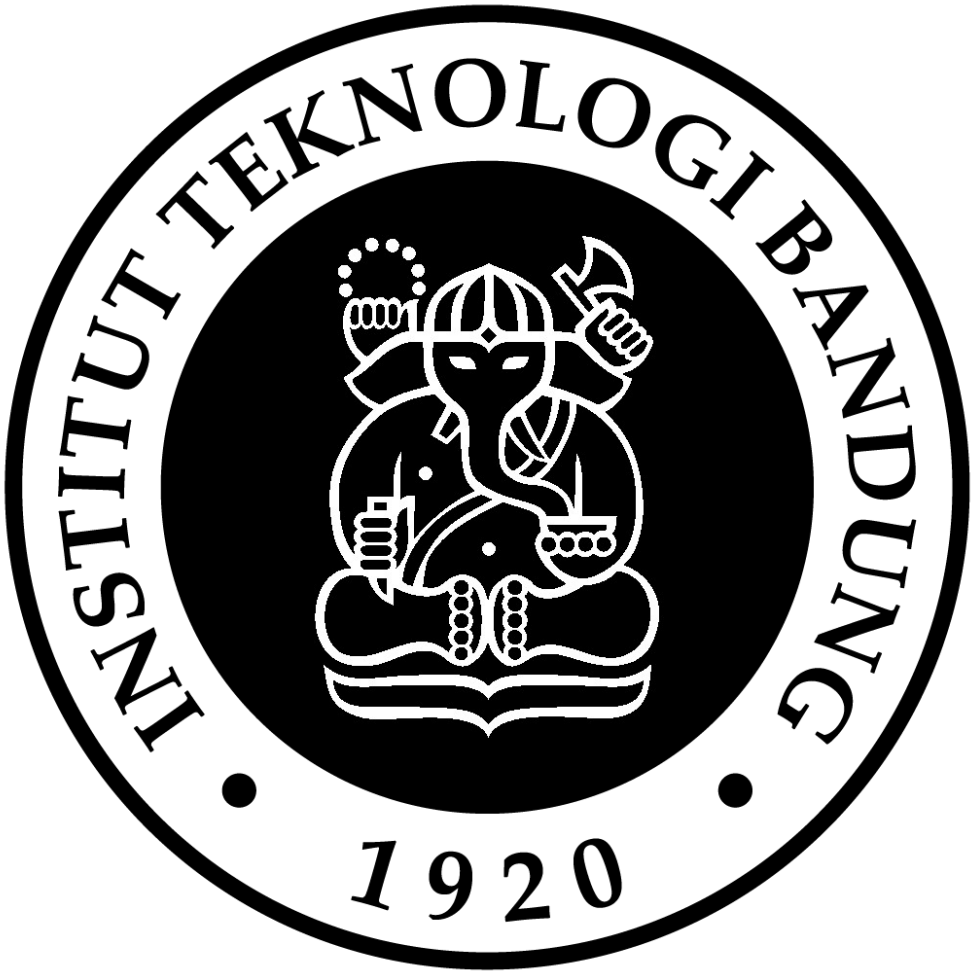
LAPORAN TUGAS KECIL
IF2211 Strategi Algoritma

Penyelesaian Permainan Kartu 24 dengan Algoritma *Brute Force*



Dibuat oleh:

Moch. Sofyan Firdaus (13521083)

1. Deskripsi Persoalan

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 (×), divisi (/) dan tanda kurung (()). Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas. (Dikutip dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah2016/MakalahStima-2016-038.pdf).

2. Langkah-Langkah Penyelesaian

Permainan kartu 24 dapat diselesaikan dengan algoritma *brute force.* Algoritma penyelesaian tersebut adalah sebagai berikut.

