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

LAPORAN TUGAS KECIL

Diajukan Untuk Memenuhi Tugas Kecil IF2211 Strategi Algoritma

Semester II 2021/2022

Oleh:

Jason Kanggara

13520080

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

A. Algoritma Brute Force

Algoritma *Brute Force* adalah pendekatan yang lempang *(straightforward)* untuk memecahkan suatu persoalan. Biasanya algoritma *brute force* didasarkan pada pernyataan yang terdapat di persoalan dan definisi/konsep yang dilibatkan. Algoritma *brute force* ini memecahkan persoalan dengan sangat sederhana, langsung, dan jelas caranya. Algoritma *brute force* merupakan algoritma yang dapat menyelesaikan hampir segala macam persoalan karena pendekatannya yang langsung. Salah satunya adalah persoalan pada tugas kecil ini, yaitu *Word Search Puzzle*, yang dapat diselesaikan dengan Algoritma *Brute Force*.

Pada persoalan *Word Search Puzzle*, langkah – langkah algoritma yang digunakan adalah sebagai berikut:

1. Menyusun karakter – karakter yang merupakan *puzzle* dari file input ke dalam bentuk array dua dimensi dan kata – kata ke dalam bentuk list.
2. *Looping* setiap kata pada list untuk ditemukan masing – masing kata pada array dua dimensi yang berisi karakter.
3. Menelusuri dari kiri ke kanan pada kata, bandingkan setiap karakter pada kata dengan karakter yang bersesuaian di dalam array dua dimensi sampai:
    a. Semua karakter yang dibandingkan cocok (pencarian berhasil), atau
    b. Ditemukan sebuah ketidakcocokan karakter (pencarian belum berhasil)
4. Bila kata belum ditemukan kecocokannya, geser ke elemen berikutnya pada array dua dimensi lalu ulangi langkah ke – 3.
5. Bila semua elemen sudah dicoba dan masih belum menemukan katanya, ulangi langkah ke – 3 dengan arah yang berbeda. (misalnya: perbandingan dari atas ke bawah, dari kiri ke kanan, secara horizontal, dll).

B. Source Code

Source code dibuat dengan bahasa C++. Program yang dibuat berasumsi bahwa karakter tersusun dalam bentuk persegi maupun persegi panjang dan setiap kata yang ingin dicari dipisahkan oleh *newline*.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <bits/stdc++.h>
#include <time.h>
using namespace std;

#define MAX 100

typedef struct
{
```

```
  char contents[MAX][MAX];
  int rowEff;
  int colEff;
} Matrix;

typedef struct
{
  string patterns[MAX];
  int length;
} List;

#define LENGTH(L) (L).length
#define ListElem(L, i) (L).patterns[i]

#define ROWS(M) (M).rowEff
#define COLS(M) (M).colEff
#define Elem(M, i, j) (M).contents[(i)][(j)]

/*** KONSTRUKTOR ***/
void createMat(Matrix *m)
{
  for (int i = 0; i < MAX; i++)
  {
    for (int j = 0; j < MAX; j++)
    {
      Elem(*m, i, j) = '#';
    }
  }
}

void createList(List *l)
{
  for (int i = 0; i < MAX; i++)
  {
    ListElem(*l, i) = "*";
  }
}
```

```
/*** FUNGSI/PROSEDUR PENDUKUNG ***/
int getLength(List l)
{
  int count = 0;
  for (int i = 0; i < MAX; i++)
  {
    if (ListElem(l, i) != "*")
    {
      count++;
    }
  }
  return count;
}

void displayList(List l)
{
  for (int i = 0; i < getLength(l); i++)
  {
    cout << ListElem(l, i) << endl;
  }
}

int getRow(Matrix m)
{
  int count = 0;
  for (int i = 0; i < MAX; i++)
  {
    if (Elem(m, i, 0) == '#')
    {
      break;
    }
    count++;
  }
  return count;
}

int getCol(Matrix m)
{
```

```
  int count = 0;
  for (int j = 0; j < MAX; j++)
  {
    if (Elem(m, 0, j) != '#')
    {
      count++;
    }
  }
  return count;
}

void displayMat(Matrix m)
{
  for (int i = 0; i < getRow(m); i++)
  {
    for (int j = 0; j < getCol(m); j++)
    {
      cout << Elem(m, i, j) << " ";
    }
    cout << endl;
  }
}

void readFileBox(string filename, Matrix *m) {
  vector<vector<char> > text;
  ifstream myfile;
  myfile.open(filename.c_str());
  string line;

  while (getline(myfile, line))
  {
    vector<char> textData;
    stringstream lineStream(line);

    char value;
    while (lineStream >> value)
    {
      textData.push_back(value);
```

