

Laporan Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II tahun 2022/2023

**Pencarian Solusi Permainan Kartu 24 dengan Algoritma Brute
Force Menggunakan Bahasa Pemrograman C++**

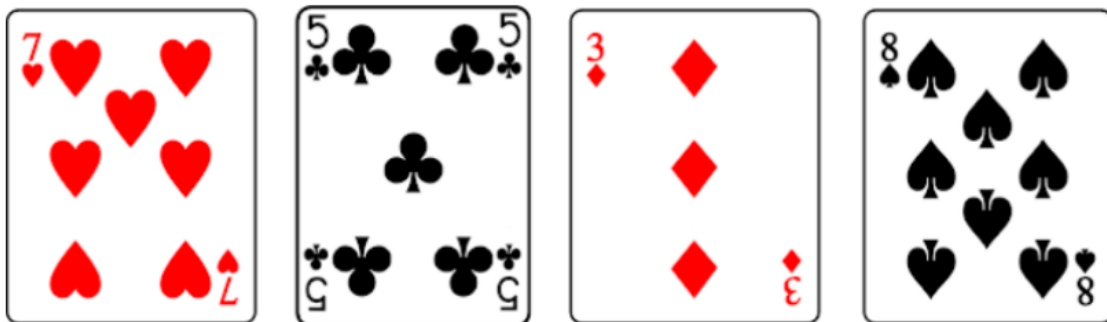


Melvin Kent Jonathan – 13521052
PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

BAB I

DESKRIPSI MASALAH

Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan hasil akhir sejumlah 24. Permainan ini menarik cukup banyak peminat dikarenakan dapat meningkatkan kemampuan berhitung serta mengasah otak agar dapat berpikir dengan cepat dan akurat. Permainan Kartu 24 biasa dimainkan dengan menggunakan kartu remi. Kartu remi terdiri dari 52 kartu yang terbagi menjadi empat suit (sekop, hati, keriting, dan wajik) yang masing-masing terdiri dari 13 kartu (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King). Yang perlu diperhatikan hanyalah nilai kartu yang didapat (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King). As bernilai 1, Jack bernilai 11, Queen bernilai 12, King bernilai 13, sedangkan kartu bilangan memiliki nilai dari bilangan itu sendiri. Pada awal permainan moderator atau salah satu pemain mengambil 4 kartu dari dek yang sudah dikocok secara random. Permainan berakhir ketika pemain berhasil menemukan solusi untuk membuat kumpulan nilainya menjadi 24. Pengubahan nilai tersebut dapat dilakukan menggunakan operasi dasar matematika penjumlahan (+), pengurangan (-), perkalian (\times), divisi (/) dan tanda kurung (). Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas. (Paragraf di atas dikutip dari sini: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah 2016/MakalahStima-2016-038.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah%202016/MakalahStima-2016-038.pdf)).



Gambar 1. Permainan Kartu 24

BAB II

ALGORITMA BRUTE FORCE DALAM PENCARIAN SOLUSI PERMAIAN KARTU 24

Algoritma *Brute Force* merupakan

Pendekatan algoritma *Brute Force* dapat dipakai dalam mencari seluruh kombinasi sebagai solusi dari permainan kartu 24. Pencarian solusi pada permainan kartu 24 pada dasarnya merupakan operasi matematika sehingga dapat diformulasikan ke dalam bentuk permutasi operator dan operan yang tersedia. Adapun agar menghasilkan seluruh kombinasi solusi, perlu kita ingat bahwa operasi matematika dalam permainan kartu 24 tidaklah harus sekuensial, melainkan angka-angka yang ada dapat dioperasikan dalam kelompoknya. Oleh karena itu, penting bagi kita untuk mengetahui kombinasi pengelompokkan angka-angka yang diperlukan untuk mencakup seluruh solusi yang ada dengan memerhatikan permutasi angka. Berikut ini adalah kombinasi pengelompokkan angka-angka yang dipakai:

1. $((A \text{ op } B) \text{ op } C) \text{ op } D$
2. $(A \text{ op } (B \text{ op } C)) \text{ op } D$
3. $A \text{ op } ((B \text{ op } C) \text{ op } D)$
4. $A \text{ op } (B \text{ op } (C \text{ op } D))$
5. $(A \text{ op } B) \text{ op } (C \text{ op } D)$