1. Petakan angka-angka 1-4 dengan operasi dasar matematika yang akan digunakan, misalnya 1, 2, 3, dan 4 berturut-turut untuk operasi penjumlahan, pengurangan, pembagian, dan perkalian. Dalam bahasa pemrograman C, hal ini dapat dilakukan dengan membuat sebuah **enum.**
2. Buat sebuah fungsi untuk mengevaluasi operasi dasar matematika tersebut, misalnya fungsi eval yang menerima tiga buah parameter: operator dalam bentuk angka yang sudah dipetakan atau enum, operan 1, dan operan 2.
3. Dapatkan semua permutasi berbeda yang dapat dibentuk oleh susunan bilangan-bilangan masukan, lalu tampung semuanya ke dalam sebuah larik (*array*).
4. Untuk setiap permutasi di atas dan untuk setiap kombinasi operasi yang dapat dibentuk (dilakukan dengan pengulangan/*looping* bersarang), lakukan evaluasi matematika terhadap setiap kemungkinan pengelompokan operasi (tanda kurung). Semua pengelempokan operasi yang mungkin adalah
    1. ((a **op** b) **op** c) **op** d
    2. (a **op** (b **op** c)) **op** d
    3. (a **op** b) **op** (c **op** d)
    4. a **op** ((b **op** c) **op** d)
    5. a **op** (b **op** (c **op** d))

dengan a, b, c, dan d merupakan operan dan op adalah operator.

5. Untuk setiap kemungkinan di atas, cek apakah hasil evaluasinya adalah 24. Jika hasilnya adalah 24, cetak ekspresi matematika yang bersesuaian atau tampung ke dalam *array of string*.

3. Kode Sumber

Kode sumber berikut ditulis dengan bahasa pemrograman C. Semua kode sumber dapat dilihat di repositori github berikut: https://github.com/msfir/card24-solver

```c
// solver.h

#ifndef SOLVER_H_
#define SOLVER_H_

typedef enum { ADD, SUB, DIV, MUL } Operator;

char operatorSymbol(Operator op);

float eval(Operator op, float left, float right);

bool arrays_equal(const int* a, const int* b, const int size);

int str_length(const char* s);

void copy_array(const int* source, int* destination, int size);

bool is_element4(int e[4], int arr[24][4], int size);

int unique_permutations4(int arr[4], int buffer[][4]);

int make24(int a, int b, int c, int d, char solutions[][22]);

#endif
```

```c
// solve.c

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "solver.h"

char operatorSymbol(Operator op) {
  char sym = '?';
  if (op == ADD) {
    sym = '+';
  } else if (op == SUB) {
    sym = '-';
  } else if (op == MUL) {
    sym = '*';
  } else if (op == DIV) {
    sym = '/';
  }
  return sym;
}
```

```
float eval(Operator op, float left, float right) {
  switch (op) {
    case ADD: return left + right;
    case SUB: return left - right;
    case MUL: return left * right;
    case DIV: return left / right;
    default: return 0;
  }
}

bool arrays_equal(const int* a, const int* b, const int size) {
  int i = 0;
  while (i < size) {
    if (a[i] != b[i]) {
      return false;
    }
    i++;
  }
  return true;
}

