LAPORAN TUGAS KECIL
IF2211 STRATEGI ALGORITMA

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Disusun oleh

Noel Christoffel Simbolon 13521096

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKOLOGI BANDUNG

2023

DAFTAR ISI

DAFTAR ISI...i

BAB I IMPLEMENTASI ALGORITMA BRUTE FORCE...1

1.1 Deskripsi Algoritma Brute Force 1

BAB II SOURCE CODE 3

2.1 `Game.java`...3

2.2 `Play.java`...28

BAB III EKSPERIMEN..31

3.1 Eksperimen...31

LAMPIRAN...36

Tautan *Remote Repository* 36

Checklist..36

BAB I
IMPLEMENTASI ALGORITMA BRUTE FORCE

1.1 Deskripsi Algoritma Brute Force

Dalam program ini, algoritma *brute force* diimplementasikan pada *method* `findsolutions()` yang terdapat pada *class* `Game` di *file* `Game.java`. Pada *method* ini, pertama-tama ditampilkan nila-nilai numerik yang merupakan representasi kartu input dari *user*. Nilai-nilai numerik ini berupa bilangan bulat yang akan dioperasikan untuk mencari semua solusi yang mungkin. Selanjutnya, dibuat `ArrayList` baru yang kosong untuk menyimpan semua solusi yang ditemukan dalam bentuk `String`. Lalu, disimpan nilai dari nilai dari *high-resolution time source* dari Java Virtual Machine (JVM) yang sedang berjalan. Nilai ini disimpan pertama kali sesaat sebelum algoritma *brute force* dieksekusi. Nilai ini disimpan untuk menghitung waktu eksekusi dari algoritma *brute force* yang diimplementasikan.

Selanjutnya, *method* mengeksekusi algoritma *brute force* dengan cara membuat empat *nested for loops* yang digunakan untuk mencari semua permutasi yang mungkin dari empat bilangan bulat yang merupakan nilai numerik representasi kartu-kartu input. Selanjutnya, di dalam *for loop* terdalam, setiap bilangan bulat tersebut disimpan nilainya dalam sebuah variabel untuk memudahkan pembuatan `String` dari solusi. Selain itu, setiap bilangan bulat tersebut juga disimpan nilainya dalam sebuah variabel, tetapi kali ini tipe variabelnya adalah `double`. Variabel-variabel dengan tipe `double` ini, dibuat untuk memudahkan proses evaluasi ekspresi matematika, khususnya operasi *real division* yang dilakukan oleh operator `/`.

Setelah itu, terdapat banya *if block* untuk mengecek setiap permutasi *operator* dan permutasi tanda kurung yang signifikan. Terdapat 64 permutasi *operator* yang mungkin, yaitu sebagai berikut.

+++	+*+	-++	-*+	*++	**+	/++	/*+
++-	+*-	-+-	-*-	*+-	**-	/+-	/*-
++*	+**	-+*	-**	*+*	***	/+*	/**
++/	+*/	-+/	-*/	*+/	**/	/+/	/*/
+-+	+/+	--+	-/+	*-+	*/+	/-+	//+
+--	+/-	---	-/-	*--	*/-	/--	//-
+-*	+/*	--*	-/*	*-*	*/*	/-*	//*
+-/	+//	--/	-//	*-/	*/ /	/-/	///

Untuk tanda kurung, hanya ada 5 permutasi signifikan yang dicek dalam *if block*. Berikut adalah permutasi-permutasi tersebut, dengan `a`, `b`, `c`, dan `d` melambangkan bilangan yang diperasikan, serta `op` melambangkan *operator*.

((a op b) op c) op d
(a op (b op c)) op d
a op ((b op c) op d)
a op (b op (c op d))
(a op b) op (c op d)

Perhatikan bahwa operator / adalah *real division* bukan *integer division*.

Alasan mengapa semua permutasi yang mungkin dengan tanpa tanda kurung dan sepasang tanda kurung tidak dicek adalah karena makna dari ekspresi matematikanya akan mirip dengan solusi lainnya. Sebagai contoh, diberikan empat bilangan bulat 4, 2, 1, dan 4, ekspresi-ekspresi berikut memiliki makna yang mirip.

```
(4 + 2) * 1 * 4
(1 * 4) * (2 + 4)
```

Contoh lain dari dua ekspresi dengan makna yang mirip adalah sebagai berikut.

```
11 * 2 * 1 + 2
((11 * 2) * 1) + 2
```

Jadi, semua permutasi yang mungkin dengan tanpa tanda kurung dan sepasang tanda kurung tidak dikalkulasi untuk meminimalkan jumlah ekspresi dengan makna yang sama.

Dalam setiap *if block*, sebuah ekspresi matematika dicek apakah hasilnya sama dengan 24. Jika iya, sebuah `String` yang berisi ekspresi matematika tersebut akan ditambahkan ke dalam `ArrayList` di awalan *method* tadi. Setelah semua pengecekan selesai, `ArrayList` tersebut akan berisi semua solusi dari permainan. Jika, tidak ditemukan solusi, `ArrayList` tersebut akan tetap kosong.

Alasan mengapa metode *brute force* menggunakan banyak *if block* dipilih adalah karena metode ini jauh lebih mudah dimengerti dibandingkan dengan metode yang menggunakan rekursi atau metode-metode lainnya karena metode-metode tersebut lebih sulit untuk dipahami.

