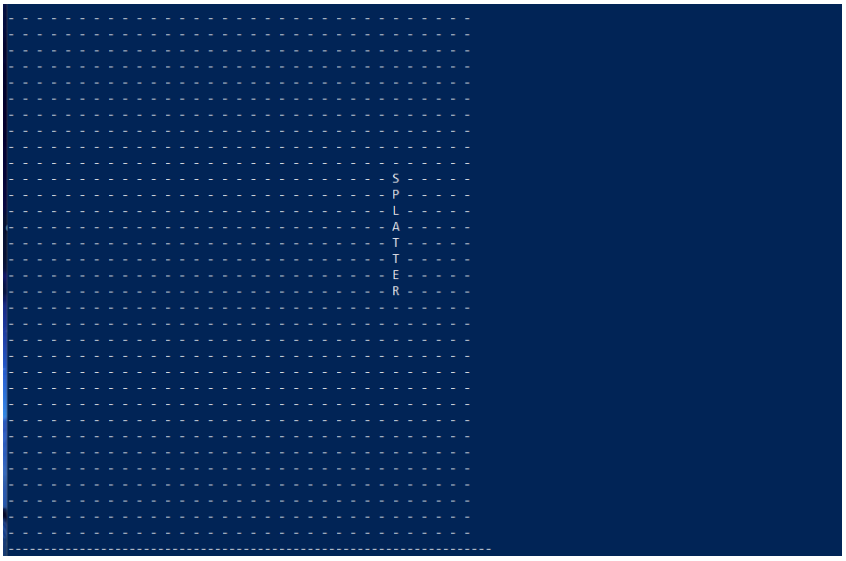

T
N
A
N
G
U
P
E
R

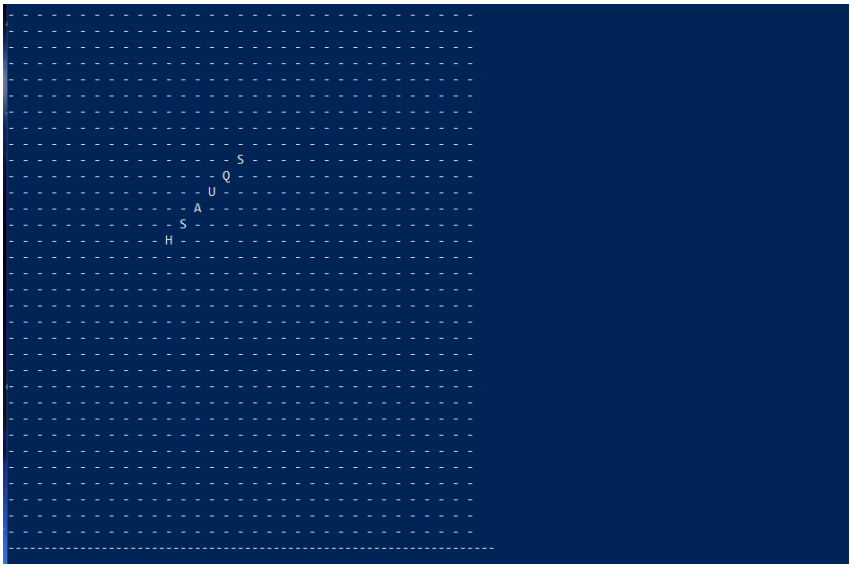

N
A
M
S
E
L
A
S

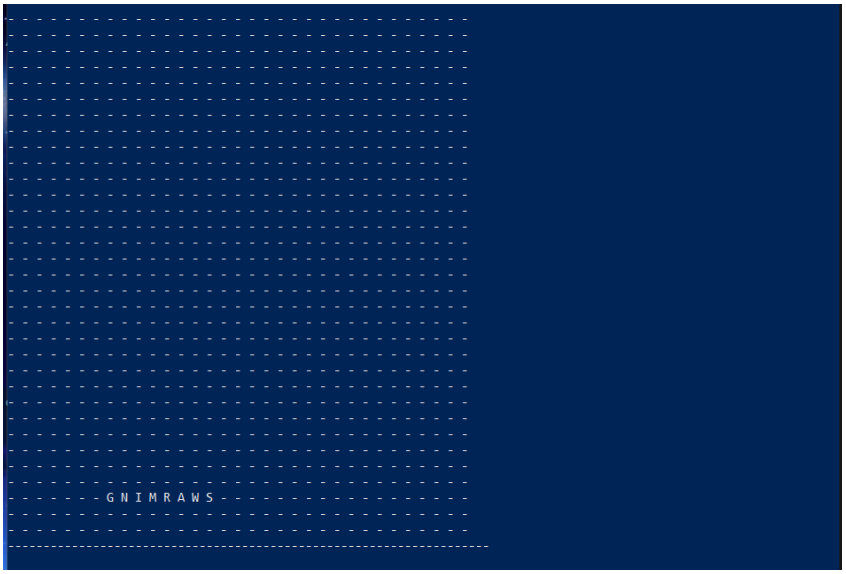

D
E
B
B
A
C
S

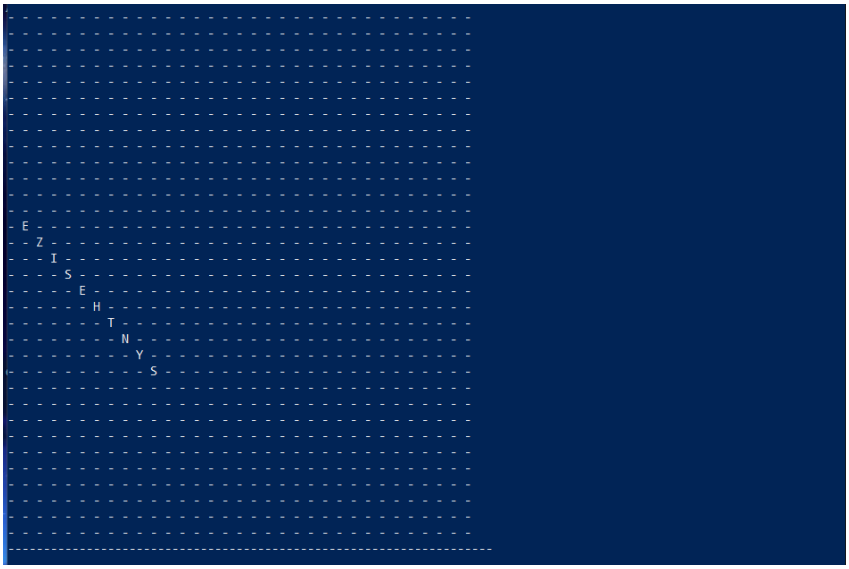

SEEMED

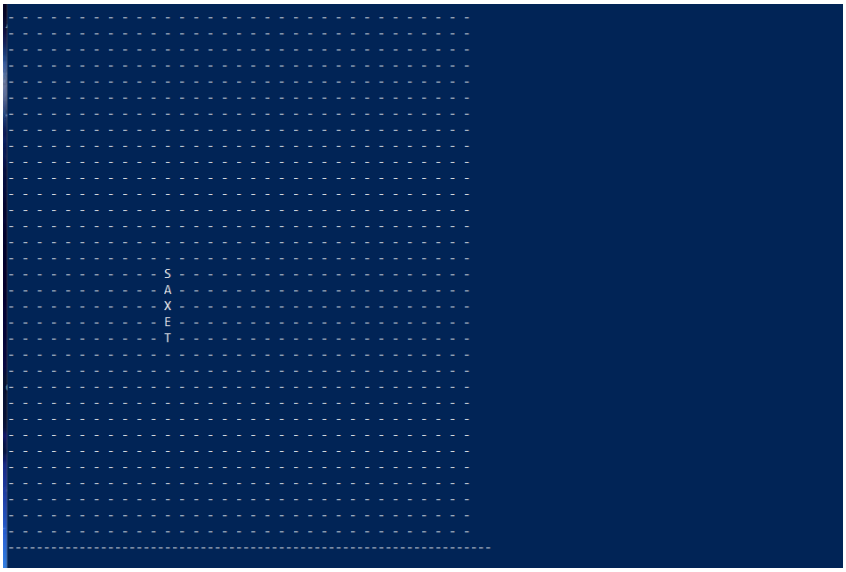

S
P
A
R
K
L
I
N
G

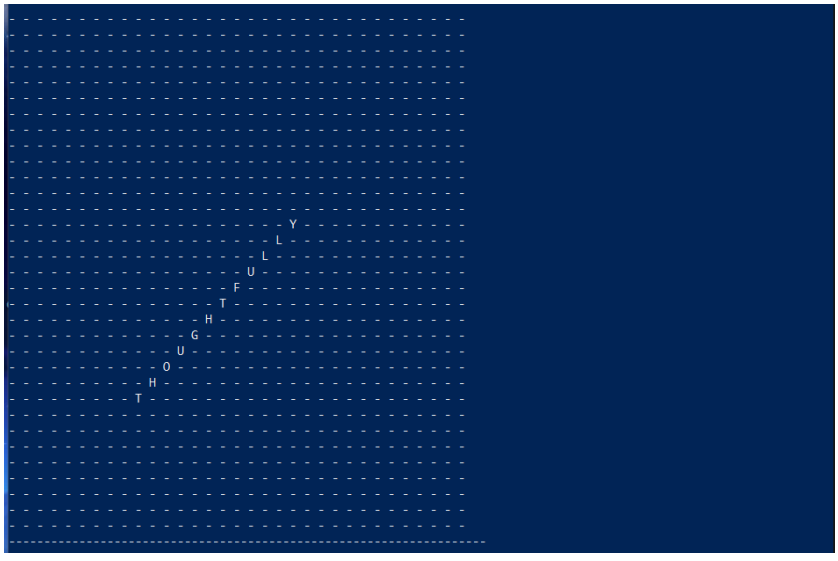

SPLATTER

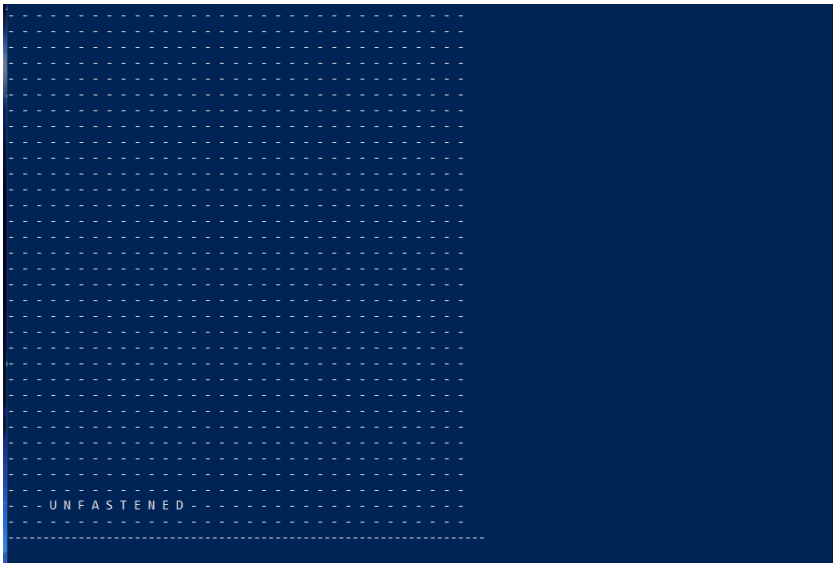