int str_length(const char* s) {
  int i = 0;
  while (s[i++] != '\0');
  return i - 1;
}

void copy_array(const int* source, int* destination, int size) {
  for (int i = 0; i < size; i++) {
    destination[i] = source[i];
  }
}

bool is_element4(int e[4], int arr[24][4], int size) {
  int i = 0;
  while (i < size) {
    if (arrays_equal(e, arr[i], 4)) {
      return true;
    }
    i++;
  }
  return false;
}

int unique_permutations4(int arr[4], int buffer[][4]) {
  int len = 0;
  for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
      for (int k = 0; k < 4; k++) {
        for (int l = 0; l < 4; l++) {
          int temp[4];
          temp[0] = arr[i];
          if (i != j) {
            temp[1] = arr[j];
            if (i != k && j != k) {
              temp[2] = arr[k];
              if (i != l && j != l && k != l) {
                temp[3] = arr[l];
                if (len == 0 || !is_element4(temp, buffer, len)) {
```

```
                    copy_array(temp, buffer[len++], 4);
                }
              }
            }
          }
        }
      }
    }
  }
  return len;
}

int make24(int a, int b, int c, int d, char solutions[][22]) {
  int n = 0;
  int buffer[24][4];
  int arr[4] = {a, b, c, d};
  int p = unique_permutations4(arr, buffer);

  for (int i = 0; i < p; i++) {
    a = buffer[i][0];
    b = buffer[i][1];
    c = buffer[i][2];
    d = buffer[i][3];
    for (Operator op1 = ADD; op1 <= MUL; op1++) {
      char opc1 = operatorSymbol(op1);
      for (Operator op2 = ADD; op2 <= MUL; op2++) {
        char opc2 = operatorSymbol(op2);
        for (Operator op3 = ADD; op3 <= MUL; op3++) {
          char opc3 = operatorSymbol(op3);
          // hardcode 5 kemungkinan urutan evaluasi
          // case 1: ((a + b) + c) + d
          if (eval(op3, eval(op2, eval(op1, a, b), c), d) == 24) {
            sprintf(solutions[n++], "((%d %c %d) %c %d) %c %d", a, opc1, b,
opc2, c, opc3, d);
          }
          // case 2: (a + (b + c)) + d
          if (eval(op3, eval(op1, a, eval(op2, b, c)), d) == 24) {
            sprintf(solutions[n++], "(%d %c (%d %c %d)) %c %d", a, opc1, b,
opc2, c, opc3, d);
          }
          // case 3: (a + b) + (c + d)
          if (eval(op2, eval(op1, a, b), eval(op3, c, d)) == 24) {
            sprintf(solutions[n++], "(%d %c %d) %c (%d %c %d)", a, opc1, b,
opc2, c, opc3, d);
          }
          // case 4: a + ((b + c) + d)
          if (eval(op1, a, eval(op3, eval(op2, b, c), d)) == 24) {
            sprintf(solutions[n++], "%d %c ((%d %c %d) %c %d)", a, opc1, b,
opc2, c, opc3, d);
          }
          // case 5: a + (b + (c + d))
          if (eval(op1, a, eval(op2, b, eval(op3, c, d))) == 24) {
            sprintf(solutions[n++], "%d %c (%d %c (%d %c %d))", a, opc1, b,
opc2, c, opc3, d);
          }
        }
      }
    }
  }
```