BAB II
SOURCE CODE

2.1 Game.java

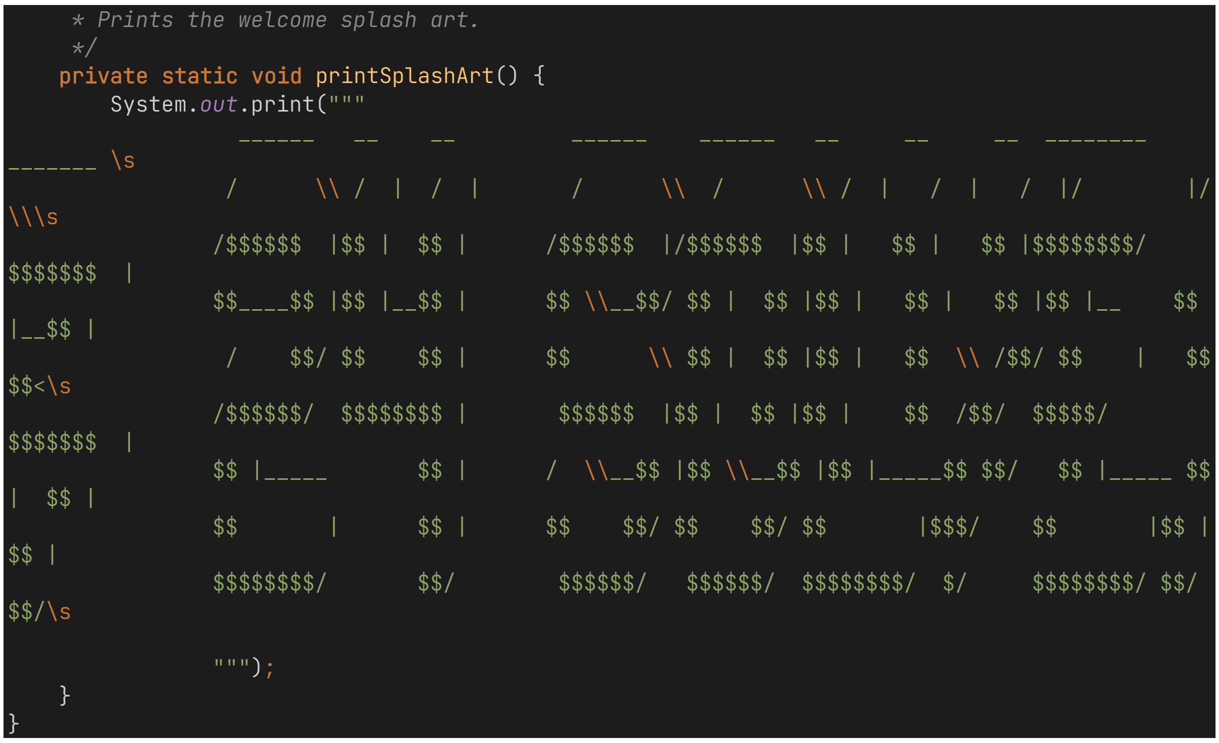
```java
package com.solver;

import java.io.FileWriter;
import java.io.PrintWriter;
import java.util.*;

/**
 * The {@code Game} class implements user input, input validation, solution finding, and file
output.
 */
public class Game {

    private static final String[] VALID_INPUTS = {
            "A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"
    };

    private static final String[] DECK_OF_CARDS = {
            "A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K",
            "A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K",
            "A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K",
            "A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"
    };

    private static final double TARGET = 24.0;

    // An array of integer containing four numeric values corresponding to the cards that are
being played
    private int[] cards;

    // All possible solutions for the cards that are being played
    private ArrayList<String> solutions;

    // Execution time of the brute force algorithm used to find all possible solutions to the
puzzle
    private double executionTime;

    /**
     * Handles user input for the cards, validates it, and determines the numeric value for
each card.
     * Sets {@code this.cards} to be an array of integers containing the corresponding numeric
values of
     * four user-inputted cards, e.g. "1 4 11 13"
     */
    void inputCardsFromUser() {
        String finalCardString = "";
        Scanner cardScanner = new Scanner(System.in);

        while (true) {

            System.out.println("Valid cards are A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, and K");
            System.out.println("Input your cards seperated by a space, e.g. \"A 10 K 3\".");
            System.out.print(">>> ");

            try {
                String cardsInput = cardScanner.nextLine();

                // Split the input string with one or more whitespace as the delimiter
                String[] arrayOfCards = cardsInput.split("\\s+");
                if (arrayOfCards.length != 4) {
                    throw new InputMismatchException();
                }

                for (int i = 0; i < arrayOfCards.length; i++) {
                    if (Arrays.asList(VALID_INPUTS).contains(arrayOfCards[i])) {
                        if (i == 3) {
                            arrayOfCards[i] = convertCards(arrayOfCards[i]);
                            finalCardString += String.format("%s", arrayOfCards[i]);
                            throw new NoSuchElementException();
                        }
                        else {
                            arrayOfCards[i] = convertCards(arrayOfCards[i]);
                            finalCardString += String.format("%s ", arrayOfCards[i]);
                        }
                    }
                    else {
                        throw new InputMismatchException();
                    }
                }
            }
            catch (InputMismatchException e) {
                System.out.print("Invalid input.\n\n");
            }
            catch (NoSuchElementException e) {
                break;
            }

        }

        String[] finalCardStringArray = finalCardString.split("\\s+");

        // Constructs a new array of integer containing the corresponding numeric value of
user-inputted cards
        this.cards = new int[4];
        for (int i = 0; i < this.cards.length; i++) {
            this.cards[i] = Integer.parseInt(finalCardStringArray[i]);
        }

    }

    /**
     * Shuffles a deck of cards, takes four cards from it, and determines the numeric value
for each card.
     * Sets {@code this.cards} to be an array of integers containing the corresponding numeric
values of
     * four randomly generated cards, e.g. "1 10 2 9"
     */
    void generateCards() {

        // Constructs a separate List of Strings, backed by the DECK_OF_CARDS array, as a deck
of cards
        List<String> newDeck = new ArrayList<>(Arrays.asList(DECK_OF_CARDS));

        // Shuffles the new deck
        Collections.shuffle(newDeck);

        // Draw four cards from the shuffled deck
        List<String> drawnCardsList = newDeck.subList(0, 4);

        String[] drawnCardsArray = drawnCardsList.toArray(new String[0]);

        System.out.print("Drawn four cards: ");

        for (int i = 0; i < drawnCardsArray.length; i++) {
            if (i == 3) {
```

```java
                System.out.println(drawnCardsArray[i]);
            }
            else {
                System.out.printf("%s ", drawnCardsArray[i]);
            }
        }

        for (int i = 0; i < drawnCardsArray.length; i++) {
            drawnCardsArray[i] = convertCards(drawnCardsArray[i]);
        }

        // Constructs a new array of integer containing the corresponding numeric value of
randomly generated cards
        this.cards = new int[4];
        for (int i = 0; i < this.cards.length; i++) {
            this.cards[i] = Integer.parseInt(drawnCardsArray[i]);
        }

    }


    /**
     * Converts King, Queen, Jack, and Ace to its respective numeric value.
     */
    private String convertCards(String cards) {

        if (Objects.equals(cards, "A")) {
            return "1";
        }
        if (Objects.equals(cards, "J")) {
            return "11";
        }
        if (Objects.equals(cards, "Q")) {
            return "12";
        }
        if (Objects.equals(cards, "K")) {
            return "13";
        }

        return cards;
    }

    /**
     * Implements brute force algorithm to find all possible solutions.
     * Each solution (in mathematical expression) is added to the {@code solutions} ArrayList
as a String.
     */
    void findSolutions() {

        System.out.printf("Corresponding numeric values: %d %d %d %d\n", cards[0], cards[1],
cards[2], cards[3]);

        // Constucts a new empty ArrayList of Strings
        this.solutions = new ArrayList<String>();

        double startTime = System.nanoTime();

        // Four nested for loops to generate all possible permutations
        // of the four integers in this.cards
        for (int i = 0; i < this.cards.length; i++) {
            for (int j = 0; j < this.cards.length; j++) {
                if (i == j) {
                    continue;
                }
                for (int k = 0; k < this.cards.length; k++) {
                    if (i == k || j == k) {
                        continue;
                    }
                    for (int l = 0; l < this.cards.length; l++) {
                        if (i == l || j == l || k == l) {
                            continue;
                        }

                        double a = this.cards[i];
                        double b = this.cards[j];
                        double c = this.cards[k];
                        double d = this.cards[l];

                        int intA = this.cards[i];
                        int intB = this.cards[j];
                        int intC = this.cards[k];
                        int intD = this.cards[l];

                        // All meaningful parentheses permutations with a, b, c, and d as
integers,
                        // and op as a mathematical operator (+, -, *, or /) are as follows:
                        //
                        // ((a op b) op c) op d
                        // (a op (b op c)) op d
                        // a op ((b op c) op d)
                        // a op (b op (c op d))
                        // (a op b) op (c op d)
                        //
                        // Note that the mathematical operator / is real division as opposed
to integer division.
                        //
                        // The reason for all possible permutations with zero and one pair of
brackets are not

                        // included is because their semantic would be similar with other
solutions.
                        // For example, given four integers 4, 2, 1, and 4, these solutions
have similar semantics:
                        // (4 + 2) * 1 * 4
                        // (1 * 4) * (2 + 4)
                        //
                        // Another example of two solutions with similar semantics:
                        // 11 * 2 * 1 + 2
                        // ((11 * 2) * 1) + 2
                        //
                        // So, all possible permutations with zero and one pair of brackets
are not computed

                        // to minimize the number of solutions with similar semantics.
                        //
                        // Many if blocks are used for checking, there are 320 of them to be
exact

                        // (64 possible operator permutations * 5 significant parentheses
permutations).

                        // The reason why this method is chosen is that it is much more
straightforward and

                        // easy to understand compared to methods using recursion or other
methods as they are

                        // bound to be more difficult to comprehend.

                        // ((a op b) op c) op d
                        if (((a + b) + c) + d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) + %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) + c) - d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) + %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) + c) * d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) + %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) + c) / d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) + %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) - c) + d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) - %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) - c) - d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) - %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) - c) * d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) - %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) - c) / d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) - %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) * c) + d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) * %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) * c) - d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) * %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) * c) * d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) * %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) * c) / d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) * %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) / c) + d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) / %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) / c) - d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) / %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) / c) * d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) / %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a + b) / c) / d == TARGET) {
                            this.solutions.add(String.format("((%d + %d) / %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) + c) + d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) + %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) + c) - d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) + %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) + c) * d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) + %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) + c) / d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) + %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) - c) + d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) - %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) - c) - d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) - %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) - c) * d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) - %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) - c) / d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) - %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) * c) + d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) * %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) * c) - d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) * %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) * c) * d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) * %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) * c) / d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) * %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) / c) + d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) / %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) / c) - d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) / %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) / c) * d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) / %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a - b) / c) / d == TARGET) {
                            this.solutions.add(String.format("((%d - %d) / %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) + c) + d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) + %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) + c) - d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) + %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) + c) * d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) + %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) + c) / d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) + %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) - c) + d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) - %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) - c) - d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) - %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) - c) * d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) - %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) - c) / d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) - %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) * c) + d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) * %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) * c) - d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) * %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) * c) * d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) * %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) * c) / d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) * %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) / c) + d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) / %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) / c) - d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) / %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) / c) * d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) / %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a * b) / c) / d == TARGET) {
                            this.solutions.add(String.format("((%d * %d) / %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) + c) + d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) + %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) + c) - d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) + %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) + c) * d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) + %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) + c) / d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) + %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) - c) + d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) - %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) - c) - d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) - %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) - c) * d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) - %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) - c) / d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) - %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) * c) + d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) * %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) * c) - d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) * %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) * c) * d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) * %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) * c) / d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) * %d) / %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) / c) + d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) / %d) + %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) / c) - d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) / %d) - %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) / c) * d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) / %d) * %d", intA,
intB, intC, intD));
                        }
                        if (((a / b) / c) / d == TARGET) {
                            this.solutions.add(String.format("((%d / %d) / %d) / %d", intA,
intB, intC, intD));
                        }

                        // (a op (b op c)) op d
                        if ((a + (b + c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d + %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b + c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d + %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b + c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d + %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b + c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d + %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b - c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d - %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b - c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d - %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b - c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d - %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b - c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d - %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b * c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d * %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b * c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d * %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b * c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d * %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b * c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d * %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b / c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d / %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b / c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d / %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b / c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d / %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a + (b / c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d + (%d / %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b + c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d + %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b + c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d + %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b + c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d + %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b + c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d + %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b - c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d - %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b - c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d - %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b - c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d - %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b - c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d - %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b * c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d * %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b * c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d * %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b * c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d * %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b * c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d * %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b / c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d / %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b / c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d / %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b / c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d / %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a - (b / c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d - (%d / %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b + c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d + %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b + c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d + %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b + c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d + %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b + c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d + %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b - c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d - %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b - c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d - %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b - c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d - %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b - c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d - %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b * c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d * %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b * c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d * %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b * c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d * %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b * c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d * %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b / c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d / %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b / c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d / %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b / c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d / %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a * (b / c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d * (%d / %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b + c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d + %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b + c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d + %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b + c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d + %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b + c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d + %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b - c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d - %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b - c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d - %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b - c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d - %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b - c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d - %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b * c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d * %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b * c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d * %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b * c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d * %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b * c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d * %d)) / %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b / c)) + d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d / %d)) + %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b / c)) - d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d / %d)) - %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b / c)) * d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d / %d)) * %d", intA,
intB, intC, intD));
                        }
                        if ((a / (b / c)) / d == TARGET) {
                            this.solutions.add(String.format("(%d / (%d / %d)) / %d", intA,
intB, intC, intD));
                        }

                        // a op ((b op c) op d)