SQUASH

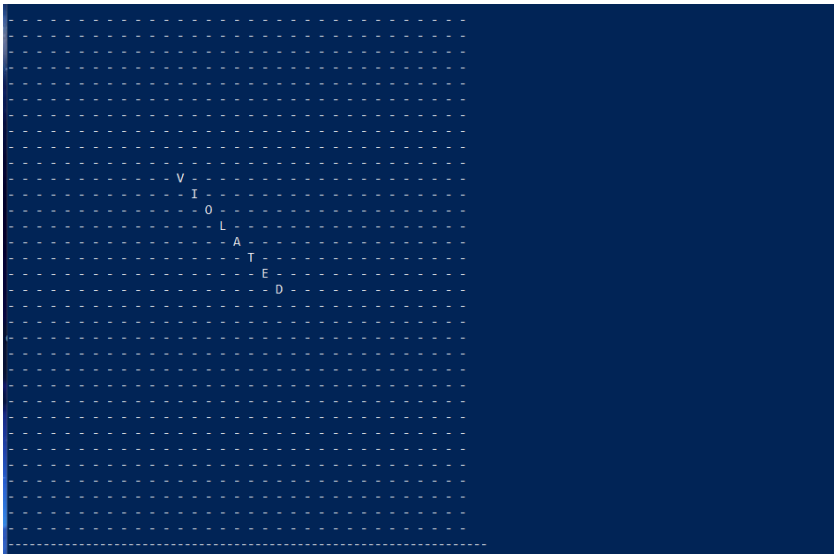

GNIMRAWS

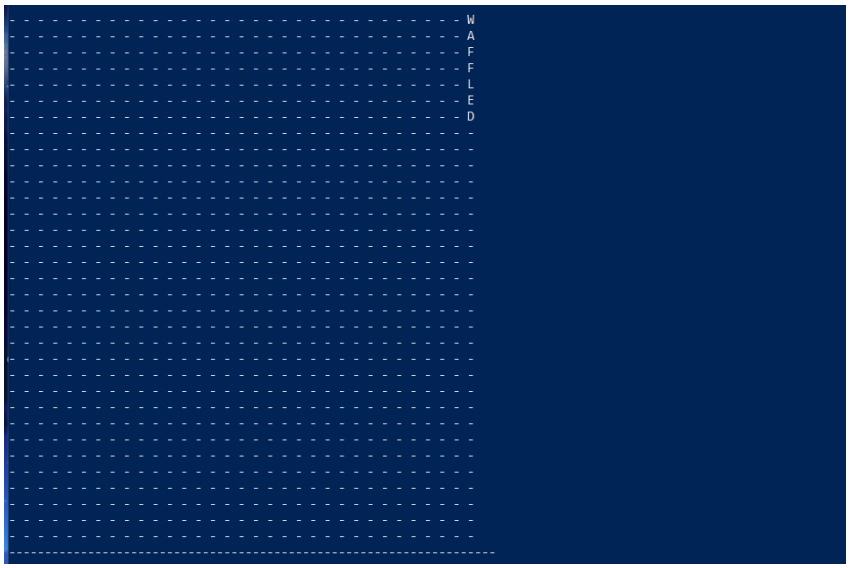


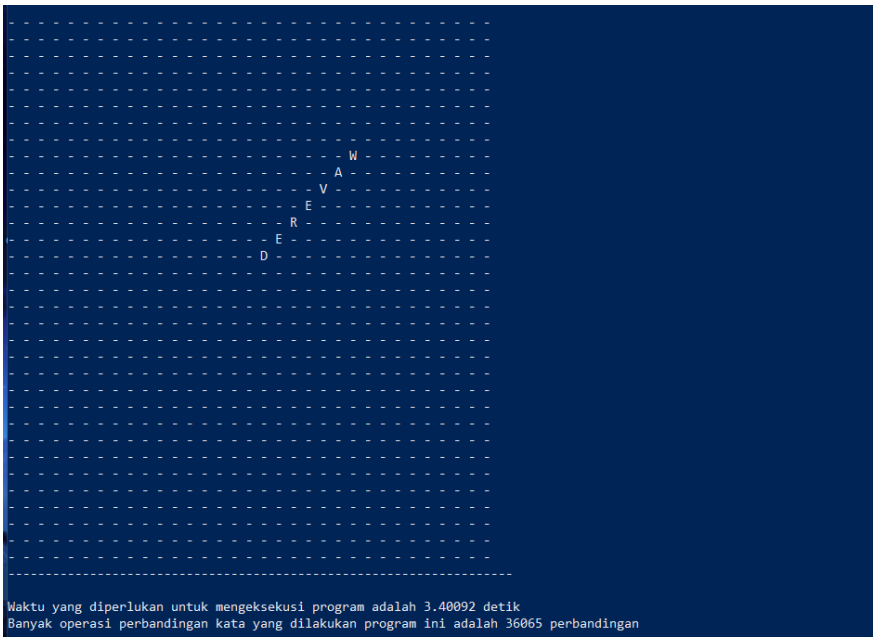





UNFASTENED


VIOLATED


WAFFLED


W
A
V
E
R
E
D
Waktu yang diperlukan untuk mengeksekusi program adalah 3.40092 detik
Banyak operasi perbandingan kata yang dilakukan program ini adalah 36065 perbandingan

Test Case 8 (Large)