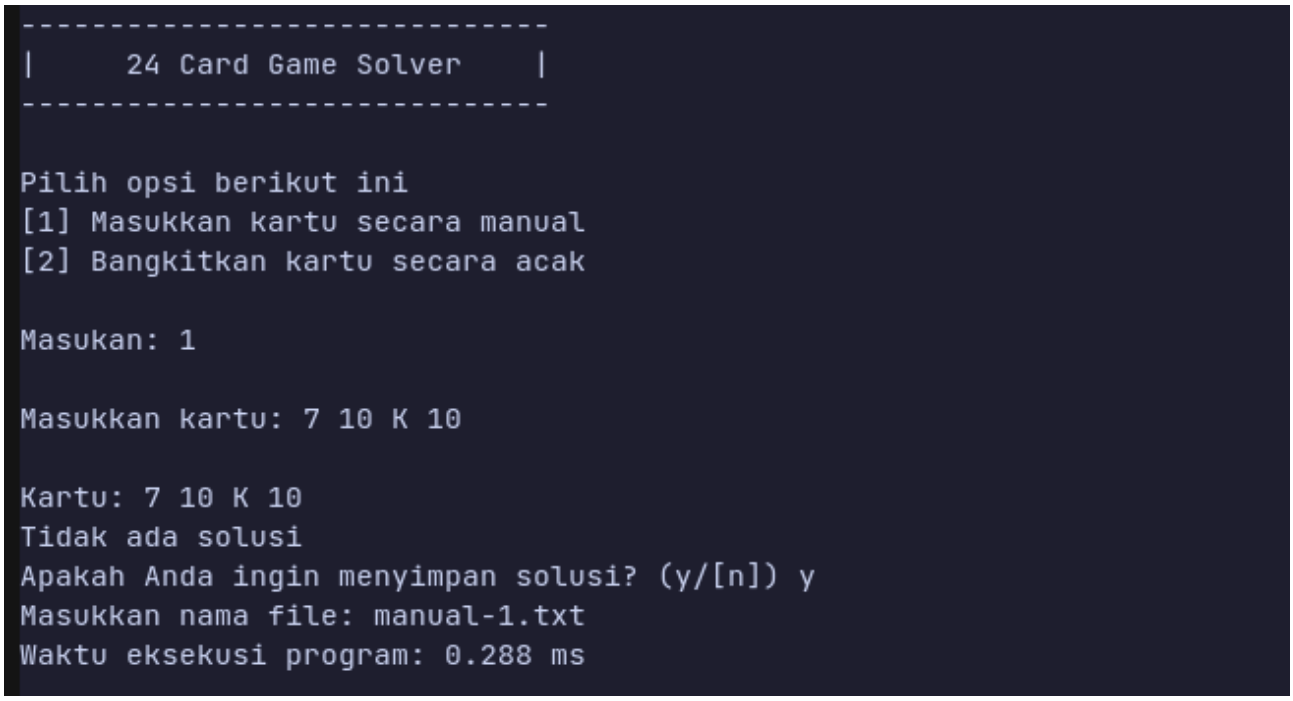
```
  return n;
}
```

```
// main.c

#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <errno.h>
#include "solver.h"

#define MAX_INPUT_BUFFER 20
#define CARDS_LENGTH 13
#define MAX_FILE_NAME_LENGTH 100

static const const char* CARDS[CARDS_LENGTH] = { "A", "2", "3", "4", "5", "6",
"7", "8", "9", "10", "J", "Q", "K" };

bool parse_input(const char* input, int* output) {
  int val;
  int n = sscanf(input, "%d", &val);
  if (n != 1) {
    char c;
    sscanf(input, "%c", &c);
    switch (c) {
      case 'A':
        *output = 1;
        return true;
      case 'J':
        *output = 11;
        return true;
      case 'Q':
        *output = 12;
        return true;
      case 'K':
        *output = 13;
        return true;
      default:
        return false;
    }
  }
  if (2 <= val && val <= 10) {
    *output = val;
    return true;
  }
  return false;
}

void save_solutions(const char* file_name, const char cards[4]
[MAX_INPUT_BUFFER], const char solutions[][22], const int num_of_solutions) {
  FILE* file;
  file = fopen(file_name, "w");
  if (errno) {
    perror("Gagal menyimpan solusi");
    exit(1);
  }
```