                        if (a + ((b + c) + d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d + %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b + c) - d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d + %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b + c) * d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d + %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b + c) / d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d + %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b - c) + d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d - %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b - c) - d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d - %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b - c) * d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d - %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b - c) / d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d - %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b * c) + d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d * %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b * c) - d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d * %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b * c) * d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d * %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b * c) / d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d * %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b / c) + d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d / %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b / c) - d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d / %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b / c) * d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d / %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a + ((b / c) / d) == TARGET) {
                            this.solutions.add(String.format("%d + ((%d / %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b + c) + d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d + %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b + c) - d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d + %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b + c) * d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d + %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b + c) / d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d + %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b - c) + d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d - %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b - c) - d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d - %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b - c) * d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d - %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b - c) / d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d - %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b * c) + d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d * %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b * c) - d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d * %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b * c) * d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d * %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b * c) / d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d * %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b / c) + d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d / %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b / c) - d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d / %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b / c) * d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d / %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a - ((b / c) / d) == TARGET) {
                            this.solutions.add(String.format("%d - ((%d / %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b + c) + d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d + %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b + c) - d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d + %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b + c) * d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d + %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b + c) / d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d + %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b - c) + d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d - %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b - c) - d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d - %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b - c) * d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d - %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b - c) / d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d - %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b * c) + d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d * %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b * c) - d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d * %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b * c) * d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d * %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b * c) / d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d * %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b / c) + d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d / %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b / c) - d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d / %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b / c) * d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d / %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a * ((b / c) / d) == TARGET) {
                            this.solutions.add(String.format("%d * ((%d / %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b + c) + d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d + %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b + c) - d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d + %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b + c) * d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d + %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b + c) / d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d + %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b - c) + d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d - %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b - c) - d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d - %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b - c) * d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d - %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b - c) / d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d - %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b * c) + d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d * %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b * c) - d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d * %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b * c) * d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d * %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b * c) / d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d * %d) / %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b / c) + d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d / %d) + %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b / c) - d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d / %d) - %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b / c) * d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d / %d) * %d)", intA,
intB, intC, intD));
                        }
                        if (a / ((b / c) / d) == TARGET) {
                            this.solutions.add(String.format("%d / ((%d / %d) / %d)", intA,
intB, intC, intD));
                        }

                        // a op (b op (c op d))
                        if (a + (b + (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d + (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b + (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d + (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b + (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d + (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b + (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d + (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b - (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d - (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b - (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d - (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b - (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d - (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b - (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d - (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b * (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d * (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b * (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d * (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b * (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d * (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b * (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d * (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b / (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d / (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b / (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d / (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b / (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d / (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a + (b / (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d + (%d / (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b + (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d + (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b + (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d + (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b + (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d + (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b + (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d + (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b - (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d - (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b - (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d - (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b - (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d - (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b - (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d - (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b * (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d * (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b * (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d * (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b * (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d * (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b * (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d * (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b / (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d / (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b / (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d / (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b / (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d / (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a - (b / (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d - (%d / (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b + (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d + (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b + (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d + (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b + (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d + (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b + (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d + (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b - (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d - (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b - (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d - (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b - (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d - (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b - (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d - (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b * (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d * (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b * (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d * (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b * (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d * (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b * (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d * (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b / (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d / (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b / (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d / (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b / (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d / (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a * (b / (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d * (%d / (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b + (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d + (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b + (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d + (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b + (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d + (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b + (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d + (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b - (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d - (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b - (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d - (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b - (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d - (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b - (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d - (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b * (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d * (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b * (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d * (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b * (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d * (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b * (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d * (%d / %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b / (c + d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d / (%d + %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b / (c - d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d / (%d - %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b / (c * d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d / (%d * %d))", intA,
intB, intC, intD));
                        }
                        if (a / (b / (c / d)) == TARGET) {
                            this.solutions.add(String.format("%d / (%d / (%d / %d))", intA,
intB, intC, intD));
                        }

                        // (a op b) op (c op d)
                        if ((a + b) + (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) + (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) + (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) + (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) + (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) + (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) + (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) + (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) - (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) - (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) - (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) - (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) - (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) - (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) - (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) - (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) * (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) * (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) * (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) * (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) * (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) * (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) * (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) * (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) / (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) / (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) / (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) / (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) / (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) / (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a + b) / (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d + %d) / (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) + (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) + (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) + (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) + (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) + (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) + (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) + (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) + (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) - (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) - (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) - (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) - (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) - (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) - (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) - (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) - (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) * (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) * (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) * (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) * (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) * (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) * (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) * (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) * (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) / (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) / (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) / (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) / (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) / (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) / (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a - b) / (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d - %d) / (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) + (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) + (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) + (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) + (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) + (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) + (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) + (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) + (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) - (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) - (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) - (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) - (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) - (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) - (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) - (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) - (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) * (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) * (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) * (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) * (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) * (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) * (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) * (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) * (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) / (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) / (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) / (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) / (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) / (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) / (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a * b) / (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d * %d) / (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) + (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) + (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) + (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) + (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) + (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) + (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) + (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) + (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) - (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) - (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) - (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) - (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) - (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) - (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) - (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) - (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) * (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) * (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) * (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) * (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) * (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) * (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) * (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) * (%d / %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) / (c + d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) / (%d + %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) / (c - d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) / (%d - %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) / (c * d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) / (%d * %d)", intA,
intB, intC, intD));
                        }
                        if ((a / b) / (c / d) == TARGET) {
                            this.solutions.add(String.format("(%d / %d) / (%d / %d)", intA,
intB, intC, intD));
                        }

                    }
                }
            }
        }

        double endTime = System.nanoTime();
        this.executionTime = (endTime - startTime) / 1000000000;

    }

    /**
     * Prints all solution Strings in {@code solutions} ArrayList
     * and determines if the user wants to save the solutions in
     * a text file. If yes, it calls {@code saveSolutions()}.
     */
    void printSolutions() {

        if (this.solutions.size() == 0) {
            System.out.println("No solutions were found.");
            return;
        }

        System.out.printf("%d solution(s) found:\n", this.solutions.size());

        for (String solution : this.solutions) {
            System.out.println(solution);
        }

    }

    /**
     * Prompts the user if they want to save the solutions to a text file or not.
     */
    void savePrompt() {

        if (this.solutions.size() == 0) {
            return;
        }
```

```java
        System.out.print("""
                Do you want to save the solutions to a text file?
                1. Yes
                2. No
                """);

        Scanner fileOptionScanner = new Scanner(System.in);

        int fileOption;

        while (true) {

            System.out.print("Enter a number (1 or 2): ");

            try {
                fileOption = fileOptionScanner.nextInt();
                if (fileOption == 1 || fileOption == 2) {
                    break;
                }
                else {
                    System.out.println("Invalid input.");
                }
            }
            catch (InputMismatchException e) {
                System.out.println("Invalid input.");
                fileOptionScanner.next();
            }

        }

        switch (fileOption) {
            case 1 -> saveSolutions();
            case 2 -> {}
        }

    }

    /**
     * Asks the user for the name of the text file, validates it,
     * writes how many solutions were found, and writes every
     * solution Strings in {@code solutions} to the file.
     * Each solution String is seperated by a newline.
     */
    private void saveSolutions() {

        System.out.println("Input file name (ends with .txt), e.g. \"solutions.txt\".");

        Scanner fileNameScanner = new Scanner(System.in);
        String fileName;

        while (true) {

            System.out.print(">>> ");

            try {
                fileName = fileNameScanner.nextLine();
                fileName = fileName.strip();
                if (fileName.endsWith(".txt")) {
                    break;
                }
                else {
                    System.out.println("Invalid input.");
                }
            }
            catch (Exception e) {
                System.out.println("Invalid input.");
            }

        }

        try (PrintWriter out = new PrintWriter(new FileWriter(fileName))) {
            out.printf("%d solution(s) found:\n", this.solutions.size());
            for (String solution : this.solutions) {
                out.println(solution);
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }

    }

    /**
     * Prints the execution time of the brute force algorithm
     * used to find all possible solutions to the puzzle.
     */
    void printExecutionTime() {
        System.out.printf("Execution time: %.7f second\n", this.executionTime);
    }
}
```