Input

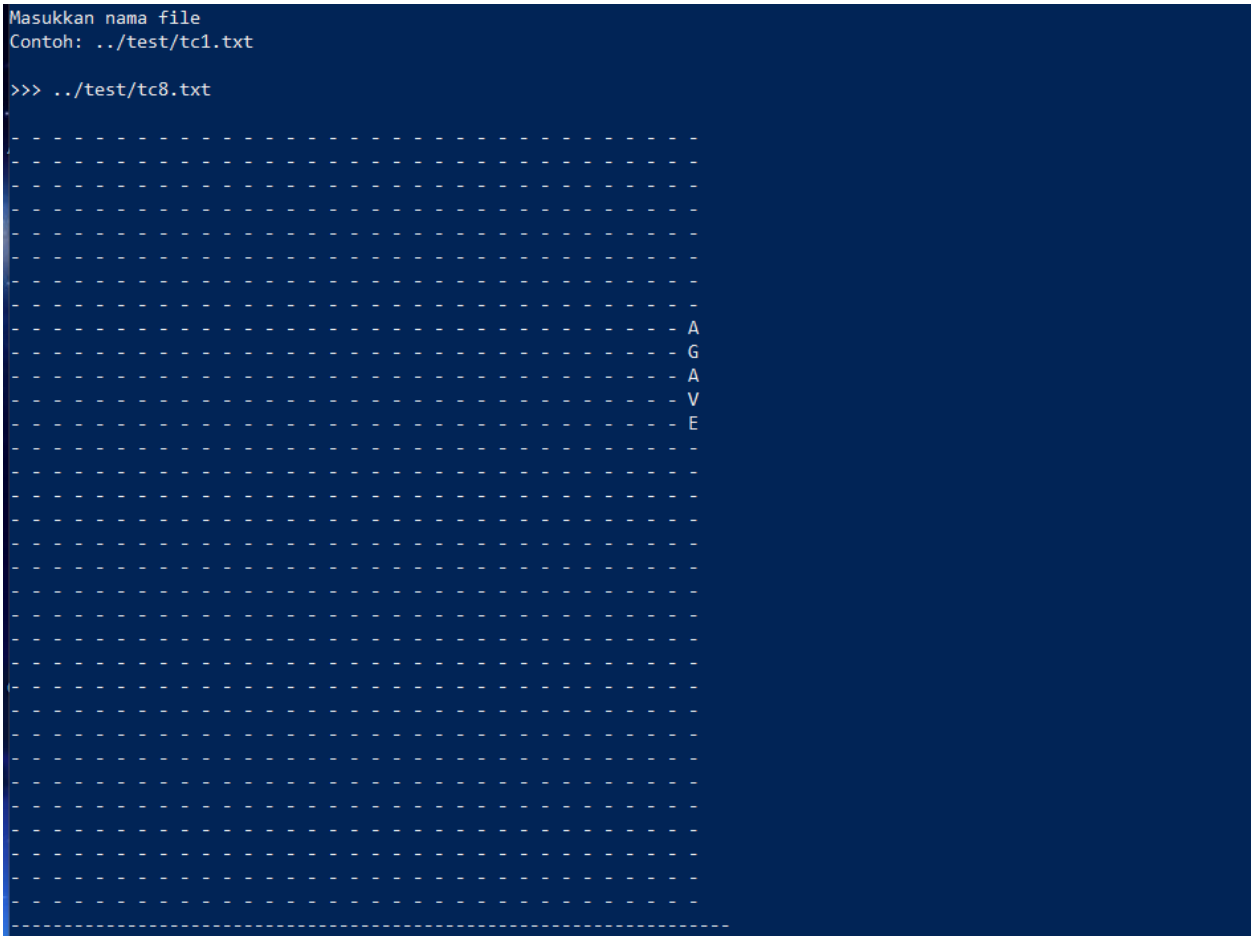
tc8.txt - Notepad

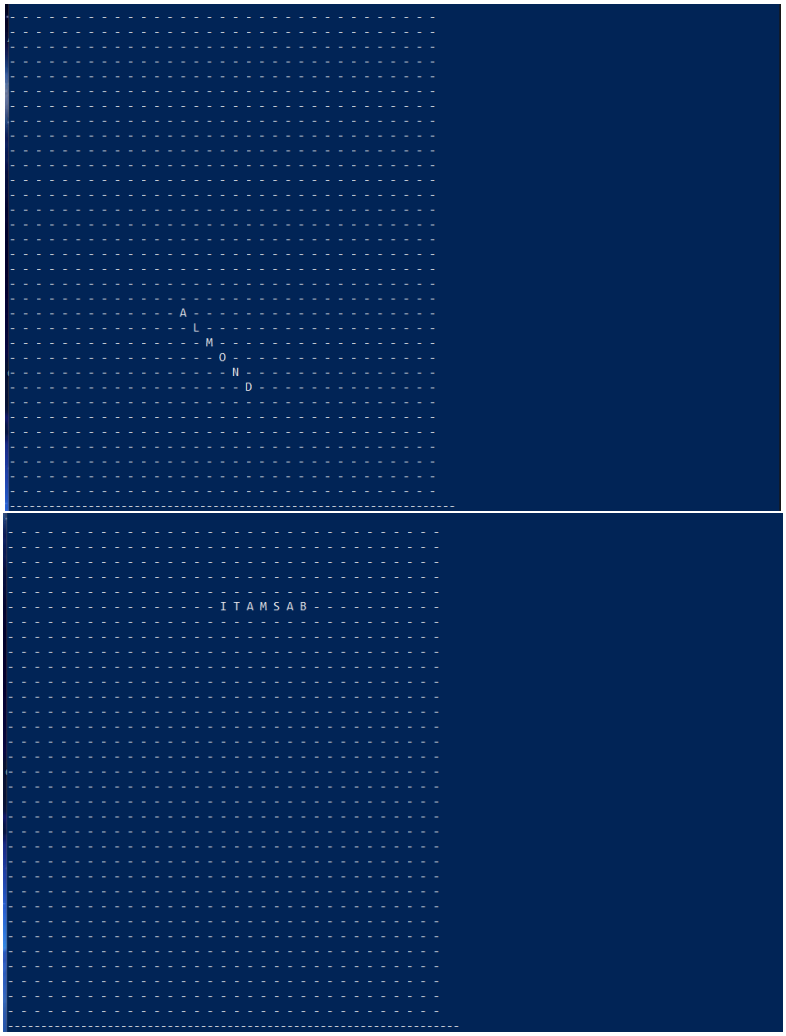
File Edit Format View Help

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

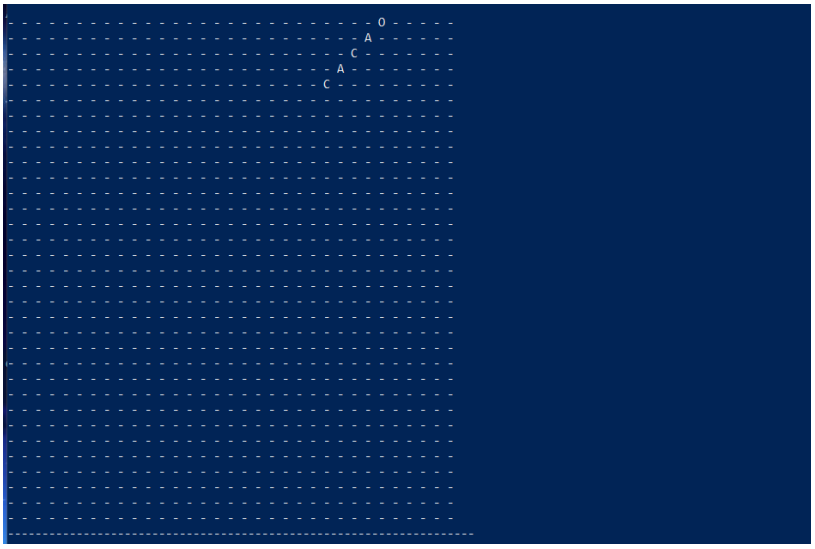
```
AGAVE
ALMOND
BASMATI
BOKCHOY
BREAD
CACAO
CAPSICUM
CASHEW
CHIVES
CUCUMBER
DRAGONFRUIT
GINGER
GRAPE
LEMONGRASS
OATMEAL
OYSTER
PASTA
POTATO
QUINOLA
RADISH
WATERMELON
```