```
    }

    text.push_back(textData);
  }
  int index;
  for (int i = 0; i < text.size(); i++)
  {
    for (int j = 0; j < text[i].size(); j++)
    {
      Elem(*m, i, j) = text[i][j];
    }
  }

}

void readFile(string filename, List *l)
{
  vector<vector<string> > text;
  ifstream myfile;
  myfile.open(filename.c_str());
  string line;

  while (getline(myfile, line))
  {
    vector<string> textData;
    stringstream lineStream(line);

    string value;
    while (lineStream >> value)
    {
      textData.push_back(value);
    }

    text.push_back(textData);
  }
  int index;
  for (int i = 0; i < text.size(); i++)
  {
```

```
    for (int j = 0; j < text[i].size(); j++)
    {
      ListElem(*l, i) = text[i][0];
    }
  }

}

List fixList(List l) {
  List lf;
  createList(&lf);
  int idx;
  for (int i = 0; i < getLength(l); i++) {
    if (ListElem(l,i) == "*") {
      idx = i;
    }
  }

  idx = idx + 1;
  int k = 0;
  for (int i = idx; i < getLength(l) + 1; i++) {
    ListElem(lf,k) = ListElem(l, i);
    k++;
  }

  return lf;
}

/*** PROSEDUR/FUNGSI STRING MATCHING ***/
void PencocokanString(Matrix T, string P, int *perbandinganHuruf)
{
  int col = getCol(T);
  int row = getRow(T);
  int m = P.length();

  int locRow, locCol;
  int iter;
  bool found;
```

```
found = false;
// Horizontal Matching
if (!found) {
  locRow = 0;
  locCol = 0;
  for (int i = 0; i < row; i++) {
    locCol = 0;
    while (locCol <= col - m && !found) {
      iter = 0;
      while (iter < m && P[iter] == Elem(T, i, locCol + iter)) {
        iter++;
        *perbandinganHuruf += 1;
      }

      if (iter == m) {
        found = true;
        locRow = i;
        for (int i = 0; i < row; i++) {
          for (int j = 0; j < col; j++) {
            if (i != locRow) {
              Elem(T,i,j) = '-';
            }
            if (i == locRow) {
              if (j < locCol) {
                Elem(T,i,j) = '-';
              }
              if (j >= m + locCol) {
                Elem(T, i, j) = '-';
              }
            }
          }
        }
      } else {
        locCol++;
        *perbandinganHuruf += 1;
      }
    }
```

```
  }

}

// Alternate Horizonal Matching
if (!found) {
  locRow = 0;
  locCol = col - 1;
  for (int i = 0; i < row; i++) {
    locCol = col - 1;
    while (locCol >= m - 1 && !found) {
      iter = 0;
      while (iter < m && P[iter] == Elem(T, i, locCol - iter)) {
        iter++;
        *perbandinganHuruf += 1;
      }

      if (iter == m) {
        found = true;
        locRow = i;
        for (int i = 0; i < row; i++) {
          for (int j = 0; j < col; j++) {
            if (i != locRow) {
              Elem(T,i,j) = '-';
            }
            if (i == locRow) {
              if (j > locCol) {
                Elem(T,i,j) = '-';
              }
              if (j <= locCol - m) {
                Elem(T,i,j) = '-';
              }
            }
          }
        }
      } else {
        locCol--;
        *perbandinganHuruf += 1;
```

```
      }
    }
  }

}

// Vertical Matching
if (!found) {
  locCol = 0;
  while (locCol < col) {
    locRow = 0;
    while (locRow <= row - m && !found) {
      iter = 0;
      while (iter < m && P[iter] == Elem(T, locRow + iter, locCol)) {
        iter++;
        *perbandinganHuruf += 1;

      }

      if (iter == m) {
        found = true;
        for (int i = 0; i < row; i++) {
          for (int j = 0; j < col; j++) {
            if (j != locCol) {
              Elem(T,i,j) = '-';
            }
            if (j == locCol) {
              if (i < locRow) {
                Elem(T,i,j) = '-';
              }
              if (i >= m + locRow) {
                Elem(T,i,j) = '-';
              }
            }
          }
        }
      } else {
        locRow++;
```