2.2 Play.java

```java
package com.solver;

import java.util.InputMismatchException;
import java.util.Scanner;

public class Play {

    /**
     * Main method that controls the flow of the game
     */
    public static void main(String[] args) {

        Game game = new Game();
        boolean running = true;
        int cardOption;
        Scanner cardOptionScanner = new Scanner(System.in);

        printSplashArt();

        while (running) {
            System.out.print("""
                    Choose an option:
                    1. Input cards manually
                    2. Randomly pick 4 cards from a deck
                    3. Exit
                    """);

            while (true) {

                System.out.print("Enter a number (1-3): ");

                try {
                    cardOption = cardOptionScanner.nextInt();
                    if (cardOption == 1 || cardOption == 2 || cardOption == 3) {
                        break;
                    }
                    else {
                        System.out.println("Invalid input.");
                    }
                }
                catch (InputMismatchException e) {
                    System.out.println("Invalid input.");
                    cardOptionScanner.next();
                }
                catch (Exception e) {
                    e.printStackTrace();
                }

            }

            switch (cardOption) {
                case 1 -> game.inputCardsFromUser();
                case 2 -> game.generateCards();
                case 3 -> {
                    System.out.println("Farewell.");
                    return;
                }
            }

            game.findSolutions();
            game.printSolutions();
            game.printExecutionTime();
            game.savePrompt();

            System.out.print("""
                    Do you wish to solve again?
                    1. Yes
                    2. No
                    """);

            int continueOption;
            Scanner continueOptionScanner = new Scanner(System.in);

            while (true) {

                System.out.print("Enter a number (1 or 2): ");

                try {
                    continueOption = continueOptionScanner.nextInt();
                    if (continueOption == 1 || continueOption == 2) {
                        break;
                    }
                    else {
                        System.out.println("Invalid input.");
                    }
                }
                catch (InputMismatchException e) {
                    System.out.println("Invalid input.");
                    continueOptionScanner.next();
                }

            }

            switch (continueOption) {
                case 1 -> {}
                case 2 -> running = false;
            }
        }

        System.out.println("Farewell.");
    }

    /**
```