Algoritma *Brute Force* dalam pencarian solusi permainan kartu 24 yang dipakai dalam program ini dapat diformulasikan ke dalam langkah-langkah berikut:

1. Bagi operasi ke dalam bentuk 4 slot angka yang diselingi slot untuk operan (terdapat 3 slot operan)
2. Permutasikan keempat angka yang tersedia ke dalam 4 slot angka yang ada dari posisi paling kiri hingga paling kanan.
3. Apabila pada sebuah slot angka terdapat angka yang ganda (sudah pernah diiterasi sebelumnya), lewati iterasi tersebut sehingga program tidak akan menjalankan permutasi slot angka selanjutnya.
4. Permutasikan keempat operan (+, -, *, /) ke dalam 3 slot operan yang ada.
5. Untuk setiap susunan, aplikasikan 5 pengelompokkan angka dan operasi yang telah disebutkan di atas.
6. Hitung hasil operasi satu per satu. Apabila operasi menghasilkan angka 24, simpan bentuk operasi ke dalam array.
7. Setelah seluruh kemungkinan susunan yang ada dihitung, hitung banyaknya solusi dengan menghitung banyak elemen array.

BAB III

IMPLEMENTASI ALGORITMA BRUTE FORCE DENGAN C++

PRIMITIF

validator dan multiValidator

```
/* INPUT VALIDATOR */
bool validator(string input) {
    return (input == "A" || input == "2" || input == "3" || input == "4" || input == "5" || input == "6" ||
            input == "7" || input == "8" || input == "9" || input == "10" || input == "J" || input == "Q" || input == "K");
}

bool multiValidator(string input1, string input2, string input3, string input4) {
    bool valid1, valid2, valid3, valid4;

    valid1 = validator(input1);
    valid2 = validator(input2);
    valid3 = validator(input3);
    valid4 = validator(input4);

    return (valid1 && valid2 && valid3 && valid4);
}
```

Berfungsi sebagai validator dari input user untuk keempat kartu yang dipakai

convertToDouble, convertToString, convertOp

Berfungsi mengkonversi data dan tipenya dari bentuk semula ke dalam bentuk yang diinginkan

```
/* CONVERTER DOUBLE TO STRING OR STRING TO DOUBLE OR INT TO STRING*/
int convertToDouble(string input) {
    double output;
    if (input == "A") {
        output = 1;
    } else if (input == "J") {
        output = 11;
    } else if (input == "Q") {
        output = 12;
    } else if (input == "K") {
        output = 13;
    } else if (input == "10") {
        output = 10;
    } else {
        output = stod(input);
    }

    return output;
}

string convertToString(double input) {
    string output;
    if (input == 1) {
        output = "A";
    } else if (input == 11) {
        output = "J";
    } else if (input == 12) {
        output = "Q";
    } else if (input == 13) {
        output = "K";
    } else {
        output = to_string(int(input));
    }

    return output;
}
```

```
string convertOp (int x) {
    string op;
    if (x == 0) {
        op = "+";
    } else if (x == 1) {
        op = "-";
    } else if (x == 2) {
        op = "*";
    } else if (x == 3) {
        op = "/";
    }

    return op;
}
```

arrayDoubleGenerator dan *randomArrayDoubleGenerator*

```
/* GENERATE ARRAY OF DOUBLES */
void arrayDoubleGenerator(string input1, string input2, string input3, string input4, TabDouble *nums) {
    double num1, num2, num3, num4;
    num1 = convertToDouble(input1);
    num2 = convertToDouble(input2);
    num3 = convertToDouble(input3);
    num4 = convertToDouble(input4);

    append(nums, num1);
    append(nums, num2);
    append(nums, num3);
    append(nums, num4);
}

void randomArrayDoubleGenerator(TabDouble *Tab) {
    double num;
    srand(time(0));
    cout << "Berikut ini 4 angka/symbol yang dipakai:";
    for (int x = 0; x < 4; x++) {
        num = 1 + (rand() % 13);
        append(Tab, num);
        cout << " " << convertToString(num);
    }
    cout << endl;
}
```