```
        *perbandinganHuruf += 1;

      }
    }
    locCol++;
  }

}

// Alternate Vertical Matching
if (!found) {
  locCol = 0;
  locRow = row - 1;
  while (locCol < col) {
    locRow = row - 1;
    while (locRow >= m - 1 && !found) {
      iter = 0;
      while (iter < m && P[iter] == Elem(T, locRow - iter, locCol)) {
        iter++;
        *perbandinganHuruf += 1;

      }

      if (iter == m) {
        found = true;
        for (int i = 0; i < row; i++) {
          for (int j = 0; j < col; j++) {
            if (j != locCol) {
              Elem(T,i,j) = '-';
            }
            if (j == locCol) {
              if (i > locRow) {
                Elem(T,i,j) = '-';
              }
              if (i <= locRow - m) {
                Elem(T,i,j) = '-';
              }
            }
```

```
            }
          }
        } else {
          locRow--;
          *perbandinganHuruf += 1;

        }
      }
      locCol++;
    }

  }

  // Row Major Diagonal Matching
  if (!found) {
    locRow = 0;
    locCol = 0;
    while (locRow < row) {
      locCol = 0;
      while (locCol <= col - m && !found) {
        iter = 0;
        while (iter < m && P[iter] == Elem(T, locRow + iter, locCol +
iter)) {
          iter++;
          *perbandinganHuruf += 1;

        }
        if (iter == m) {
          found = true;
          for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
              if (i < locRow) {
                Elem(T,i,j) = '-';
              }
              if (j < locCol) {
                Elem(T,i,j) = '-';
              }
              if (i >= locRow + m) {
```

```
              Elem(T,i,j) = '-';
            }
            if (j >= locCol + m) {
              Elem(T,i,j) = '-';
            }
            for (int k = 0; k < m; k++) {
              if (i > locRow + k && j <= locCol + k) {
                Elem(T,i,j) = '-';
              }
              if (i <= locRow + k && j > locCol + k) {
                Elem(T,i,j) = '-';
              }
            }
          }
        }
      } else {
        locCol++;
        *perbandinganHuruf += 1;

      }
    }
    locRow++;
  }

}


// Alternate Row Major Diagonal Matching
if (!found) {
  locRow = 0;
  locCol = col - 1;
  while (locRow < row) {
    locCol = col - 1;
    while (locCol >= m - 1 && !found) {
      iter = 0;
      while (iter < m && P[iter] == Elem(T, locRow + iter, locCol -
iter)) {
        iter++;
        *perbandinganHuruf += 1;
```

```
      }

      if (iter == m) {
        found = true;
        for (int i = 0; i < row; i++) {
          for (int j = col - 1; j >= 0; j--) {
            if (i < locRow) {
              Elem(T,i,j) = '-';
            }
            if (j > locCol) {
              Elem(T,i,j) = '-';
            }
            if (j <= locCol - m) {
              Elem(T,i,j) = '-';
            }
            if (i >= locRow + m) {
              Elem(T,i,j) = '-';
            }
            for (int k = 0; k < m; k++) {
              if (i > locRow + k && j == locCol - k) {
                Elem(T,i,j) = '-';
              }
              if (i == locRow + k && j < locCol - k) {
                Elem(T,i,j) = '-';
              }
            }
          }
        }
      } else {
        locCol--;
        *perbandinganHuruf += 1;

      }
    }
    locRow++;
  }
}
```

```
  // Column Major Diagonal Matching
  if (!found) {
    locRow = 0;
    locCol = 0;
    while (locCol < col) {
      locRow = 0;
      while (locRow <= row - m && !found) {
        iter = 0;
        while (iter < m && P[iter] == Elem(T, locRow + iter, locCol +
iter)) {
          iter++;
          *perbandinganHuruf += 1;
        }

        if (iter == m) {
          found = true;
           for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
              if (i < locRow) {
                Elem(T,i,j) = '-';
              }
              if (j < locCol) {
                Elem(T,i,j) = '-';
              }
              if (i >= locRow + m) {
                Elem(T,i,j) = '-';
              }
              if (j >= locCol + m) {
                Elem(T,i,j) = '-';
              }
              for (int k = 0; k < m; k++) {
                if (i > locRow + k && j <= locCol + k) {
                  Elem(T,i,j) = '-';
                }
                if (i <= locRow + k && j > locCol + k) {
                  Elem(T,i,j) = '-';
                }
```