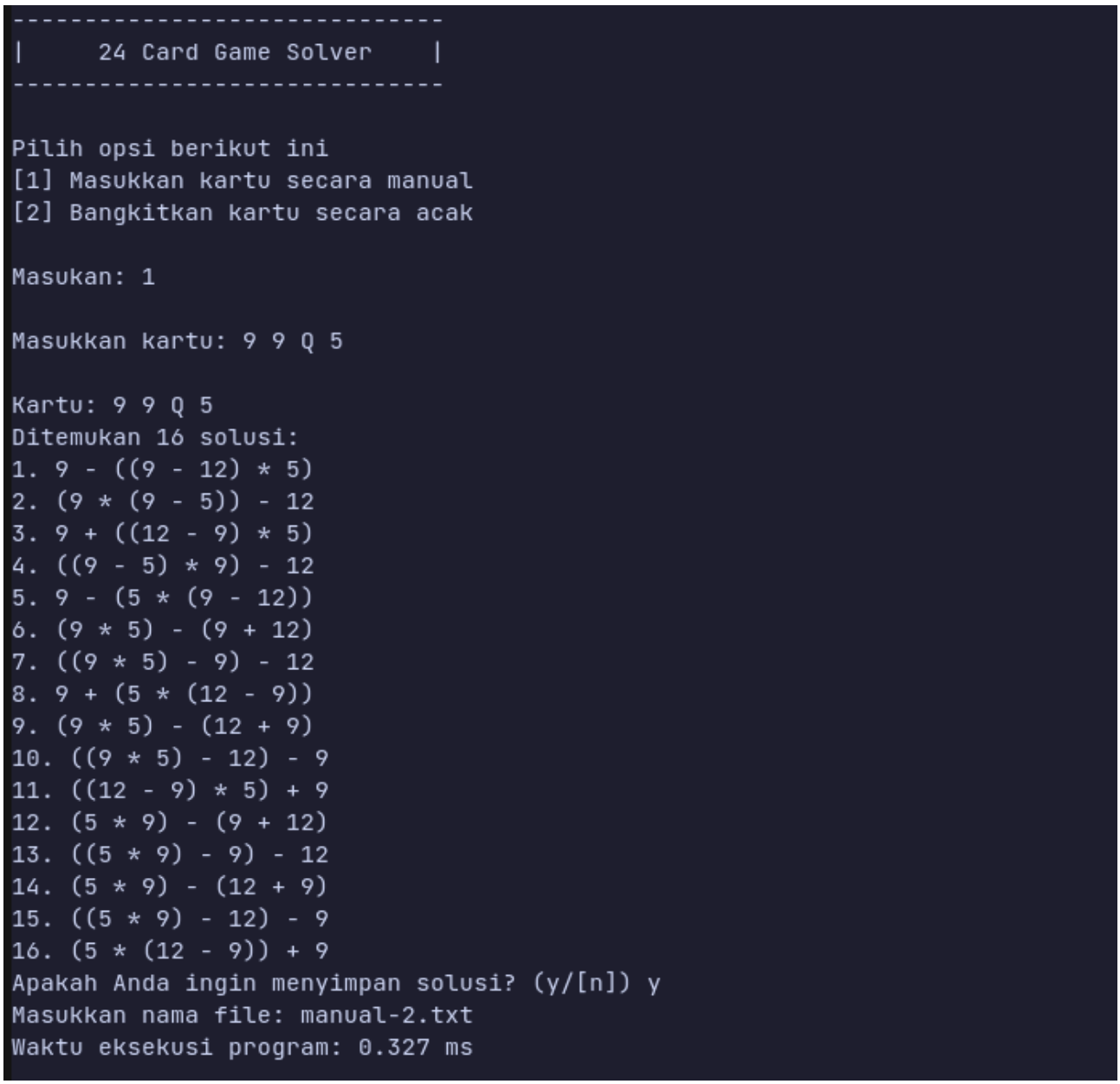
```
  fprintf(file, "Kartu: %s %s %s %s\n", cards[0], cards[1], cards[2],
cards[3]);
  if (num_of_solutions > 0) {
    fputs("Solusi:", file);
    for (int i = 0; i < num_of_solutions; i++) {
      fprintf(file, "\n%d. %s", i + 1, solutions[i]);
    }
  } else {
    fputs("Tidak ada solusi", file);
  }
  fclose(file);
}

void generate_cards(char cards[4][MAX_INPUT_BUFFER], int cards_int[4]) {
  time_t t;
  srand(time(&t));
  for (int i = 0; i < 4; i++) {
    int idx = rand() % CARDS_LENGTH;
    sprintf(cards[i], "%s", CARDS[idx]);
    cards_int[i] = idx + 1;
  }
}

void clear_stdin() {
  int c;
  while ((c = getchar()) != '\n' && c != EOF);
}

void get_user_input(char cards[4][MAX_INPUT_BUFFER], int cards_int[4]) {
  bool valid;
  do {
    valid = true;
    printf("Masukkan kartu: ");
    scanf("%s %s %s %s", cards[0], cards[1], cards[2], cards[3]);
    for (int i = 0; i < 4 && valid; i++) {
      if (!parse_input(cards[i], &cards_int[i])) {
        valid = false;
      }
    }
    if (!valid) {
      puts("Masukan tidak valid");
    }
    clear_stdin();
  } while (!valid);
}

int main_menu() {
  int choice;
  bool valid;
  do {
    puts("");
    puts("Pilih opsi berikut ini");
    puts("[1] Masukkan kartu secara manual");
    puts("[2] Bangkitkan kartu secara acak");
    puts("");
    printf("Masukan: ");
    scanf("%d", &choice);
    valid = choice == 1 || choice == 2;
    if (!valid) {
```

```
      puts("Masukan tidak sesuai");
    }