Output

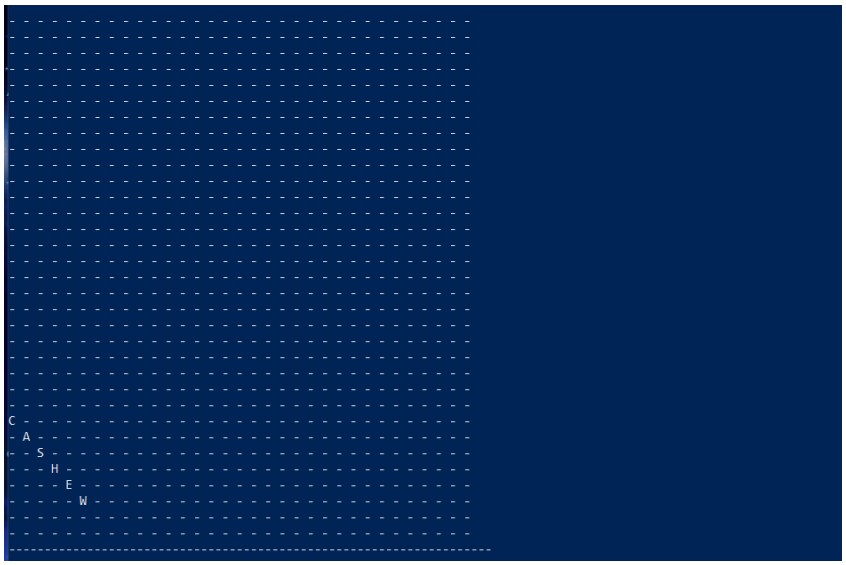


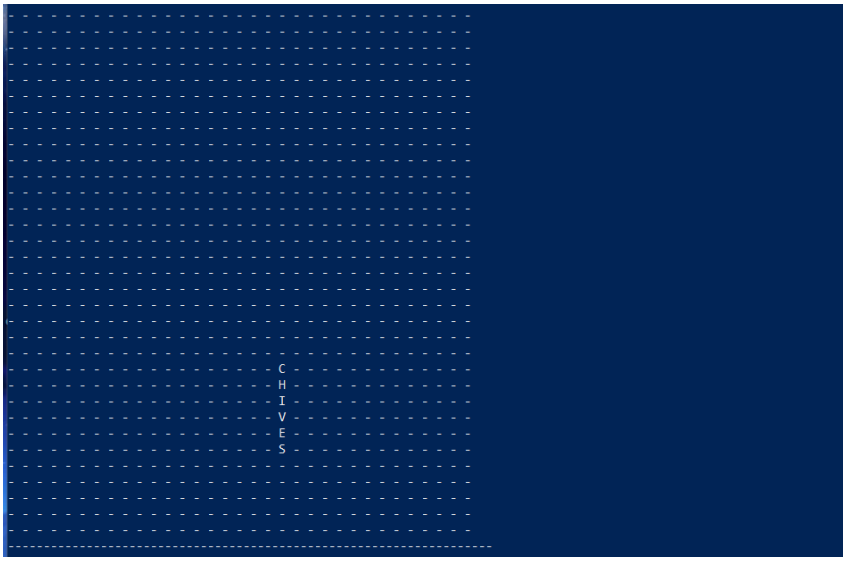

A
L
M
O
N
D
ITAMSAB
Y
O
H
C
K
O
B


D
A
E
R
B

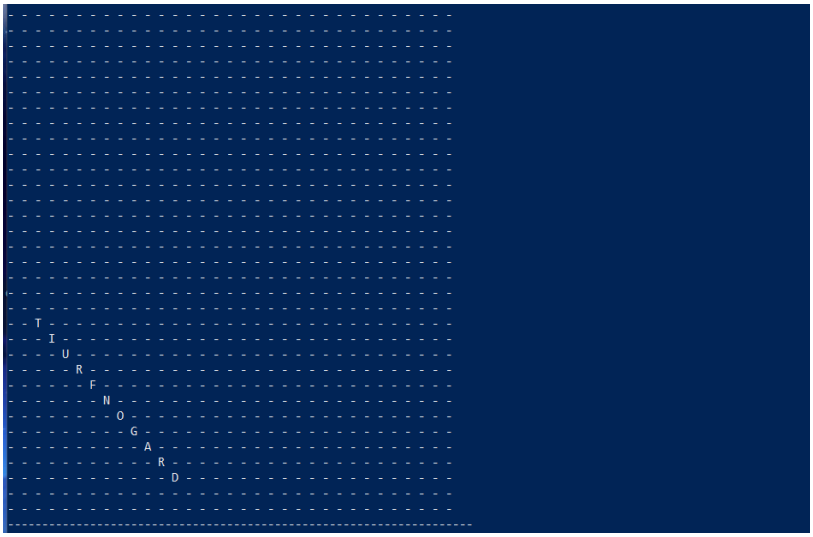

O
A
C
A
C

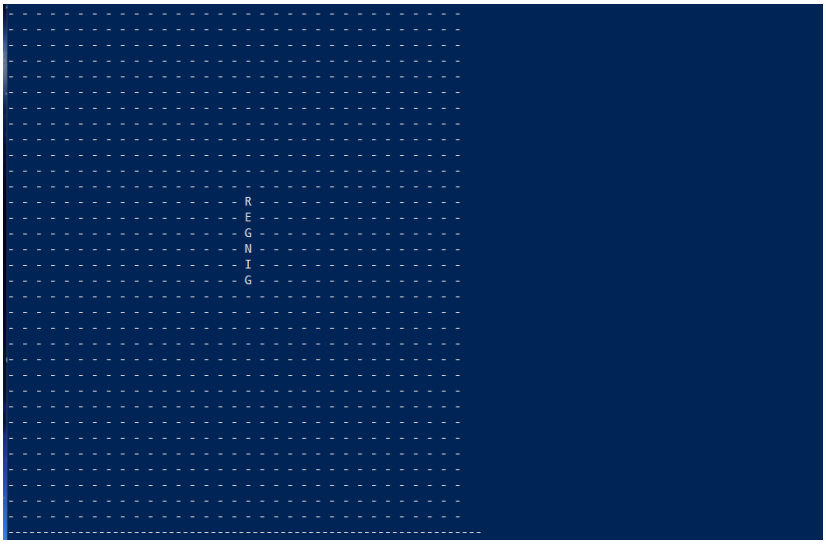

CAPSICUM

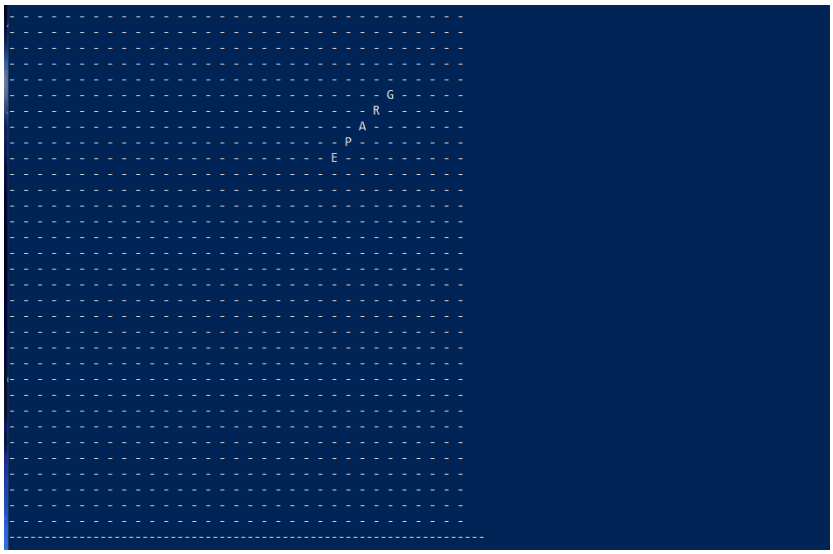

CASHEW


CHIVES

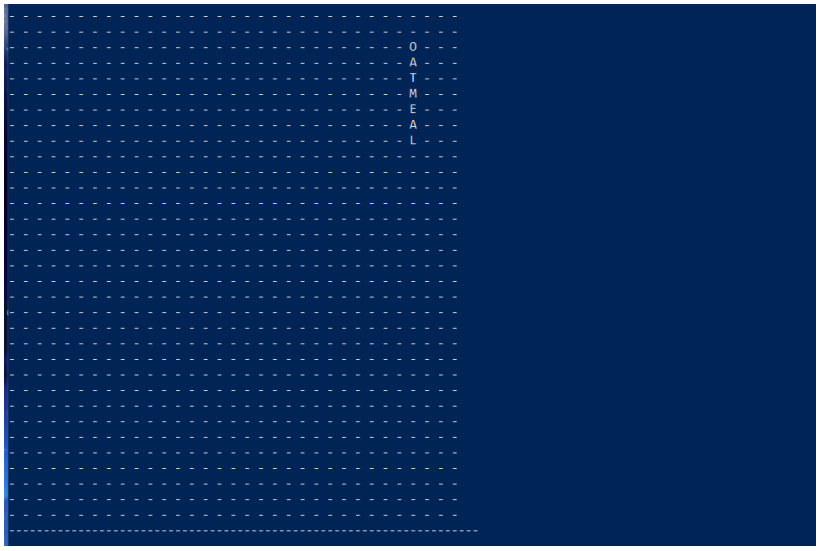

REBMUCUC

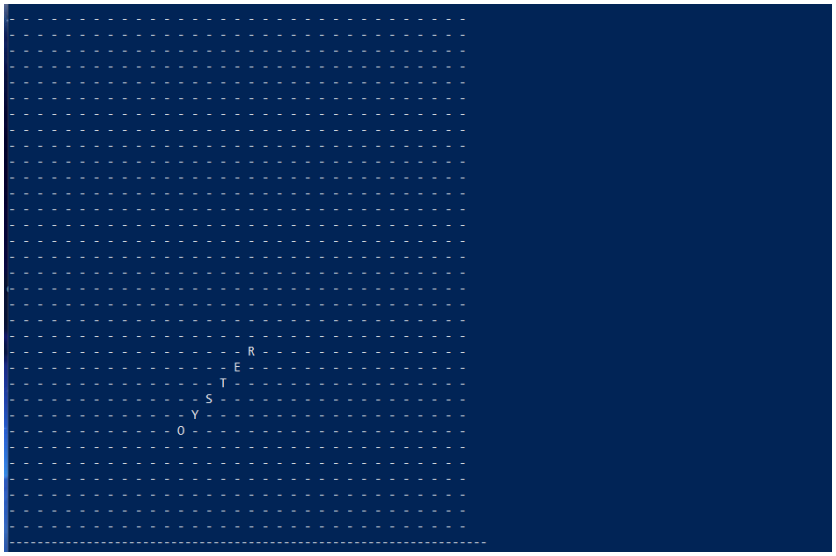

TIUR
FNOGA
RD

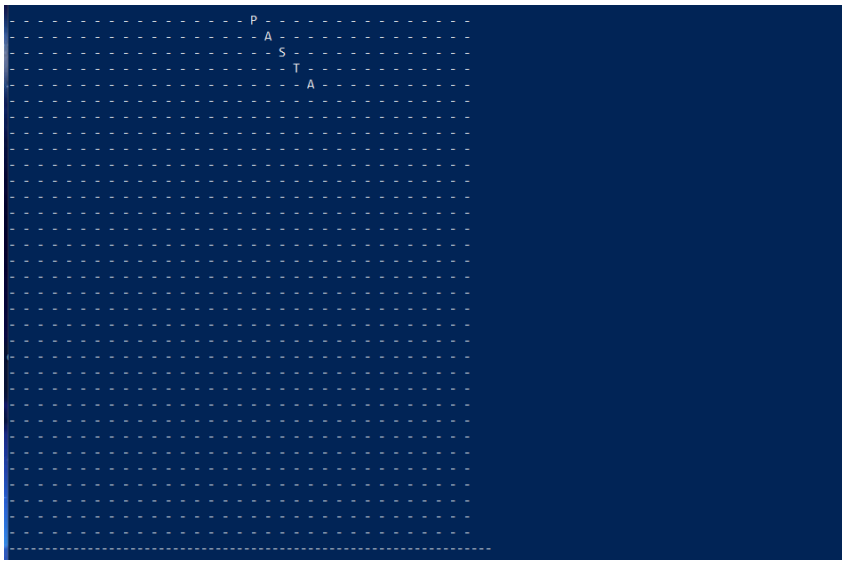

REGNIG

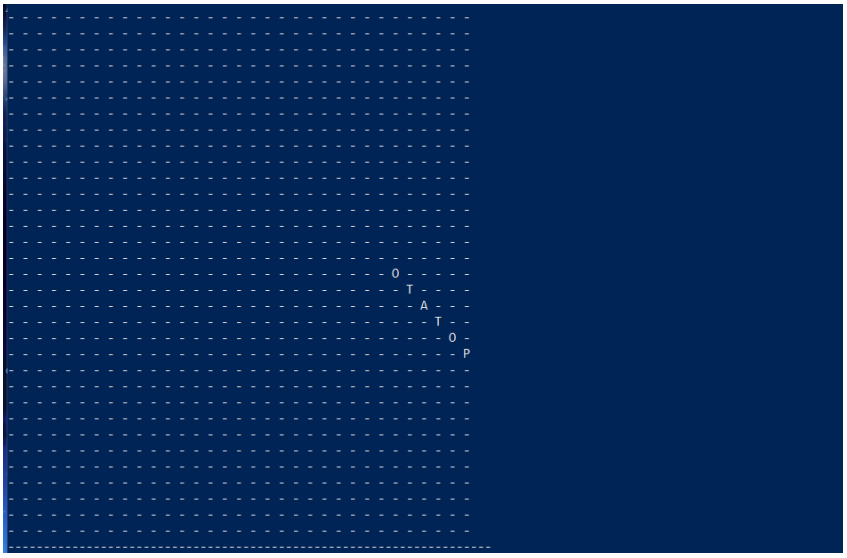

GRAPE

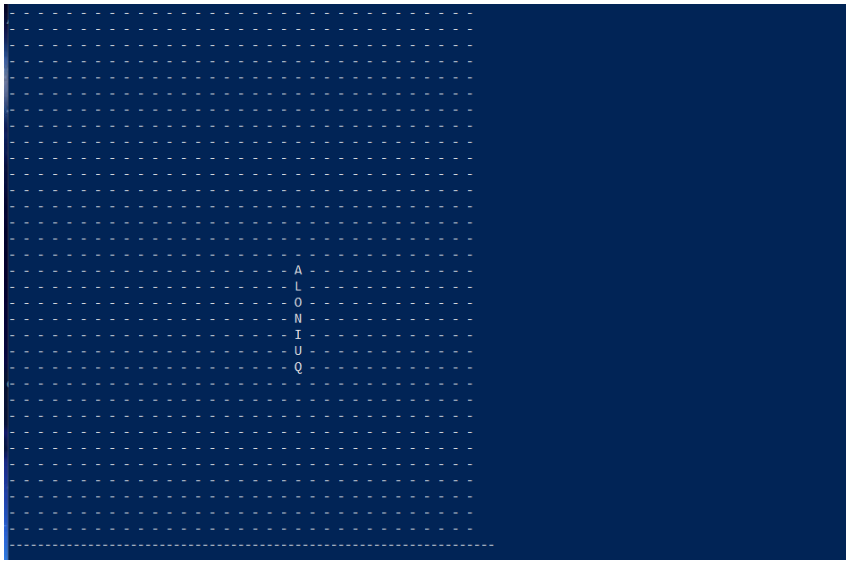

SSARGNOMEL

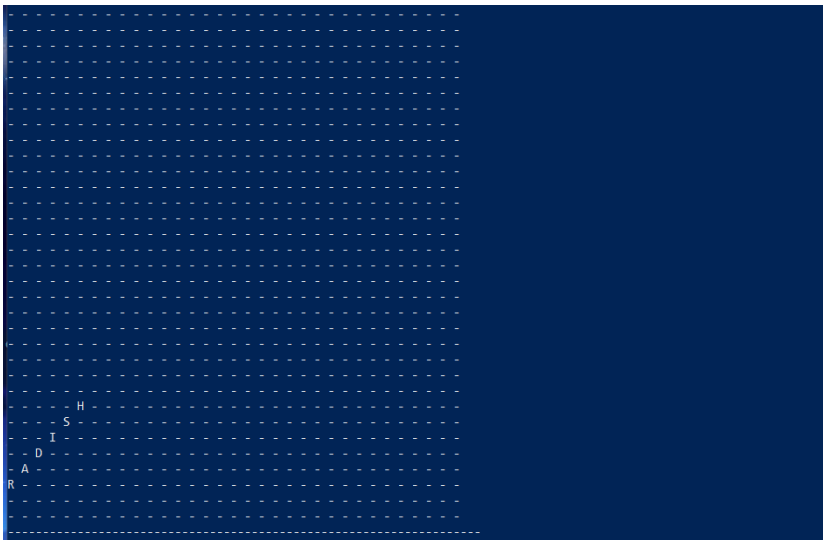