```
            }
          }
        }
      } else {
        locRow++;
        *perbandinganHuruf += 1;
      }
    }
    locCol++;
  }
}

// Alternate Column Major Diagonal Matching
if (!found) {
  locRow = row - 1;
  locCol = 0;
  while (locCol < col) {
    locRow = row - 1;
    while (locRow >= m - 1 && !found) {
      iter = 0;
      while (iter < m && P[iter] == Elem(T, locRow - iter, locCol +
iter)) {
        iter++;
        *perbandinganHuruf += 1;
      }
      if (iter == m) {
        found = true;
        for (int i = 0; i < row; i++) {
          for (int j = col - 1; j >= 0; j--) {
            if (j < locCol) {
              Elem(T,i,j) = '-';
            }
            if (i > locRow) {
              Elem(T,i,j) = '-';
            }
            if (j >= locCol + m) {
              Elem(T,i,j) = '-';
            }
```

```
            if (i <= locRow - m) {
              Elem(T,i,j) = '-';
            }
            for (int k = 0; k < m; k++) {
              if (i < locRow - k && j <= locCol + k) {
                Elem(T,i,j) = '-';
              }
              if (i == locRow - k && j > locCol + k) {
                Elem(T,i,j) = '-';
              }
            }
          }
        }
      } else {
        locRow--;
        *perbandinganHuruf += 1;
      }
    }
    locCol++;
  }
}

// Diagonal checking from bottom right
if (!found) {
  locRow = row - 1;
  locCol = col - 1;
  while (locCol >= 0) {
    locRow = row - 1;
    while (locRow >= m - 1 && !found) {
      iter = 0;
      while (iter < m && P[iter] == Elem(T, locRow - iter, locCol -
iter)) {
        iter++;
        *perbandinganHuruf += 1;
      }

      if (iter == m) {
        found = true;
```

```
          for (int i = 0; i < row; i++) {
            for (int j = col - 1; j >= 0; j--) {
              if (j > locCol) {
                Elem(T,i,j) = '-';
              }
              if (i > locRow) {
                Elem(T,i,j) = '-';
              }
              if (j <= locCol - m) {
                Elem(T,i,j) = '-';
              }
              if (i <= locRow - m) {
                Elem(T,i,j) = '-';
              }
              for (int k = 0; k < m; k++) {
                if (i < locRow - k && j == locCol - k) {
                  Elem(T,i,j) = '-';
                }
                if (i == locRow - k && j < locCol - k) {
                  Elem(T,i,j) = '-';
                }
              }
            }
          }
        } else {
          locRow--;
          *perbandinganHuruf += 1;
        }
      }
      locCol--;
    }
  }

  if (found) {
    displayMat(T);
    cout << endl;
    cout << "----------------------ENDLINE----------------------" <<
endl << endl;
```

```
  }
}

/*** MAIN FUNCTION ***/

int main()
{

  Matrix m;
  List l;
  int perbandinganHuruf = 0;

  createList(&l);

  createMat(&m);

  string namaFile;

  cout << "Masukkan nama file (*.txt) : ";
  cin >> namaFile;

  readFile("../test/" + namaFile + ".txt", &l);
  readFileBox("../test/" + namaFile + ".txt", &m);

  l = fixList(l);

  cout << "----------------------WORD BOX----------------------" <<
endl;
  displayMat(m);
  cout << "----------------------WORD CHOICES----------------------" <<
endl;
  displayList(l);

  cout << "----------------------RESULT----------------------" << endl
<< endl;