Berfungsi menghasilkan array of Doubles berdasarkan input string yang ada atau secara random.

op

```
double op(int idx, double num1, double num2) {
    double result;

    if (idx == 0) { // addition
        result = num1 + num2;
    } else if (idx == 1) { // subtraction
        result = num1 - num2;
    } else if (idx == 2) { // multiplication
        result = num1 * num2;
    } else if (idx == 3) { // division
        result = num1 / num2;
    }

    return result;
}
```

Berfungsi menjadi operator berdasarkan masukan idx serta mengembalikan hasil dari operasi.

FUNGSI-FUNGSI UTAMA

inputHandling

```
void inputHandling(TabDouble *nums){
    // ALUR 1
    int menu;
    string input1, input2, input3, input4;

    cout << "\n===== MENU =====\n";
    cout << "1. Input simbol/angka secara manual.\n";
    cout << "2. Generate simbol/angka secara otomatis. \n";
    do {
        cout << "\nSilakan pilih menu 1 atau 2: ";
        cin >> menu;
        if (menu != 1 && menu != 2) {
            cout << "Masukan salah!\n";
        }
    } while (menu != 1 && menu != 2);

    if (menu == 1) {
        do {
            cout << "Masukkan 4 angka/huruf: \n";
            cin >> input1;
            cin >> input2;
            cin >> input3;
            cin >> input4;

            if (!multiValidator(input1, input2, input3, input4)) {
                cout << "Masukan tidak sesuai.\n";
            }
        } while (!multiValidator(input1, input2, input3, input4));

        arrayDoubleGenerator(input1, input2, input3, input4, nums);
    } else { // menu == 2
        randomArrayDoubleGenerator(nums);
    }
}
```

Berfungsi menerima input user dan memvalidasi input tersebut, lalu memanggil fungsi array generator berdasarkan masukan dari user.

processing

```
160 void processing (TabDouble nums, TabString *solutions) {
161     // PROSES
162     int i, j, k, l; // untuk permutasi angka
163     int a, b, c; // untuk permutasi operator
164     double result;
165     string symbol1, symbol2, symbol3, symbol4;
166     string op1, op2, op3;
167     string solution;
168     TabDouble container1;
169     TabDouble container2;
170     TabDouble container3;
171     TabDouble container4;
172
173     MakeEmpty(&container1);
174     for(i = 0; i < 4; i++){
175         if (!search(container1, nums.TI[i])) {
176             append(&container1, nums.TI[i]);
177             MakeEmpty(&container2);
178             for(j = 0; j < 4; j++){
179                 if (j != i && !search(container2, nums.TI[j])) {
180                     append(&container2, nums.TI[j]);
181                     MakeEmpty(&container3);
182                     for(k = 0; k < 4; k++){
183                         if (k != i && k != j && !search(container3, nums.TI[k])) {
184                             append(&container3, nums.TI[k]);
185                             MakeEmpty(&container4);
186                             for(l = 0; l < 4; l++){
187                                 if (l != i && l != j && l != k && !search(container4, nums.TI[l])) {
188                                     append(&container4, nums.TI[l]);
189                                     for (a = 0; a < 4; a++) {
190                                         for (b = 0; b < 4; b++) {
191                                             for (c = 0; c < 4; c++) {
192                                                 op1 = convertOp(a);
193                                                 op2 = convertOp(b);
194                                                 op3 = convertOp(c);
195                                                 symbol1 = convertToString(nums.TI[i]);
196                                                 symbol2 = convertToString(nums.TI[j]);
197                                                 symbol3 = convertToString(nums.TI[k]);
198                                                 symbol4 = convertToString(nums.TI[l]);
199
200                                                 // Variasi 1
201                                                 result = op(c, op(b, op(a, nums.TI[i], nums.TI[j]), nums.TI[k]), nums.TI[l]);
202
203                                                 if (result == 24) {
204                                                     solution = "(" + symbol1 + op1 + symbol2 + ")" + op2 + symbol3 + ")" + op3 + symbol4;
205                                                     appendString(solutions, solution);
206                                                 }
207
208                                                 // Variasi 2
209                                                 result = op(c, op(a, nums.TI[i], op(b, nums.TI[j], nums.TI[k])), nums.TI[l]);
210
211                                                 if (result == 24) {
212                                                     solution = "(" + symbol1 + op1 + "(" + symbol2 + op2 + symbol3 + ")" + op3 + symbol4;
213                                                     appendString(solutions, solution);
214                                                 }
215
216                                                 // Variasi 3
217                                                 result = op(a, nums.TI[i], op(c, op(b, nums.TI[j], nums.TI[k]), nums.TI[l]));
218
219                                                 if (result == 24) {
220                                                     solution = symbol1 + op1 + "(" + symbol2 + op2 + symbol3 + ")" + op3 + symbol4 + ")";
221                                                     appendString(solutions, solution);
222                                                 }
223
224                                                 // Variasi 4
225                                                 result = op(a, nums.TI[i], op(b, nums.TI[j], op(c, nums.TI[k], nums.TI[l])));
226
227                                                 if (result == 24) {
228                                                     solution = symbol1 + op1 + "(" + symbol2 + op2 + "(" + symbol3 + op3 + symbol4 + ")";
229                                                     appendString(solutions, solution);
230                                                 }
231
232                                                 // Variasi 5
233                                                 result = op(b, op(a, nums.TI[i], nums.TI[j]), op(c, nums.TI[k], nums.TI[l]));
234
235                                                 if (result == 24) {
236                                                     solution = "(" + symbol1 + op1 + symbol2 + ")" + op2 + "(" + symbol3 + op3 + symbol4 + ")";
237                                                     appendString(solutions, solution);
238                                                 }
239
240     }
241     }
242     }
243     }
244     }
245     }
246     }
247     }
248     }
249     }
250 }
```