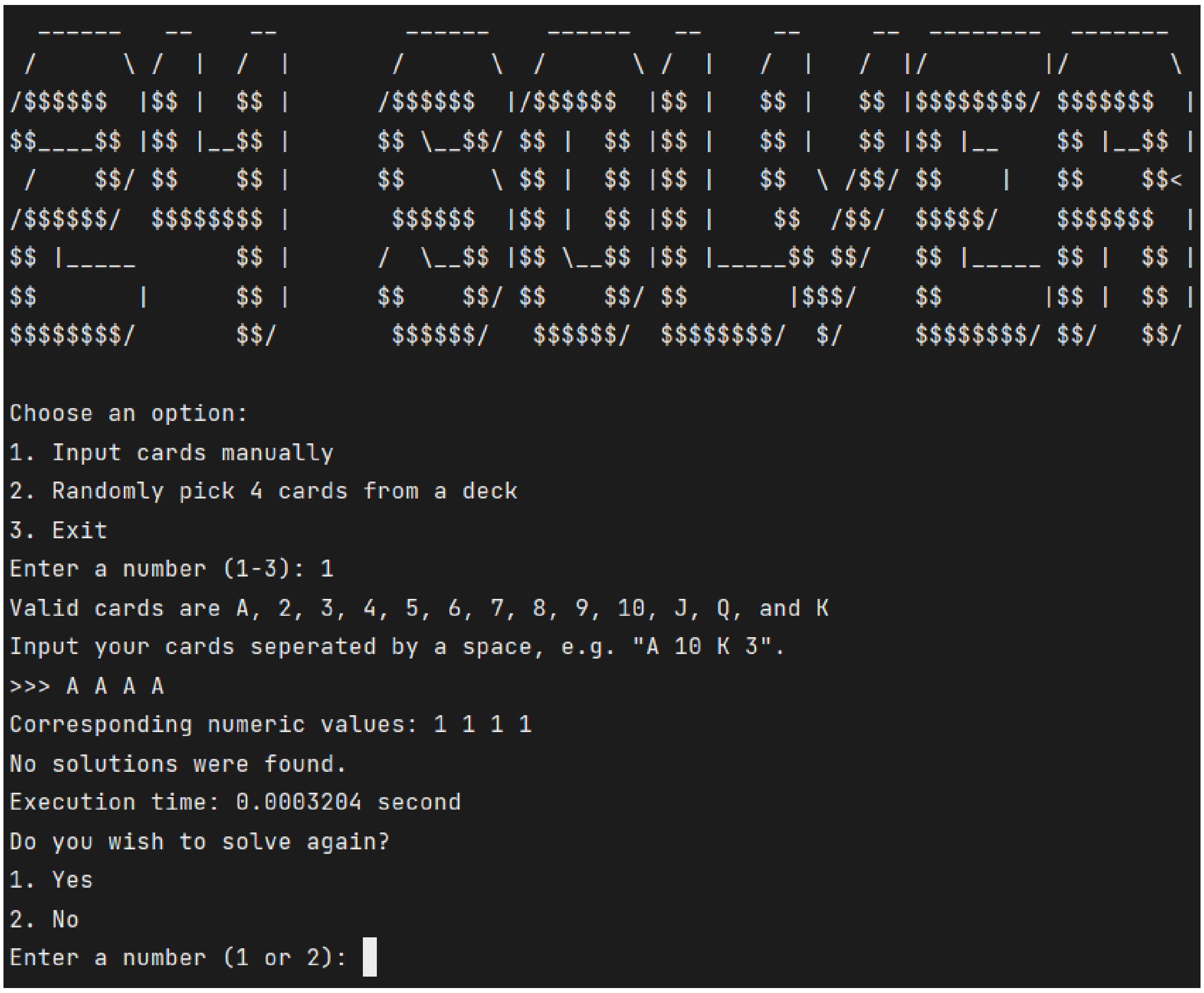


```
     * Prints the welcome splash art.
     */
    private static void printSplashArt() {
        System.out.print("""
                  ______   __    __         ______    ______   __     __     __  ________
_______ \s
                 /      \\ /  |  /  |       /      \\  /      \\ /  |   /  |   /  |/        |/
\\\s
                /$$$$$$  |$$ |  $$ |      /$$$$$$  |/$$$$$$  |$$ |   $$ |   $$ |$$$$$$$$/
$$$$$$$  |
                $$____$$ |$$ |__$$ |      $$ \\__$$/ $$ |  $$ |$$ |   $$ |   $$ |$$ |__    $$
|__$$ |
                 /    $$/ $$    $$ |      $$      \\ $$ |  $$ |$$ |   $$  \\ /$$/ $$    |   $$
$$<\s
                /$$$$$$/  $$$$$$$$ |       $$$$$$  |$$ |  $$ |$$ |    $$  /$$/  $$$$$/
$$$$$$$  |
                $$ |_____       $$ |      /  \\__$$ |$$ \\__$$ |$$ |_____$$ $$/   $$ |_____ $$
|  $$ |
                $$       |      $$ |      $$    $$/ $$    $$/ $$       |$$$/    $$        |$$ |
$$ |
                $$$$$$$$/       $$/        $$$$$$/   $$$$$$/  $$$$$$$$/  $/     $$$$$$$$/ $$/
$$/\s

                """);
    }
}
```