  clock_t start = clock();
```

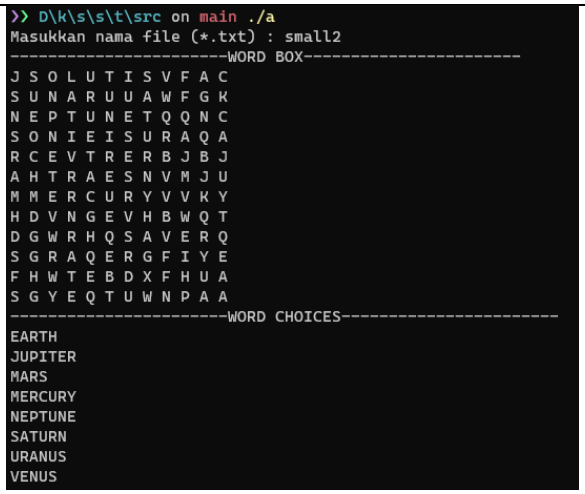
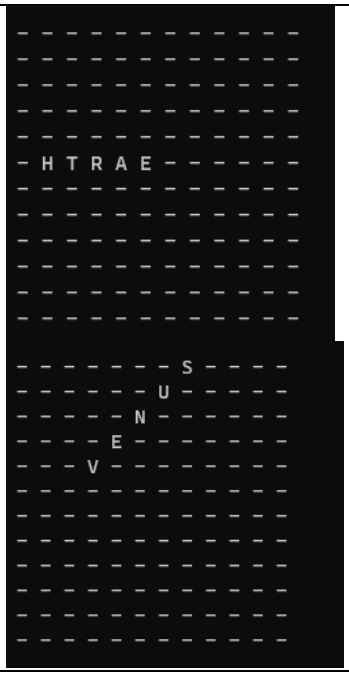
```
  for (int i = 0; i < getLength(l); i++) {
    PencocokanString(m, ListElem(l,i), &perbandinganHuruf);
  }

  cout << "-----------------------CONCLUSION-----------------------" <<
endl << endl;

  cout << "Total number of letter comparison: " << perbandinganHuruf << "
times" << endl;
  cout << "Execution Time of the program with Result Printing : ";
  cout.precision(10);
  cout << fixed << float(clock() - start) / CLOCKS_PER_SEC << "s" << endl;