OATMEAL

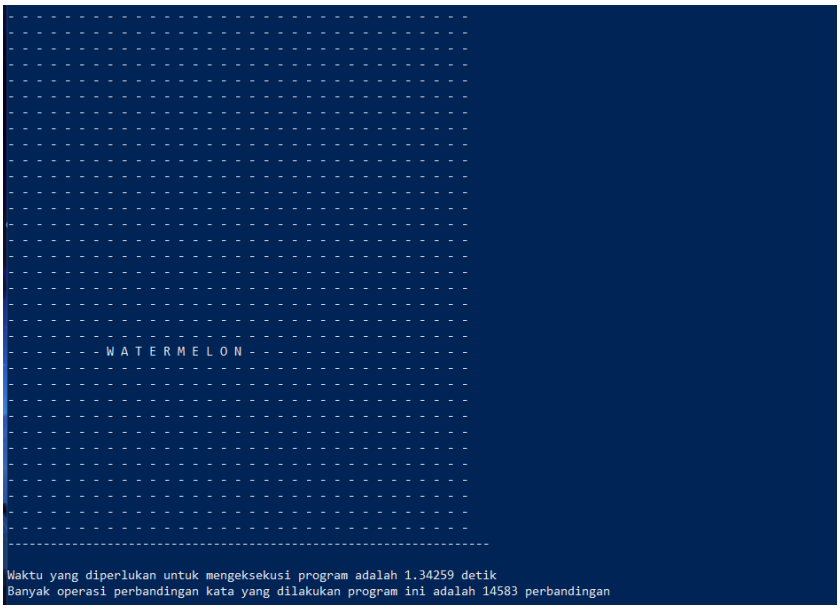

R
E
T
S
Y
O


PASTA


OTATOP

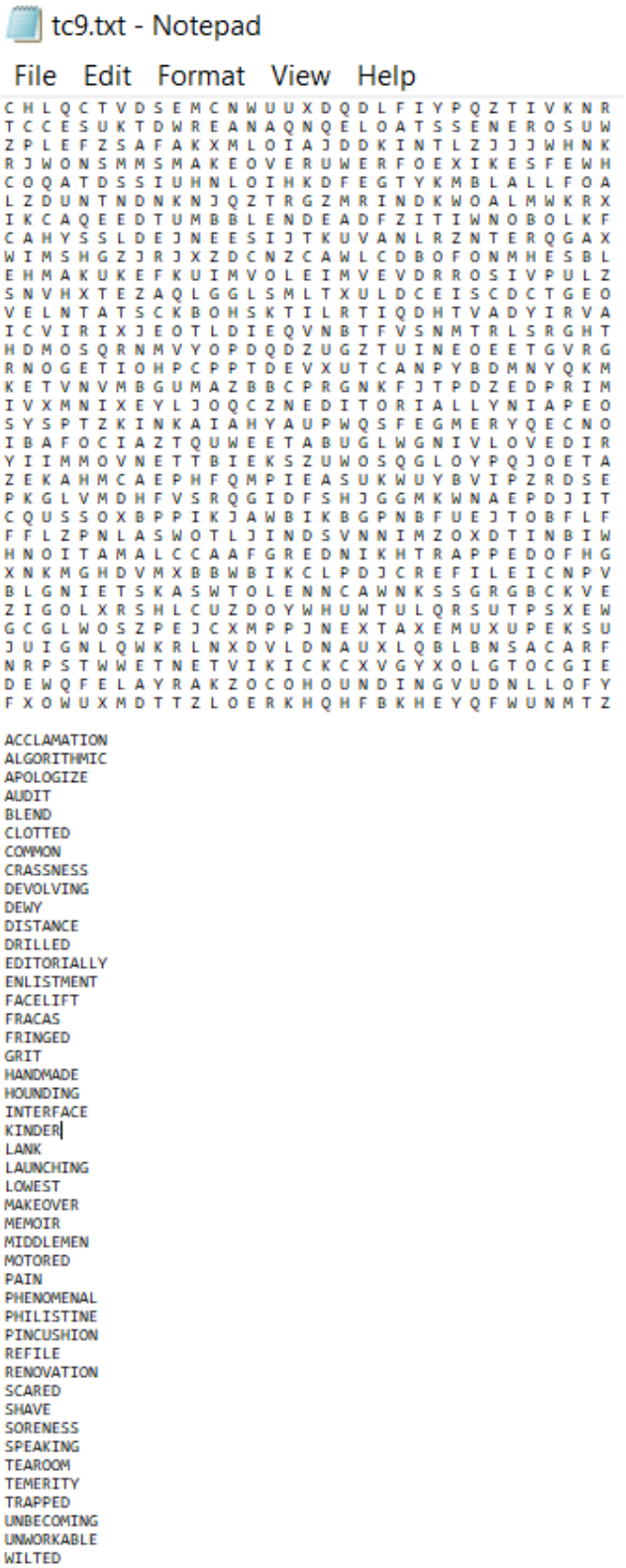

ALONIUQ


H
S
I
D
A
R

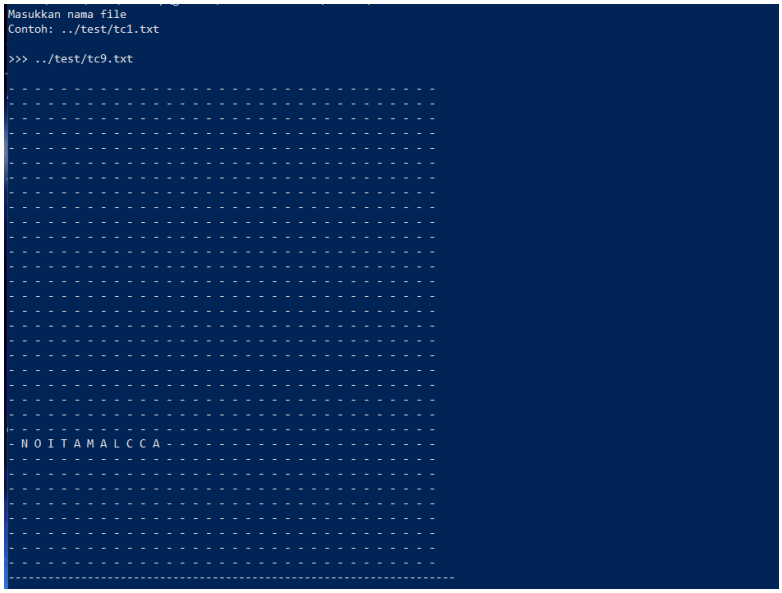

W A T E R M E L O N
Waktu yang diperlukan untuk mengeksekusi program adalah 1.34259 detik
Banyak operasi perbandingan kata yang dilakukan program ini adalah 14583 perbandingan

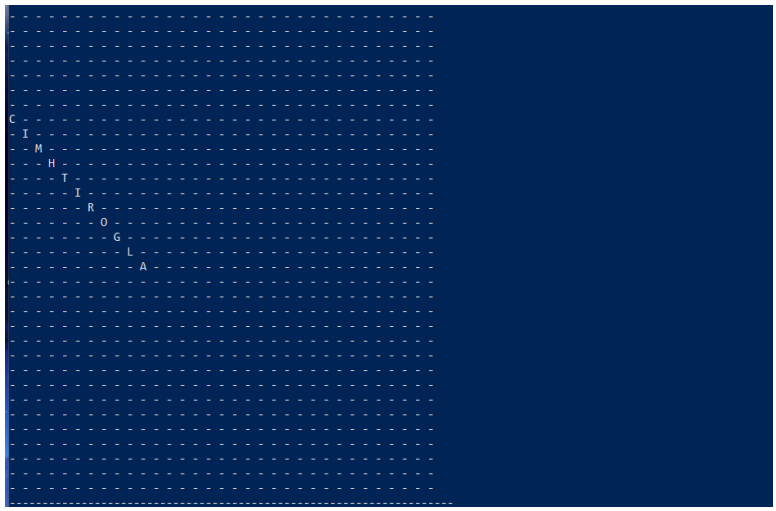
Test Case 9 (Large)

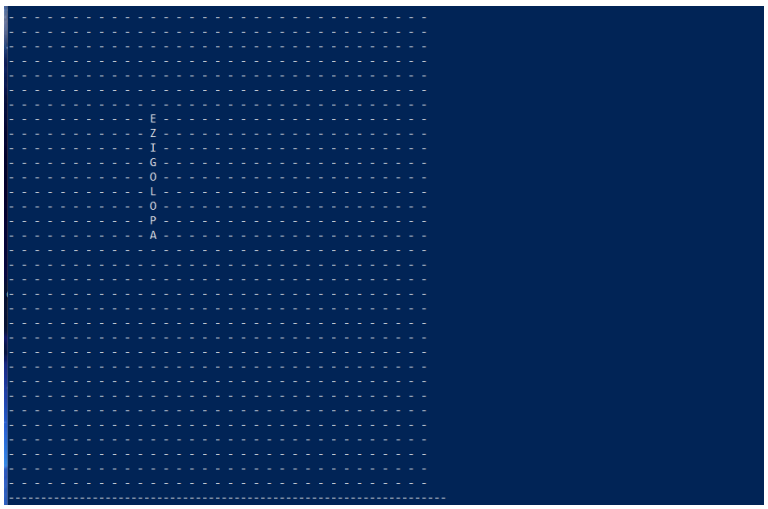
Input

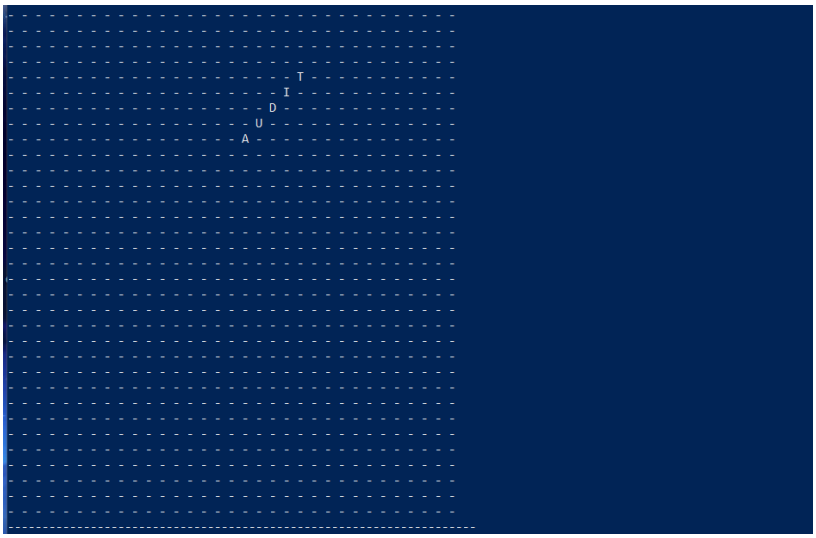


Output


Masukkan nama file
Contoh: ../test/tc1.txt
>>> ../test/tc9.txt
N O I T A M A L C C A