  } while (!valid);
  clear_stdin();
  return choice;
}

int main() {
  char cards[4][MAX_INPUT_BUFFER];
  int cards_int[4];
  char solutions[500][22];
  puts("------------------------------");
  puts("|     24 Card Game Solver    |");
  puts("------------------------------");
  int choice = main_menu();
  puts("");
  if (choice == 1) {
    get_user_input(cards, cards_int);
    puts("");
  } else {
    generate_cards(cards, cards_int);
  }
  printf("Kartu: %s %s %s %s\n", cards[0], cards[1], cards[2], cards[3]);
  clock_t elapsed_time = clock();
  int n = make24(cards_int[0], cards_int[1], cards_int[2], cards_int[3],
solutions);
  elapsed_time = clock() - elapsed_time;
  if (n > 0) {
    printf("Ditemukan %d solusi:\n", n);
    for (int i = 0; i < n; i++) {
      printf("%d. %s\n", i + 1, solutions[i]);
    }
  } else {
    puts("Tidak ada solusi");
  }
  char ans[MAX_INPUT_BUFFER];
  do {
    printf("Apakah Anda ingin menyimpan solusi? (y/[n]) ");
    fgets(ans, MAX_INPUT_BUFFER, stdin);
  } while (ans[0] != 'y' && ans[0] != 'n' && ans[0] != '\n');
  if (ans[0] == 'y') {
    char file_name[MAX_FILE_NAME_LENGTH];
    printf("Masukkan nama file: ");
    fgets(file_name, MAX_FILE_NAME_LENGTH, stdin);
    int len = str_length(file_name);
    if (len > 0 && file_name[len - 1] == '\n') {
      file_name[--len] = '\0';
    }
    save_solutions(file_name, cards, solutions, n);
  }
  printf("Waktu eksekusi program: %.3lf ms\n", (double) elapsed_time /
CLOCKS_PER_SEC * 1000);
}
```