BAB III
EKSPERIMEN

3.1 Eksperimen

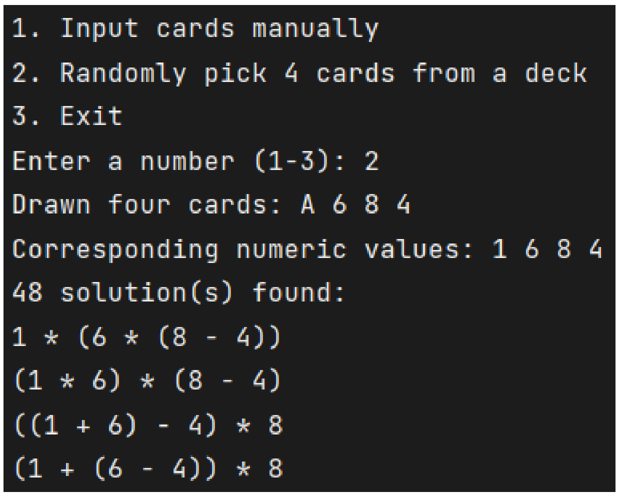
1. Kartu: A A A A



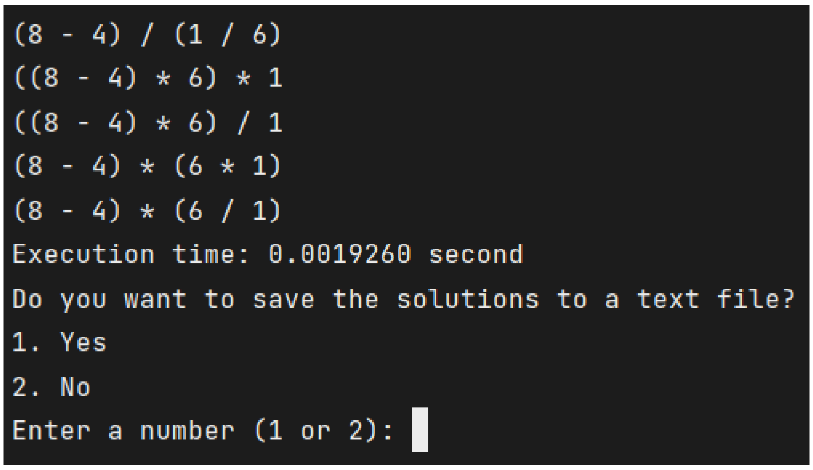
```
  ------   --    --         ------    ------   --     --     --  --------  -------
 /      \ /  |  /  |       /      \  /      \ /  |   /  |   /  |/        |/       \
/$$$$$$  |$$ |  $$ |      /$$$$$$  |/$$$$$$  |$$ |   $$ |   $$ |$$$$$$$$/ $$$$$$$  |
$$____$$ |$$ |__$$ |      $$ \__$$/ $$ |  $$ |$$ |   $$ |   $$ |$$ |__    $$ |__$$ |
 /    $$/ $$    $$ |      $$      \ $$ |  $$ |$$ |   $$  \ /$$/ $$    |   $$    $$<
/$$$$$$/  $$$$$$$$ |       $$$$$$  |$$ |  $$ |$$ |    $$  /$$/  $$$$$/    $$$$$$$  |
$$ |_____       $$ |      /  \__$$ |$$ \__$$ |$$ |_____$$ $$/    $$ |_____ $$ |  $$ |
$$       |      $$ |      $$    $$/ $$    $$/ $$       |$$$/    $$       |$$ |  $$ |
$$$$$$$$/       $$/        $$$$$$/   $$$$$$/  $$$$$$$$/  $/     $$$$$$$$/ $$/   $$/

Choose an option:
1. Input cards manually
2. Randomly pick 4 cards from a deck
3. Exit
Enter a number (1-3): 1
Valid cards are A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, and K
Input your cards seperated by a space, e.g. "A 10 K 3".
>>> A A A A
Corresponding numeric values: 1 1 1 1
No solutions were found.
Execution time: 0.0003204 second
Do you wish to solve again?
1. Yes
2. No
Enter a number (1 or 2):
```