  return 0;
}
```

C. Hasil Program

No	Input	Output
1		

<table>
<tr>
<td></td>
<td></td>
<td>
<pre>
J - - - - - - - - - - -
- U - - - - - - - - - -
- - P - - - - - - - - -
- - - I - - - - - - - -
- - - - T - - - - - - -
- - - - - E - - - - - -
- - - - - - R - - - - -
- - - - - - - - - - - -
- - - - - - - - - - - -
- - - - - - - - - - - -
- - - - - - - - - - - -
- - - - - - - - - - - -

----------CONCLUSION----------

Total number of letter comparison: 1972 times
Execution Time of the program with Result Printing : 0.18099999043s
</pre>
</td>
</tr>
<tr>
<td>2</td>
<td>
<pre>
Masukkan nama file (*.txt) : small1
----------WORD BOX----------
L L J K C A B L F C I
O R O L O B O C O I M
G E L A C S A P R S X
L P S T A H W V T A V
A N R L X L X Q R B I
I H P L E D O X A H S
K J Y A P H P Y N O U
F A B M A D A N Z J A
E V I S N O H T Y P L
A A Y L B M E S S A C
W E I V B A L O G O L
----------WORD CHOICES----------
ADA
FORTRAN
PHP
ALGOL
JAVA
PYTHON
ASSEMBLY
LABVIEW
SMALLTALK
BASIC
LOGO
VISUALC
COBOL
PASCAL
DELPHI
PERL
</pre>
</td>
<td>
<pre>
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - A D A - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -

- - - - - - - - F - -
- - - - - - - - O - -
- - - - - - - - R - -
- - - - - - - - T - -
- - - - - - - - R - -
- - - - - - - - A - -