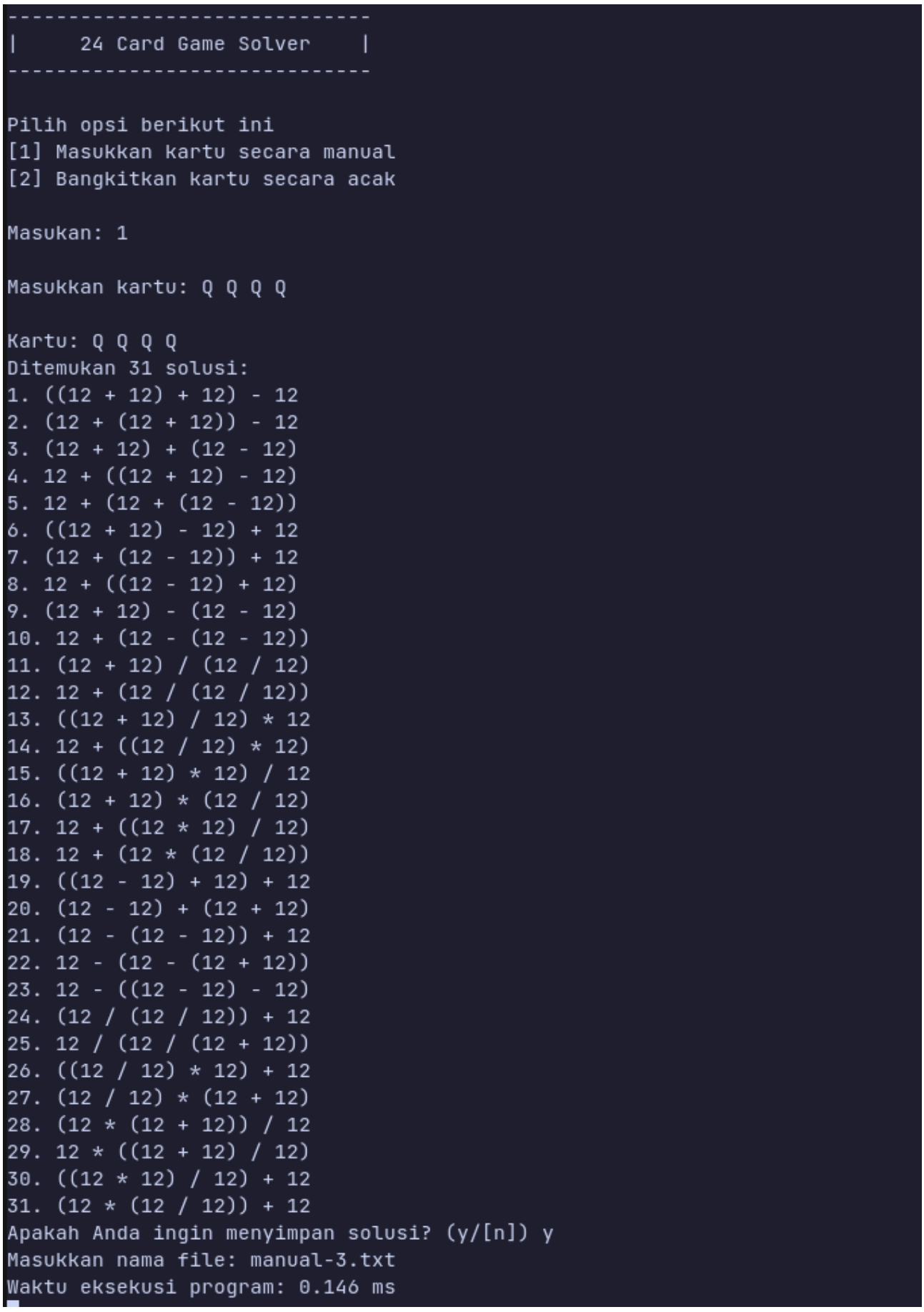
4. Contoh Eksekusi Program



Gambar 1. Contoh input manual 7 10 K 10



Gambar 2. Contoh input manual 9 9 Q 5



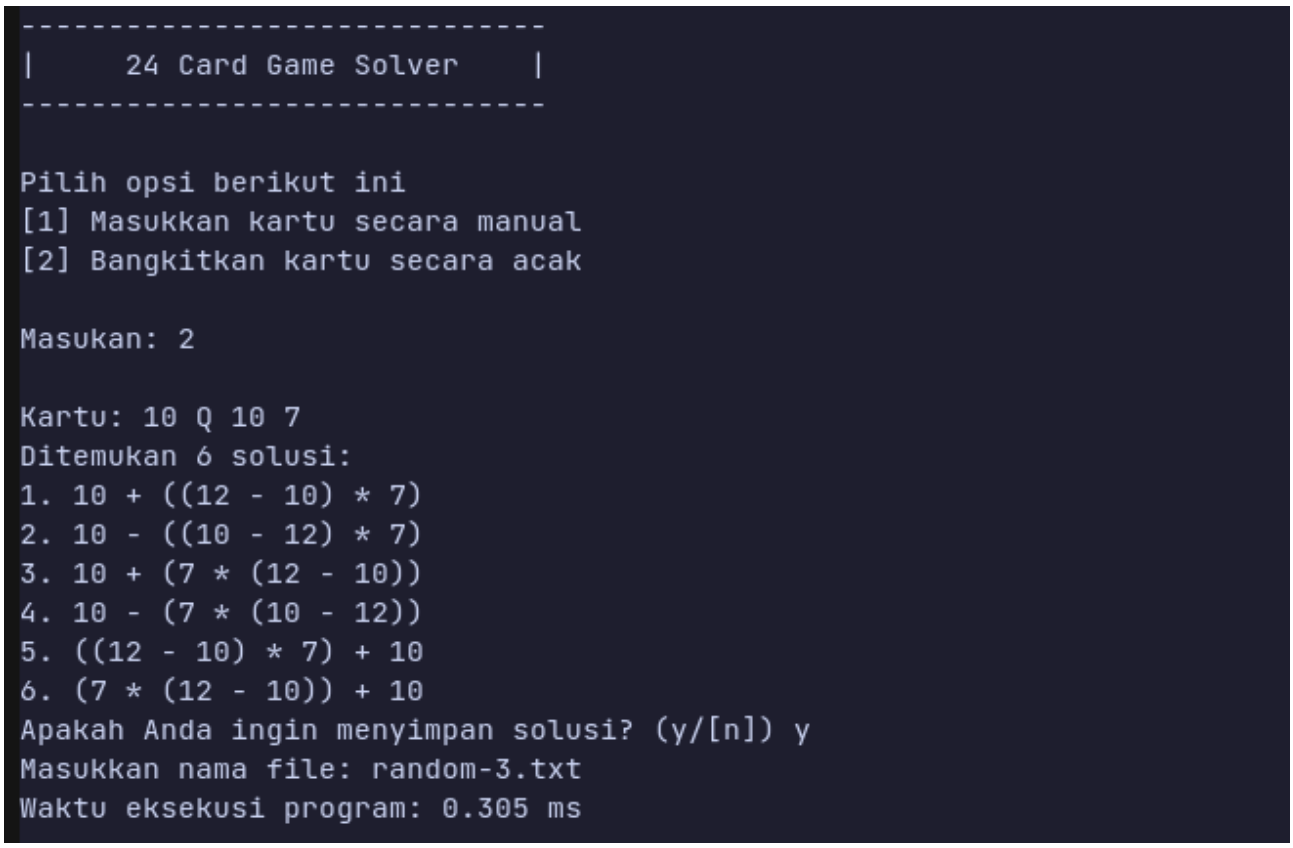
Gambar 3. Contoh input manual Q Q Q Q

```
------------------------------
|     24 Card Game Solver    |
------------------------------

Pilih opsi berikut ini
[1] Masukkan kartu secara manual
[2] Bangkitkan kartu secara acak

Masukan: 2

Kartu: 3 5 5 3
Ditemukan 1 solusi:
1. (5 * 5) - (3 / 3)
Apakah Anda ingin menyimpan solusi? (y/[n]) y
Masukkan nama file: random-1.txt
Waktu eksekusi program: 0.201 ms
```

Gambar 4. Contoh acak 3 5 5 3

```
------------------------------
|     24 Card Game Solver    |
------------------------------

Pilih opsi berikut ini
[1] Masukkan kartu secara manual
[2] Bangkitkan kartu secara acak

Masukan: 2

Kartu: 9 K Q 9
Ditemukan 10 solusi:
1. (9 * (13 - 9)) - 12
2. (13 - (9 / 9)) + 12
3. 13 - ((9 / 9) - 12)
4. ((13 - 9) * 9) - 12
5. (13 + 12) - (9 / 9)
6. 13 + (12 - (9 / 9))
7. (12 - (9 / 9)) + 13
8. 12 - ((9 / 9) - 13)
9. (12 + 13) - (9 / 9)
10. 12 + (13 - (9 / 9))
Apakah Anda ingin menyimpan solusi? (y/[n]) y
Masukkan nama file: random-2.txt
Waktu eksekusi program: 0.367 ms
```

Gambar 5. Contoh acak 9 K Q 9



Gambar 6. Contoh acak 10 Q 10 7

5. Lampiran

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	✔	

Link repositori: https://github.com/msfir/card24-solver