Tidak memiliki *output file* karena tidak ditemukan solusi.

2. Kartu: A 6 8 4



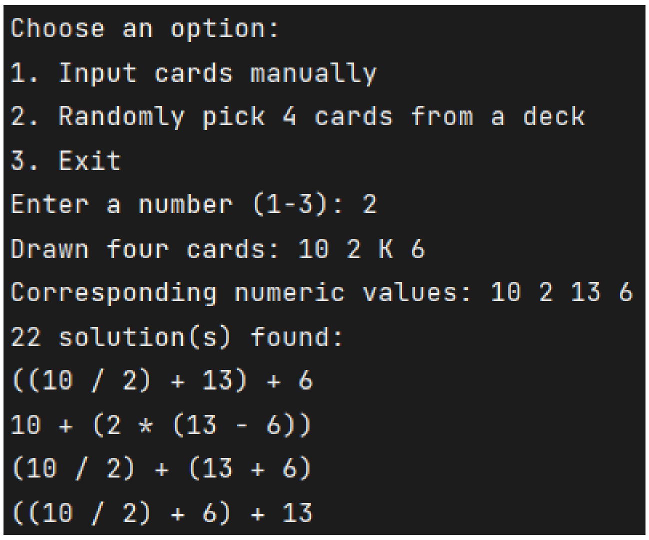
```
1. Input cards manually
2. Randomly pick 4 cards from a deck
3. Exit
Enter a number (1-3): 2
Drawn four cards: A 6 8 4
Corresponding numeric values: 1 6 8 4
48 solution(s) found:
1 * (6 * (8 - 4))
(1 * 6) * (8 - 4)
((1 + 6) - 4) * 8
(1 + (6 - 4)) * 8
```



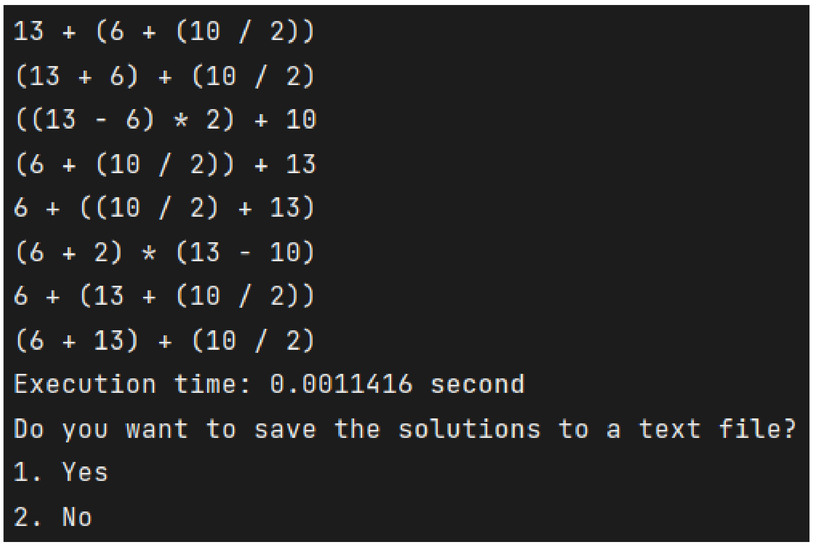
```
(8 - 4) / (1 / 6)
((8 - 4) * 6) * 1
((8 - 4) * 6) / 1
(8 - 4) * (6 * 1)
(8 - 4) * (6 / 1)
Execution time: 0.0019260 second
Do you want to save the solutions to a text file?
1. Yes
2. No
Enter a number (1 or 2):
```

Output file: `2.txt`

3. Kartu: 10 2 K 6



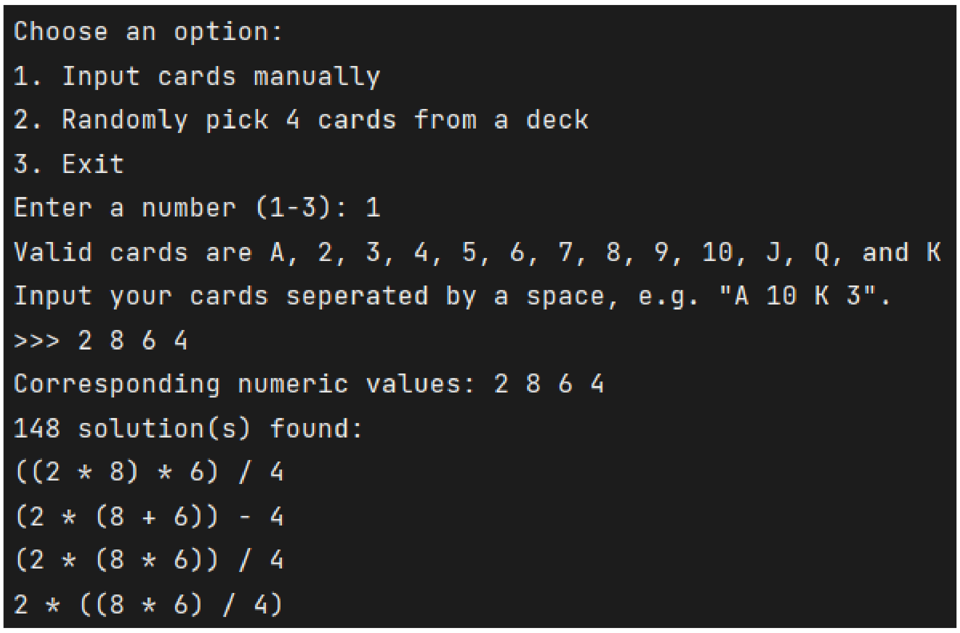
```
Choose an option:
1. Input cards manually
2. Randomly pick 4 cards from a deck
3. Exit
Enter a number (1-3): 2
Drawn four cards: 10 2 K 6
Corresponding numeric values: 10 2 13 6
22 solution(s) found:
((10 / 2) + 13) + 6
10 + (2 * (13 - 6))
(10 / 2) + (13 + 6)
((10 / 2) + 6) + 13
```