- - - - - - - - N - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -

- L - - - - - - - - -
- R - - - - - - - - -
- E - - - - - - - - -
- P - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -

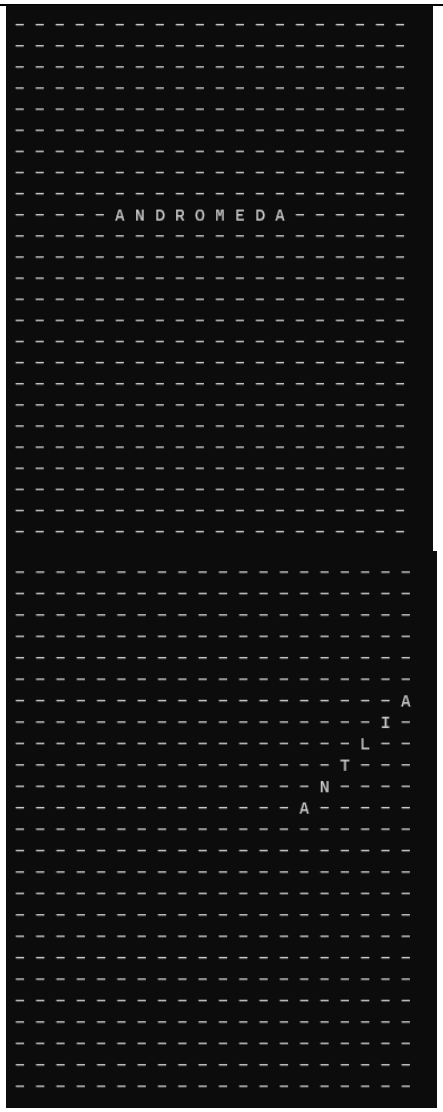
Total number of letter comparison: 2442 times
Execution Time of the program with Result Printing : 0.2930000126s
</pre>
</td>
</tr>
</table>

3	A A A A A A A A A A A D A A A A A A A A A A A L A A B A D A A D A A A R A A A A A A A A A A A O A A A A A A A A A Y A W A A A H E L L O A A A A T A A A A A A A A A M A A H A A A A G A A A O A N A G A A A O A A A R A I A A I A A O A A A N A G A A A R A D A A A I A H A S L E E P A A A N A T A A A A A A A A A G A ----------WORD CHOICES---------- BAD HELLO GOOD MORNING NIGHT DAY WORLD RIGHT SLEEP	- B A D - G - - - - - - - - - - - O - - - - - - - - - - - O - - - - - - - - - - - D - N - - - - - - - - - - - I - - - - - - - - - - - G - - - - - - - - - - - H - - - - - - - - - - - T - - - - - - - - - - -

4

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

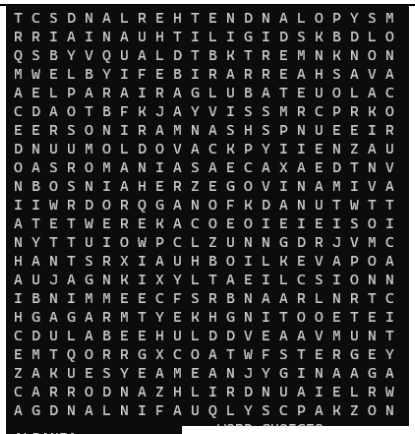
ANDROMEDA
ANTLIA
AQUARIUS
ARIES
CANCER
CAPRICORNUS
CARINA
CASSIOPEIA
CETUS
CONSTELLATIONS
DRACO
GEMINI
HERCULES
HYDRA
LEO
LEPUS
LIBRA
LYNX
LYRA
NORMA
ORION
PAVO
PEGASUS
PHOENIX
PISCES
SAGITTARIUS
SCORPIUS
SCULPTOR
SERPENS