C
I
M
H
T
I
R
O
G
L
A

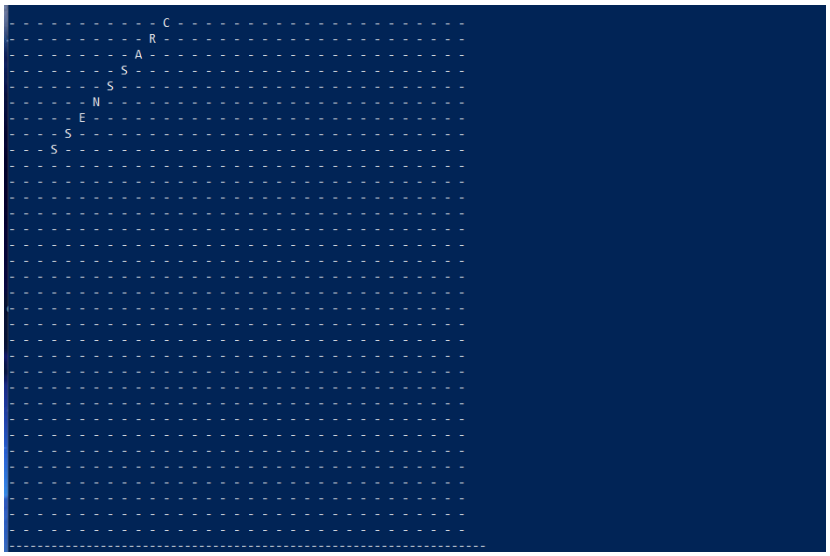

E
Z
I
G
O
L
O
P
A


T
I
D
U
A

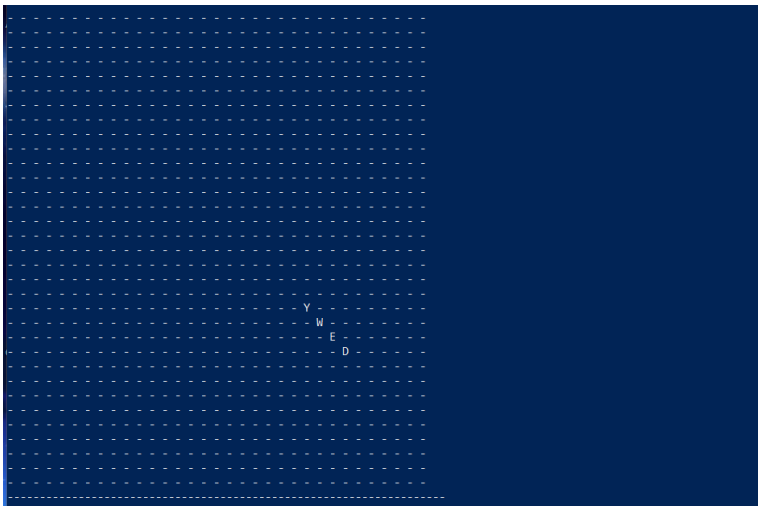

BLEND


C
L
O
T
T
E
D

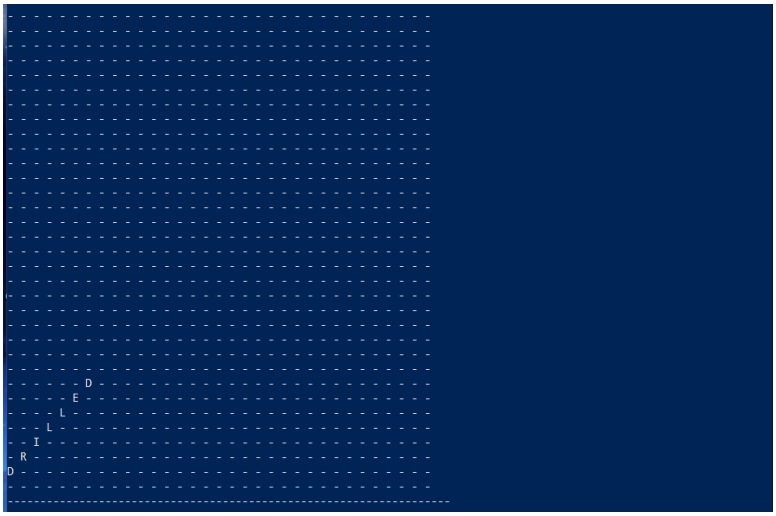

C
O
M
M
O
N

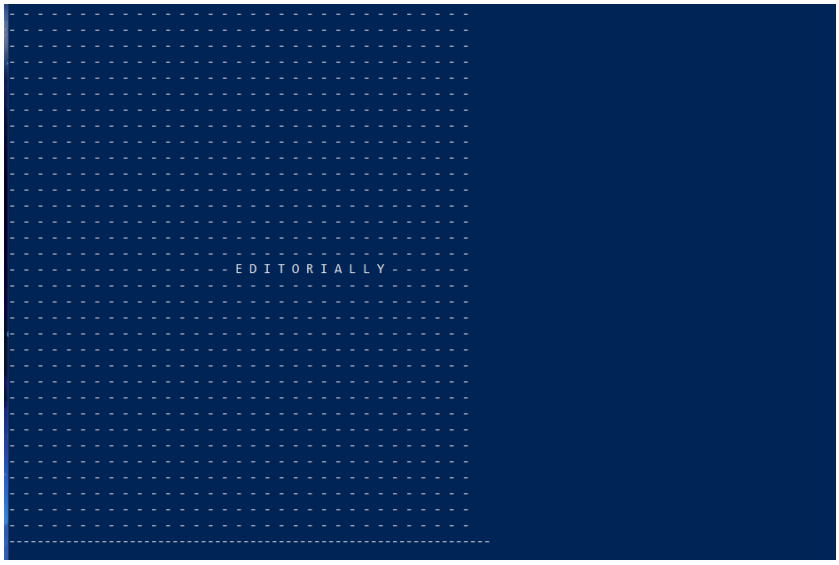

C
R
A
S
S
N
E
S
S


G N I V L O V E D

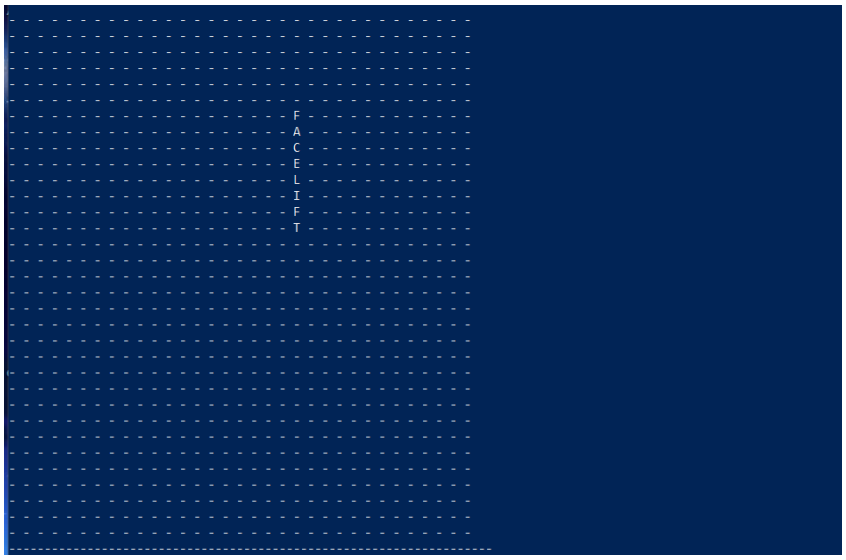

Y
W
E
D


E
C
N
A
T
S
I
D

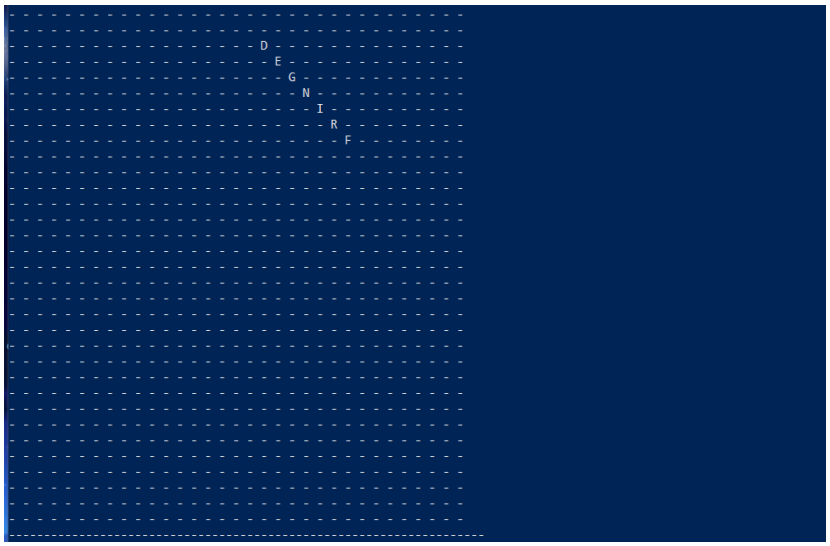

D
E
L
L
I
R
D

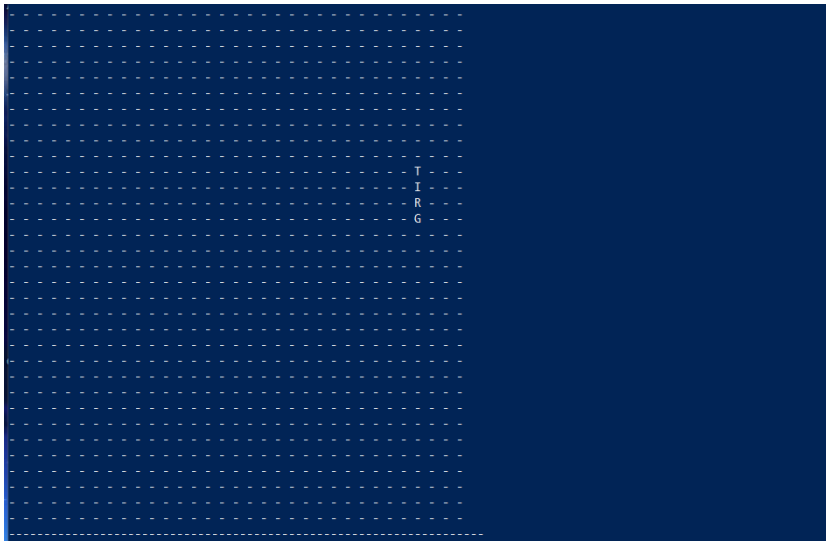

EDITORIALLY


E
N
L
I
S
T
M
E
N
T

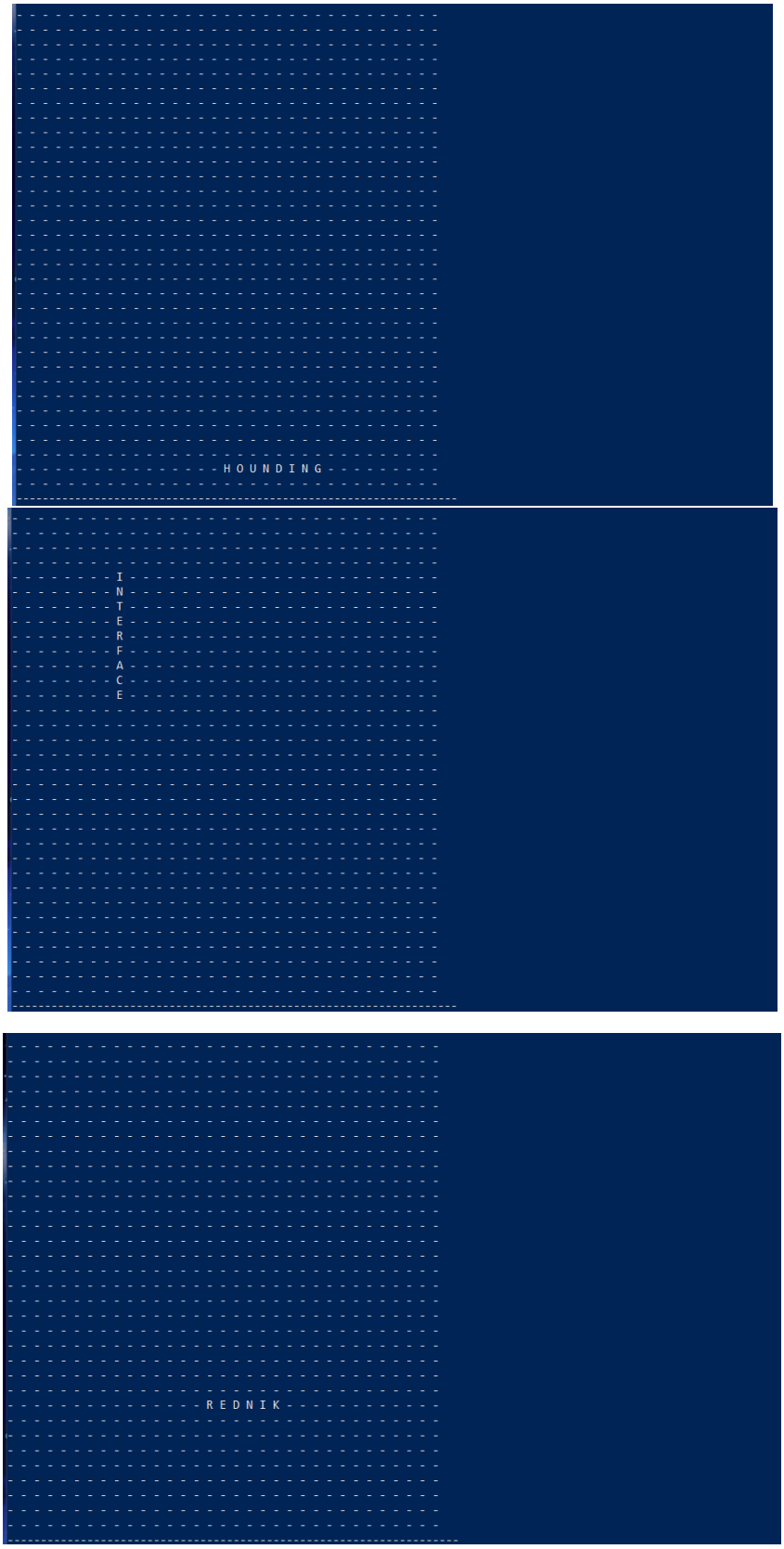

F
A
C
E
L
I
F
T

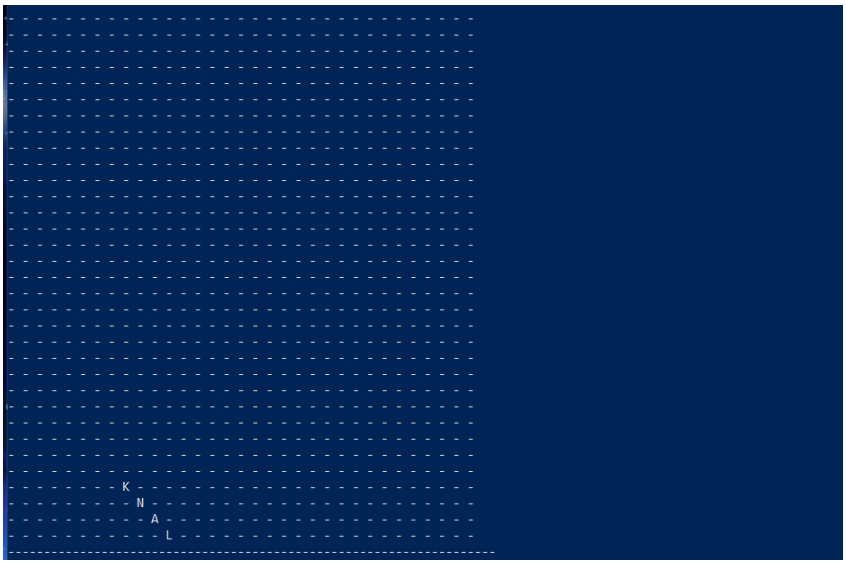

SACARF