main

```
252 int main() {
253     TabDouble nums;
254     TabString solutions;
255
256     string time;
257
258     int go = 1;
259
260     while (go == 1) {
261         MakeEmpty(&nums);
262         MakeEmpty(&solutions);
263         cout << "\n===== MENU =====\n";
264         cout << "1. GO!\n";
265         cout << "2. Exit\n";
266         do {
267             cout << "Silakan pilih menu 1 atau 2: ";
268             cin >> go;
269             if (go != 1 && go != 2) {
270                 cout << "Masukan salah!\n";
271             }
272         } while (go != 1 && go != 2);
273
274         if (go == 1) {
275             // INPUT
276             inputHandling(&nums);
277
278             // PROSES
279             auto start = high_resolution_clock::now();
280             processing(nums, &solutions);
281             auto stop = high_resolution_clock::now();
282             auto duration = duration_cast<microseconds>(stop - start); // dalam mircoseconds
283             time = to_string(double(duration.count()) / 1000); // dalam milliseconds
284
285             cout << "\nExecution Time: " << time << " milliseconds\n";
286
287             // OUTPUT
```

```
290         int idx;
291         if (solutions.Neff == 0) {
292             cout << "Tidak ada solusi.\n";
293         } else {
294             cout << solutions.Neff << " solutions found." << endl;
295             idx = 0;
296             while (idx < solutions.Neff) {
297                 cout << solutions.TI[idx] << endl;
298                 idx++;
299             }
300
301             char save;
302             do {
303                 cout << "Apakah Anda ingin menyimpan solusi? (y/n): ";
304                 cin >> save;
305                 if (save != 'y' && save != 'n') {
306                     cout << "Input tidak valid. Harap masukkan input yang valid.\n";
307                 }
308             } while (save != 'y' && save != 'n');
309
310             if (save == 'y') {
311                 ofstream fileOut;
312                 string fileName;
313
314                 cout << "Masukan nama file: ";
315                 cin >> fileName;
316                 fileOut.open(fileName);
317                 idx = 0;
318                 while (idx < solutions.Neff) {
319                     fileOut << solutions.TI[idx] << endl;
320                     idx++;
321                 }
322                 fileOut.close();
323             }
324         }
325     }
326 }
327 return 0;
328 }
```