STARS
TAURUS
VIRGO



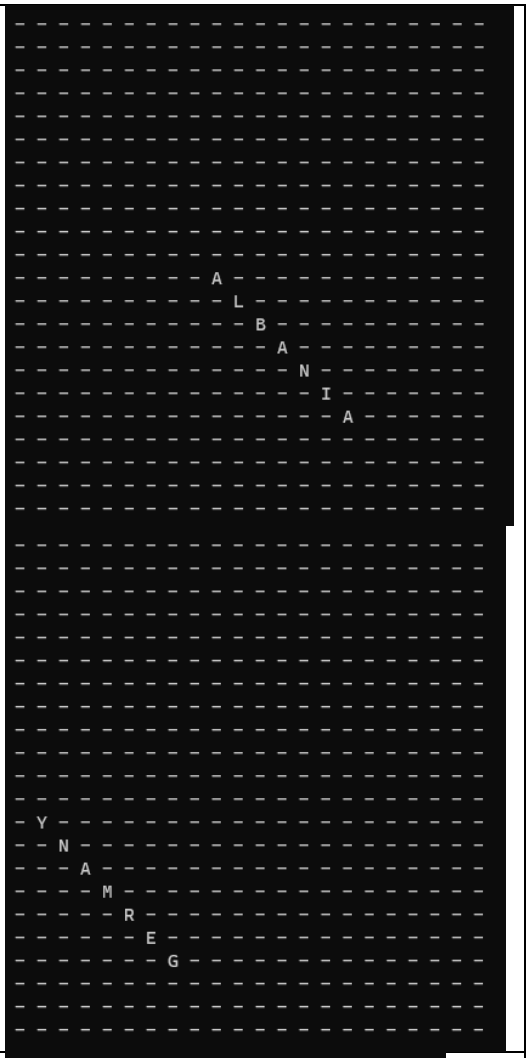
ANDROMEDA

A
I
L
T
N
A

5

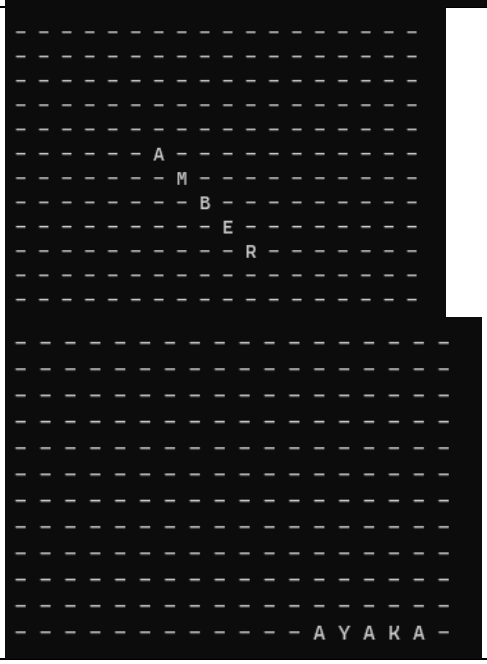


ALBANIA
GERMANY
NORWAY
ANDORRA
GREECE
POLAND
ARMENIA
HUNGARY
PORTUGAL
AUSTRIA
ICELAND
ROMANIA
AZERBAIJAN
IRELAND
RUSSIA
BELARUS
ITALY
SANMARINO
BELGIUM
KAZAKHSTAN
SERBIA
BOSNIAHERZEGOVINA
LATVIA
SLOVAKIA
BULGARIA
LIECHTENSTEIN



6

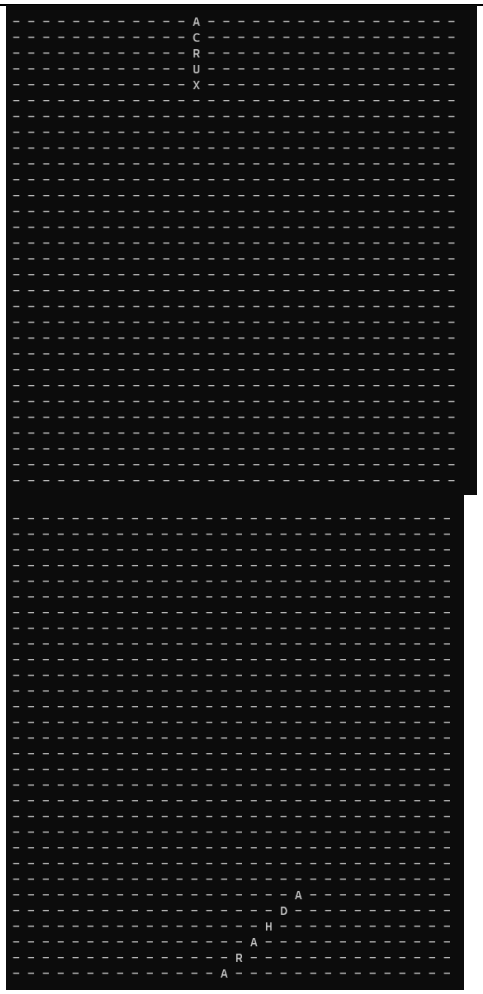
AMBER
AYAKA
BARBARA
BEIDOU
DIONA
EULA
FISCHL
GANYU
HUTAO
JEAN
KEQING
KLEE
KOKOMI
LISA
NINGGUANG
NOELLE
QIQI
BAAL
SAYU
ROSARIA
SHENHE
SUCROSE
XIANGLING
YANFEI
YOIMIYA
YUNJIN



7

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

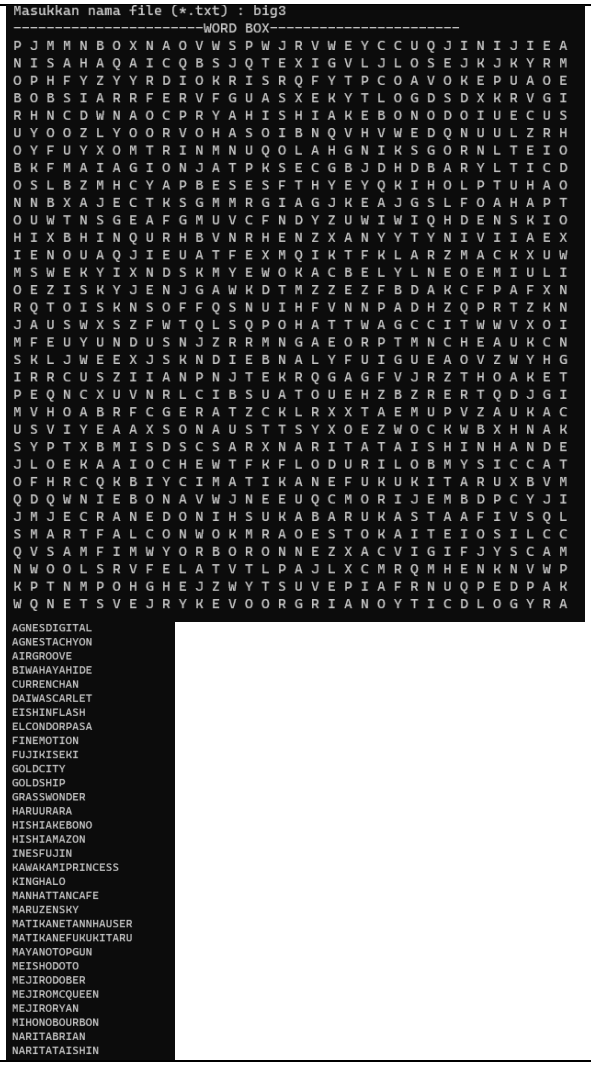
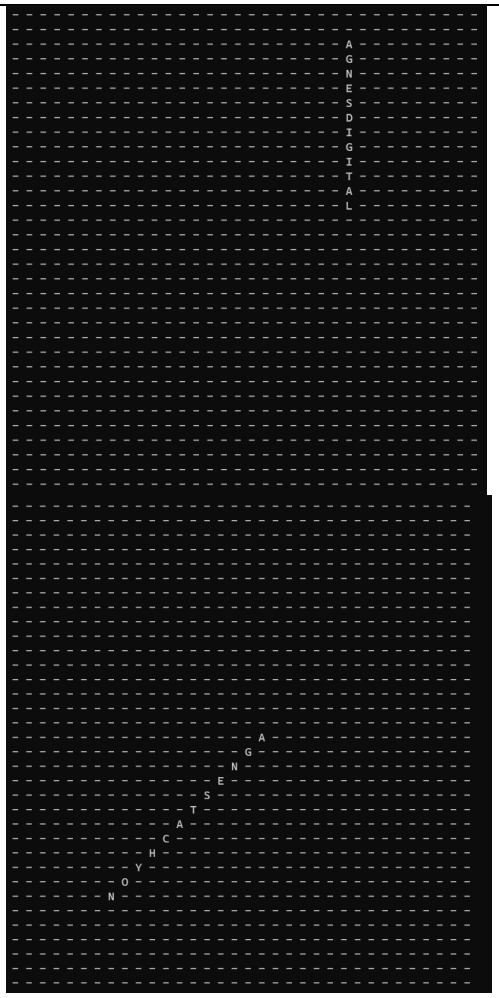
ACRUX
ADHARA
ALDEBARAN
ALNAIR
ALNILAM
ALTAIR
ANTARES
ARCTURUS
BELLATRIX
BETELGEUSE
CANOPUS
ELNATH
FOMALHAUT
HADAR
MIAPLACIDUS
MIMOSA
POLLUX
PROCYON
REGULUS
RIGEL
RIGILKENT
SHAULA
SIRIUS
SPICA
VEGA



8

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

AMBER
AYAKA
BARBARA
BEIDOU
DIONA
EULA
FISCHL
GANYU
HUTAO
JEAN
KEQING
KLEE
KOKOMI
LISA
NINGGUANG
NOELLE
QIQI
BAAL
SAYU
ROSARIA
SUCROSE
XIANGLING
YANFEI
YOIMIYA
YUNJIN
SHENHE

9		

D. Link Source Code

https://github.com/jasonk19/STIMA-Tucil1.git

E. Tabel Cek List

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.	√	