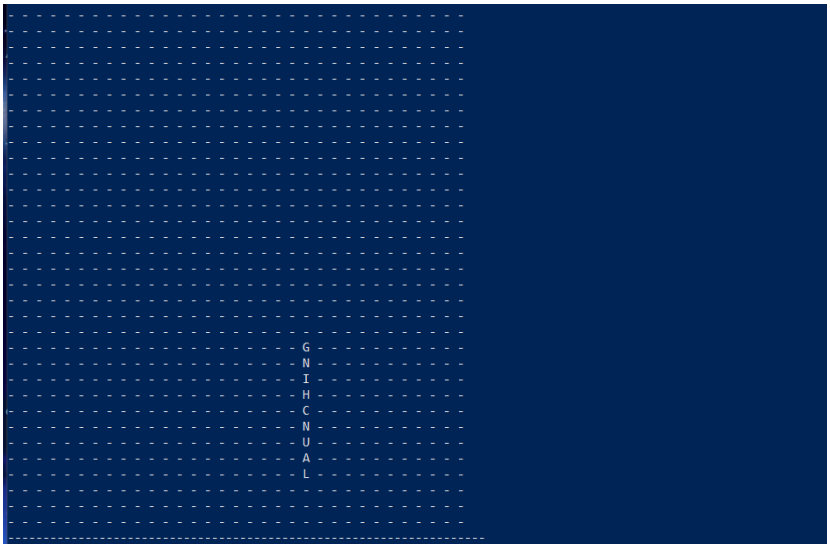

D
E
G
N
I
R
F


T
I
R
G


E
D
A
M
D
N
A
H

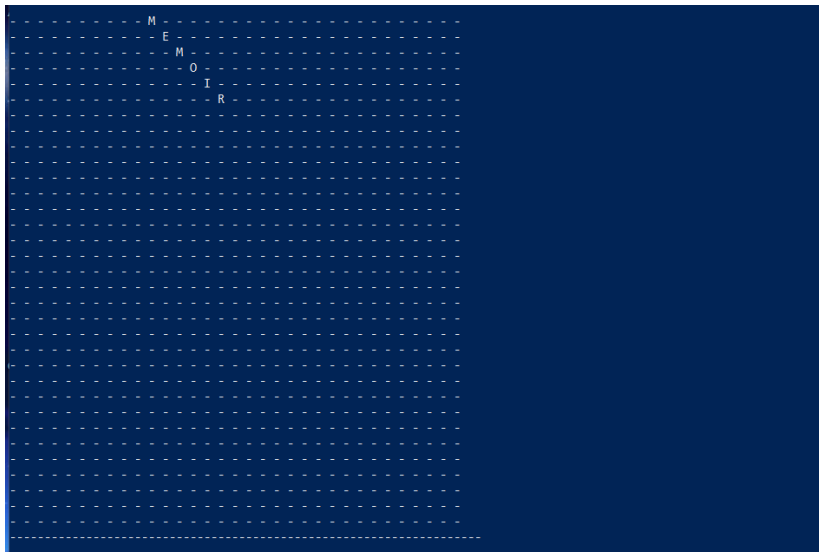

HOUNDING
INTERFACE
REDNIK

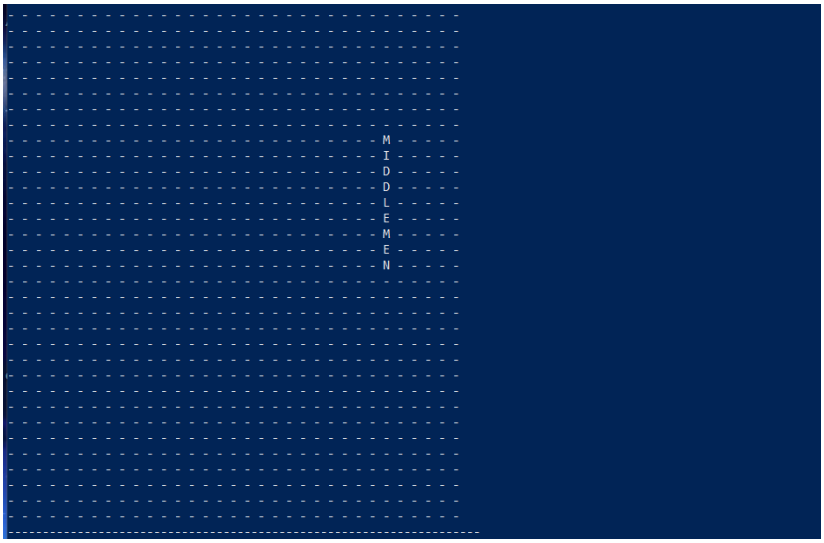

K
N
A
L

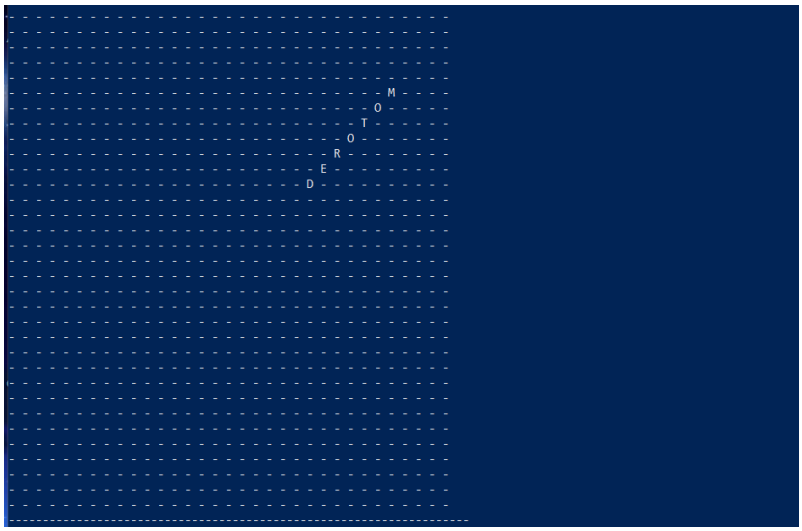

G
N
I
H
C
N
U
A
L


T
S
E
W
O
L

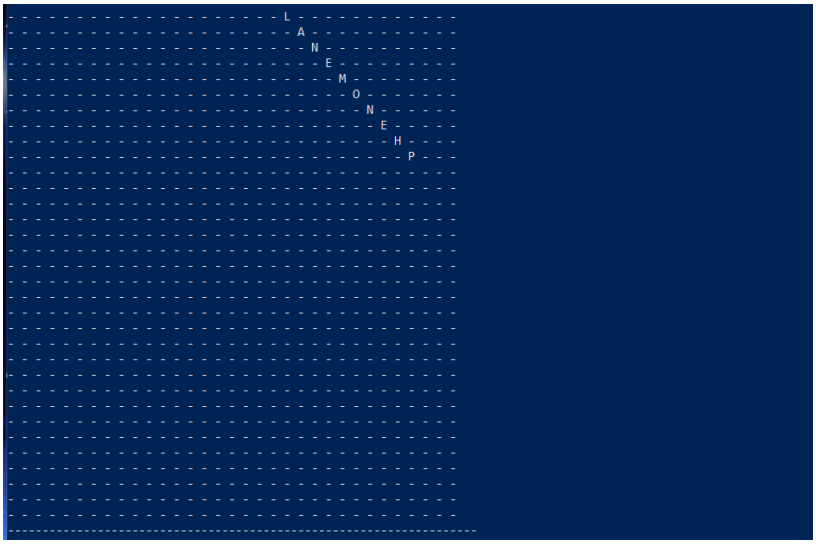

MAKEOVER


M
E
M
O
I
R

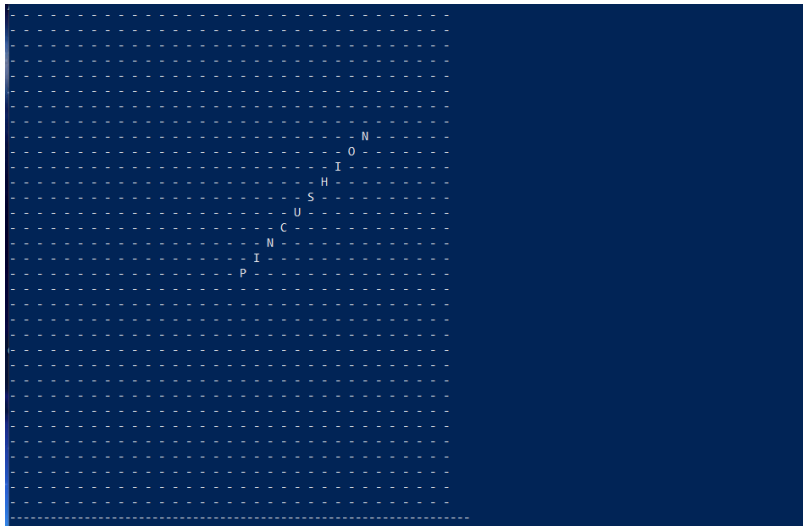

M
I
D
D
L
E
M
E
N


M
O
T
O
R
E
D

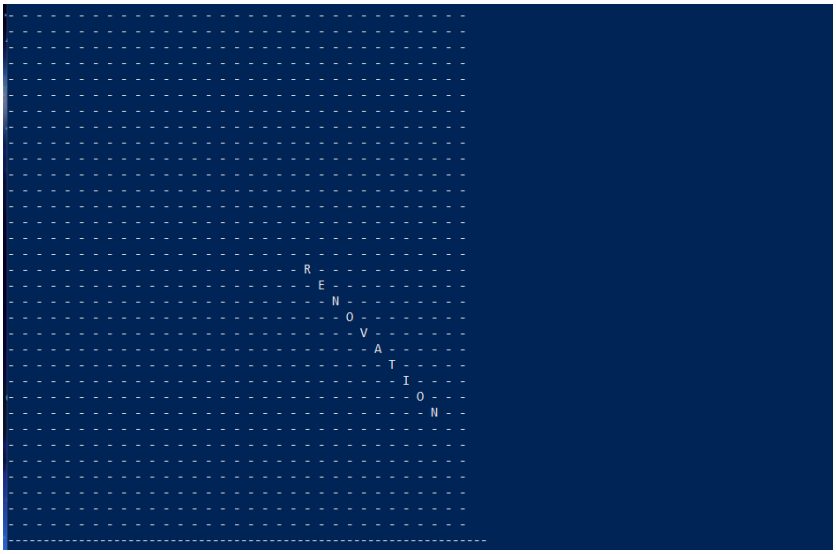

N I A P

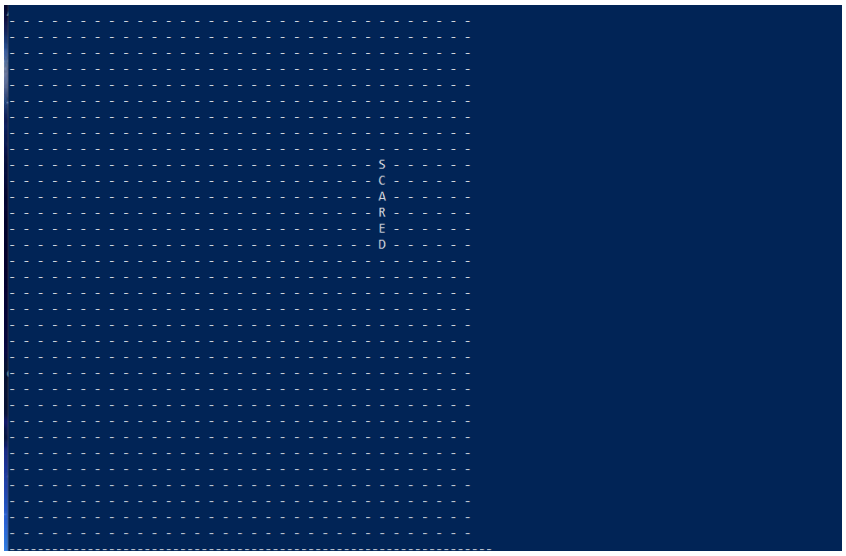

L
A
N
E
M
O
N
E
H
P


E
N
I
T
S
I
L
I
H
P

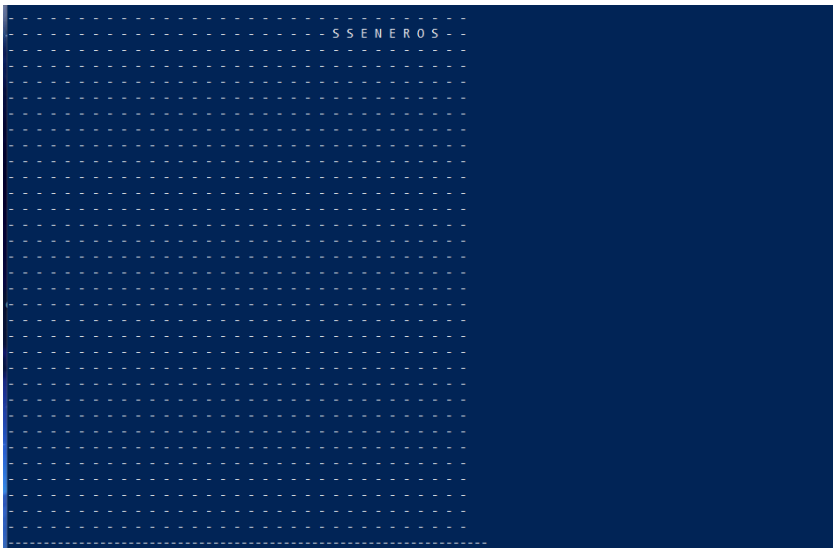

N
O
I
H
S
U
C
N
I
P

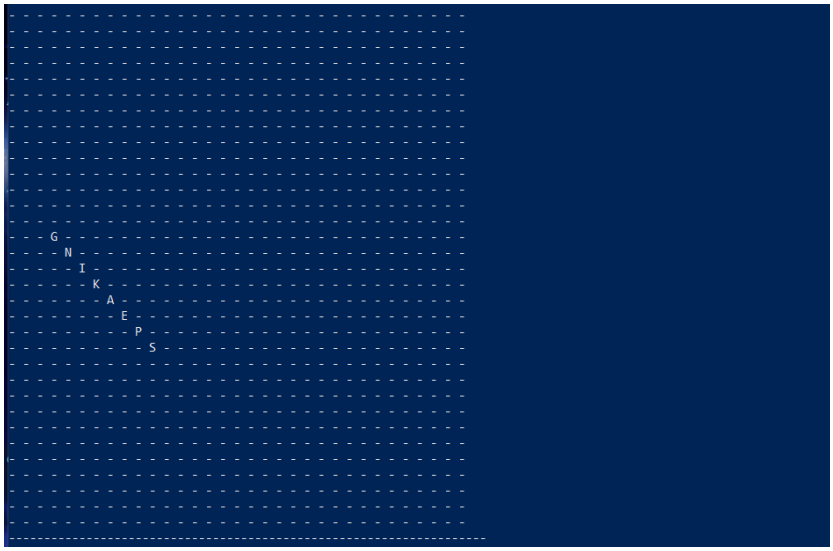