BAB IV

CONTOH KASUS

Kasus 1

2 3 4 5

```
===== MENU =====
1. Input simbol/angka secara manual.
2. Generate simbol/angka secara otomatis.

Silakan pilih menu 1 atau 2: 1
Masukkan 4 angka/huruf:
2 3 4 5

Execution Time: 0.861000 milliseconds
40 solutions found.
```

```
Execution Time: 0.861000 milliseconds
40 solutions found.
2*((3+4)+5)
2*(3+(4+5))
2*((3+5)+4)
2*(3+(5+4))
2*((4+3)+5)
2*(4+(3+5))
2*((4+5)+3)
2*(4+(5+3))
2*((5+3)+4)
2*(5+(3+4))
2*((5+4)+3)
2*(5+(4+3))
((3-2)+5)*4
(3-(2-5))*4
((3+4)+5)*2
(3+(4+5))*2
((3+5)-2)*4
(3+(5-2))*4
((3+5)+4)*2
(3+(5+4))*2
4*((3-2)+5)
4*(3-(2-5))
((4+3)+5)*2
(4+(3+5))*2
4*((3+5)-2)
4*(3+(5-2))
4*((5-2)+3)
4*(5-(2-3))
((4+5)+3)*2
(4+(5+3))*2
4*((5+3)-2)
4*(5+(3-2))
((5-2)+3)*4
(5-(2-3))*4
((5+3)-2)*4
(5+(3-2))*4
((5+3)+4)*2
(5+(3+4))*2
((5+4)+3)*2
(5+(4+3))*2
```

Kasus 2

4 5 6 7

```
===== MENU =====
1. Input simbol/angka secara manual.
2. Generate simbol/angka secara otomatis.

Silakan pilih menu 1 atau 2: 1
Masukkan 4 angka/huruf:
4 5 6 7

Execution Time: 0.748000 milliseconds
20 solutions found.
4*((5-6)+7)
4*(5-(6-7))
4*((5+7)-6)
4*(5+(7-6))
4*((7+5)-6)
4*(7+(5-6))
4*((7-6)+5)
4*(7-(6-5))
((5-6)+7)*4
(5-(6-7))*4
((5+7)-6)*4
(5+(7-6))*4
(5+7)*(6-4)
(6-4)*(5+7)
(6-4)*(7+5)
((7+5)-6)*4
(7+(5-6))*4
(7+5)*(6-4)
((7-6)+5)*4
(7-(6-5))*4
```

Kasus 3

6 6 6 6

```
===== MENU =====
1. Input simbol/angka secara manual.
2. Generate simbol/angka secara otomatis.

Silakan pilih menu 1 atau 2: 1
Masukkan 4 angka/huruf:
6 6 6 6

Execution Time: 0.207000 milliseconds
7 solutions found.
((6+6)+6)+6
(6+(6+6))+6
6+((6+6)+6)
6+(6+(6+6))
(6+6)+(6+6)
(6*6)-(6+6)
((6*6)-6)-6
```

Kasus 4

3 A J K

```
===== MENU =====
1. Input simbol/angka secara manual.
2. Generate simbol/angka secara otomatis.

Silakan pilih menu 1 atau 2: 1
Masukkan 4 angka/huruf:
3 A J K

Execution Time: 0.827000 milliseconds
Tidak ada solusi.
```

Kasus 5

Random (J 5 5 5)

```
===== MENU =====
1. Input simbol/angka secara manual.
2. Generate simbol/angka secara otomatis.

Silakan pilih menu 1 atau 2: 2
Berikut ini 4 angka/simbol yang dipakai: J 5 5 5

Execution Time: 0.227000 milliseconds
Tidak ada solusi.
```

Kasus 6

Random (8 K Q 3)

```
===== MENU =====
1. Input simbol/angka secara manual.
2. Generate simbol/angka secara otomatis.

Silakan pilih menu 1 atau 2: 2
Berikut ini 4 angka/simbol yang dipakai: 8 K Q 3

Execution Time: 0.547000 milliseconds
44 solutions found.
```

44 solutions found.