REFILE


RENOVATION

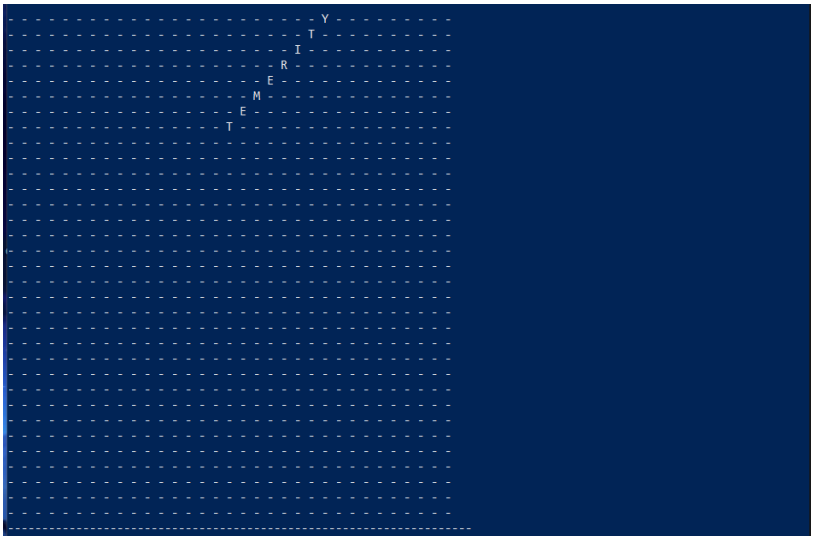

SCARED

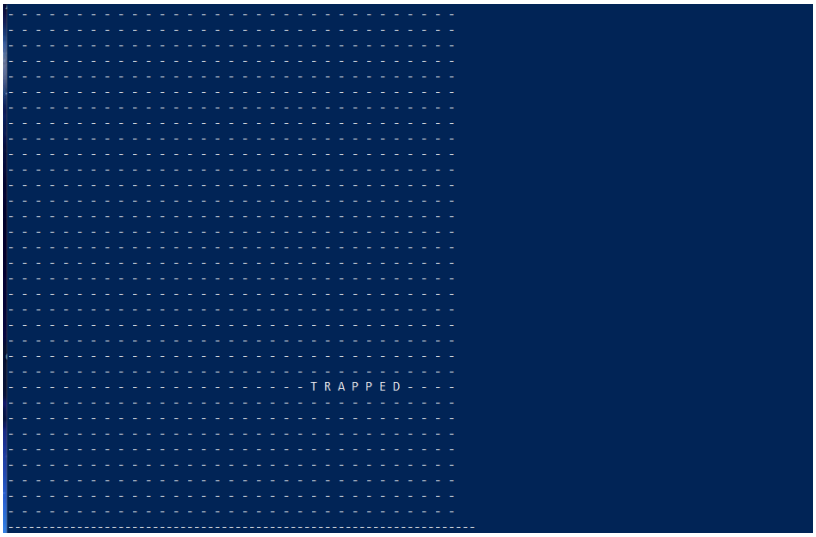

SHAVE

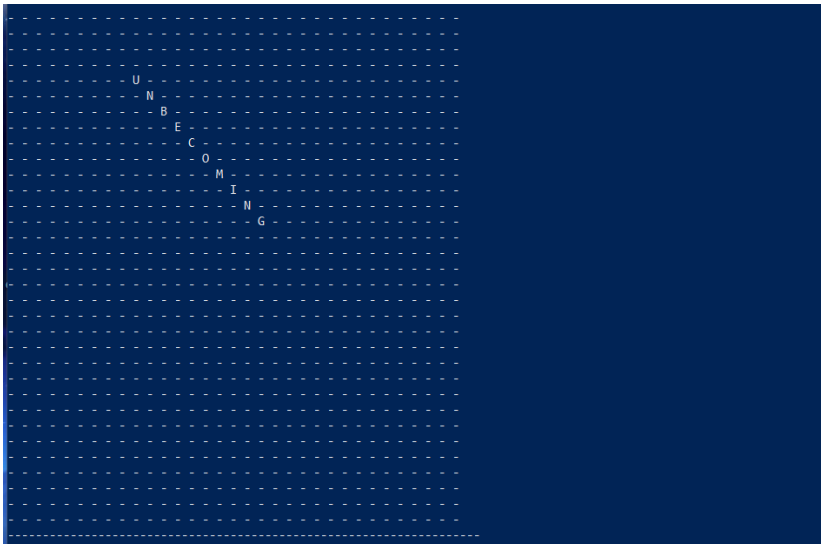

SSENEROS

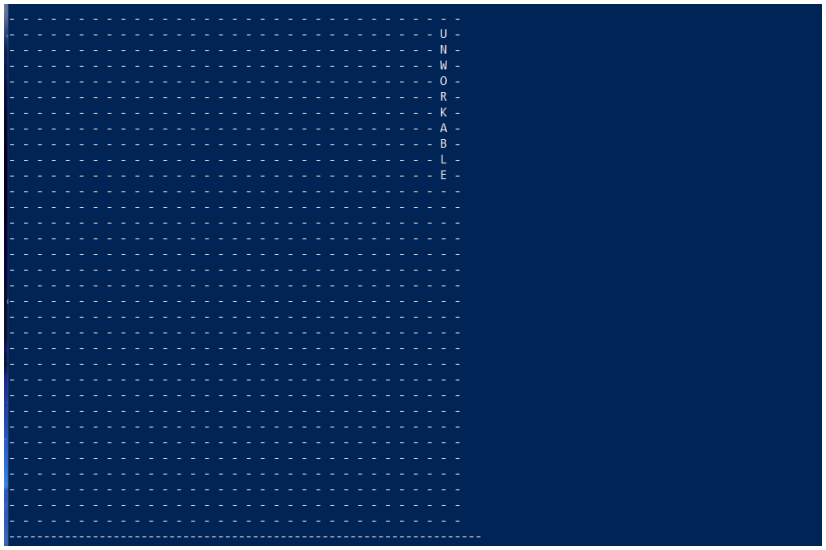

G
N
I
K
A
E
P
S

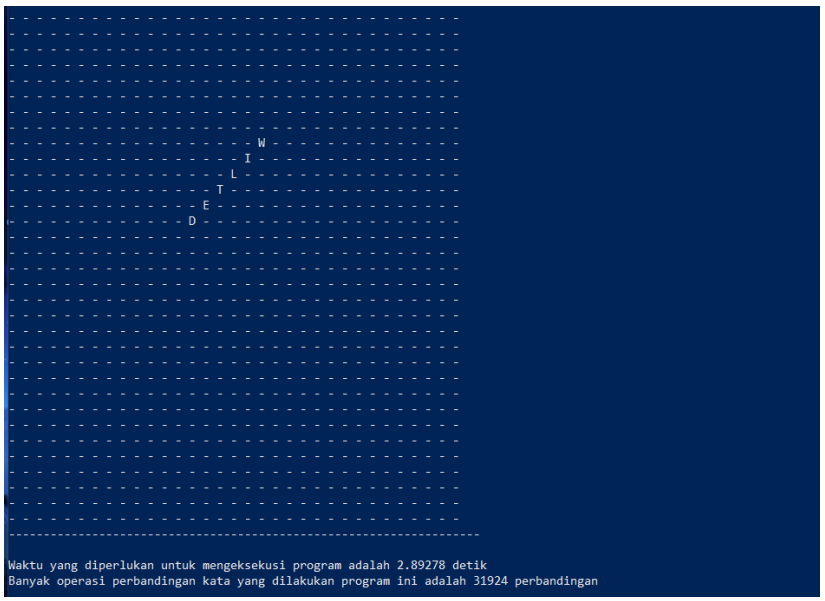

M
O
O
R
A
E
T


Y
T
I
R
E
M
E
T


TRAPPED


U
N
B
E
C
O
M
I
N
G


U
N
W
O
R
K
A
B
L
E


W
I
L
T
E
D
Waktu yang diperlukan untuk mengeksekusi program adalah 2.89278 detik
Banyak operasi perbandingan kata yang dilakukan program ini adalah 31924 perbandingan

BAB V
TABEL PENILAIAN

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.	√	