$$(8*(K-Q))^3$$

$$8*((K-Q)^3)$$

$$(8/(K-Q))^3$$

$$8/((K-Q)/3)$$

$$8*(3*(K-Q))$$

$$(8^3)*(K-Q)$$

$$8*(3/(K-Q))$$

$$(8^3)/(K-Q)$$

$$(K-(8+3))*Q$$

$$((K-8)-3)*Q$$

$$((K-Q)*8)^3$$

$$(K-Q)*(8^3)$$

$$((K-Q)^3)*8$$

$$(K-Q)*(3^8)$$

$$((K+3)/8)*Q$$

$$(K+3)/(8/Q)$$

$$(K-(3+8))*Q$$

$$((K-3)-8)*Q$$

$$((K+3)*Q)/8$$

$$(K+3)*(Q/8)$$

$$(Q/8)*(K+3)$$

$$Q/(8/(K+3))$$

$$(Q/8)*(3+K)$$

$$Q/(8/(3+K))$$

$$Q*(K-(8+3))$$

$$Q*((K-8)-3)$$

$$(Q*(K+3))/8$$

$$Q*((K+3)/8)$$

$$Q*(K-(3+8))$$

$$Q*((K-3)-8)$$

$$(Q*(3+K))/8$$

$$Q*((3+K)/8)$$

$$3*(8*(K-Q))$$

$$(3^8)*(K-Q)$$

$$3*(8/(K-Q))$$

$$(3^8)/(K-Q)$$

$$((3+K)/8)*Q$$

$$(3+K)/(8/Q)$$

$$((3+K)*Q)/8$$

$$(3+K)*(Q/8)$$

$$(3*(K-Q))*8$$

$$3*((K-Q)*8)$$

$$(3/(K-Q))*8$$

$$3/((K-Q)/8)$$

Kasus 7

Random (A K J 2)

```
===== MENU =====
1. Input simbol/angka secara manual.
2. Generate simbol/angka secara otomatis.

Silakan pilih menu 1 atau 2: 2
Berikut ini 4 angka/simbol yang dipakai: A K J 2

Execution Time: 0.891000 milliseconds
24 solutions found.
K-((A-2)*J)
K-(J*(A-2))
K-(J/(A-2))
K+(J*(2-A))
(K+J)*(2-A)
K+(J/(2-A))
(K+J)/(2-A)
K+((2-A)*J)
(K*(2-A))+J
(K/(2-A))+J
J-((A-2)*K)
J-(K*(A-2))
J-(K/(A-2))
J+(K*(2-A))
(J+K)*(2-A)
J+(K/(2-A))
(J+K)/(2-A)
J+((2-A)*K)
(J*(2-A))+K
(J/(2-A))+K
((2-A)*K)+J
(2-A)*(K+J)
((2-A)*J)+K
(2-A)*(J+K)
Apakah Anda ingin menyimpan solusi? (y/n):
```

Kasus 8

Random (2 4 K 10)

===== MENU =====

1. Input simbol/angka secara manual.
2. Generate simbol/angka secara otomatis.

Silakan pilih menu 1 atau 2: 2

Berikut ini 4 angka/simbol yang dipakai: 2 4 K 10

Execution Time: 0.761000 milliseconds

16 solutions found.

$(2*(4+K))-10$

$2*(4*(K-10))$

$(2*4)*(K-10)$

$(2*(K+4))-10$

$(2*(K-10))*4$

$2*((K-10)*4)$

$4*(2*(K-10))$

$(4*2)*(K-10)$

$((4+K)*2)-10$

$(4*(K-10))*2$

$4*((K-10)*2)$

$((K+4)*2)-10$

$((K-10)*2)*4$

$(K-10)*(2*4)$

$((K-10)*4)*2$

$(K-10)*(4*2)$

Apakah Anda ingin menyimpan solusi? (y/n):

LAMPIRAN

Repository GitHub

https://github.com/melvinkj/Tucil1_13521052

Checkpoint Pengerjaan

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil running	✓	
3. Program dapat membaca input / generate sendiri dan memberikan luaran	✓	
4. Solusi yang diberikan program memenuhi (berhasil mencapai 24)	✓	
5. Program dapat menyimpan solusi dalam file teks	✓	