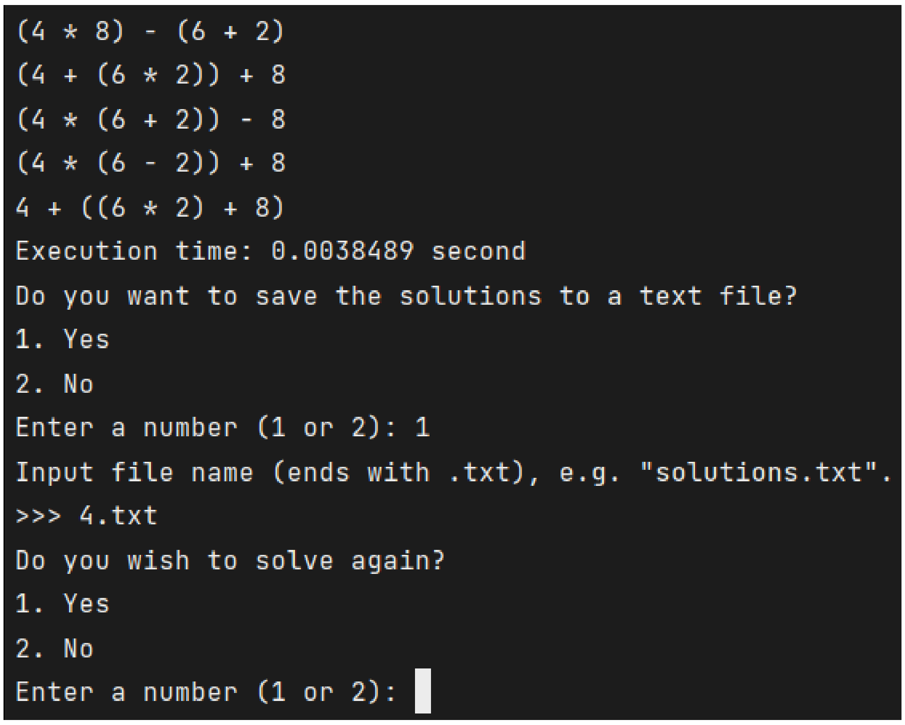
```
13 + (6 + (10 / 2))
(13 + 6) + (10 / 2)
((13 - 6) * 2) + 10
(6 + (10 / 2)) + 13
6 + ((10 / 2) + 13)
(6 + 2) * (13 - 10)
6 + (13 + (10 / 2))
(6 + 13) + (10 / 2)
Execution time: 0.0011416 second
Do you want to save the solutions to a text file?
1. Yes
2. No
```

Output file: `3.txt`

4. Kartu: 2 8 6 4



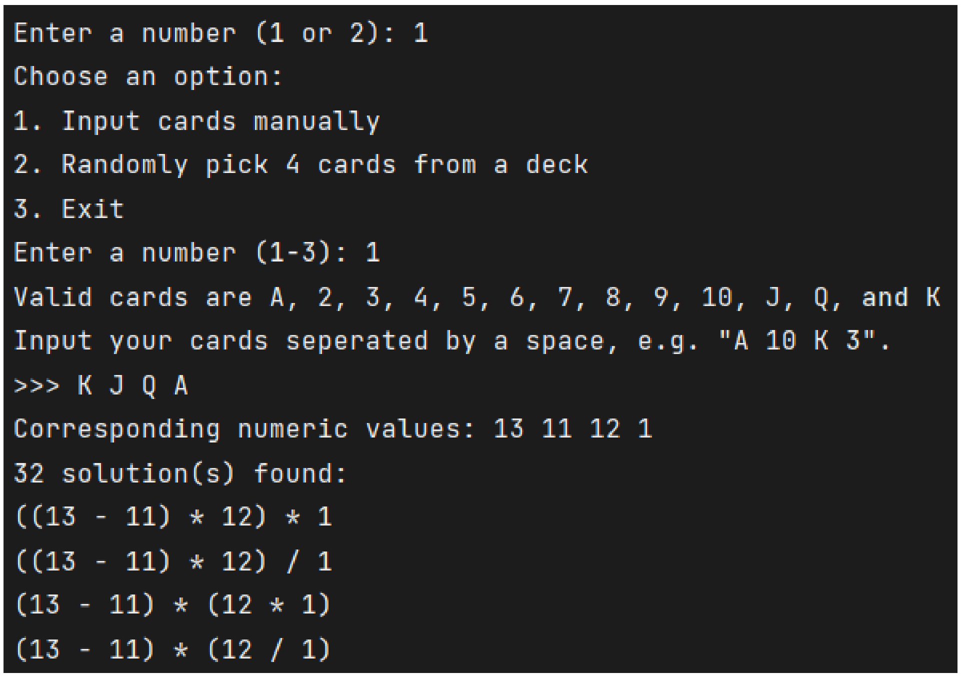
```
Choose an option:
1. Input cards manually
2. Randomly pick 4 cards from a deck
3. Exit
Enter a number (1-3): 1
Valid cards are A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, and K
Input your cards seperated by a space, e.g. "A 10 K 3".
>>> 2 8 6 4
Corresponding numeric values: 2 8 6 4
148 solution(s) found:
((2 * 8) * 6) / 4
(2 * (8 + 6)) - 4
(2 * (8 * 6)) / 4
2 * ((8 * 6) / 4)
```



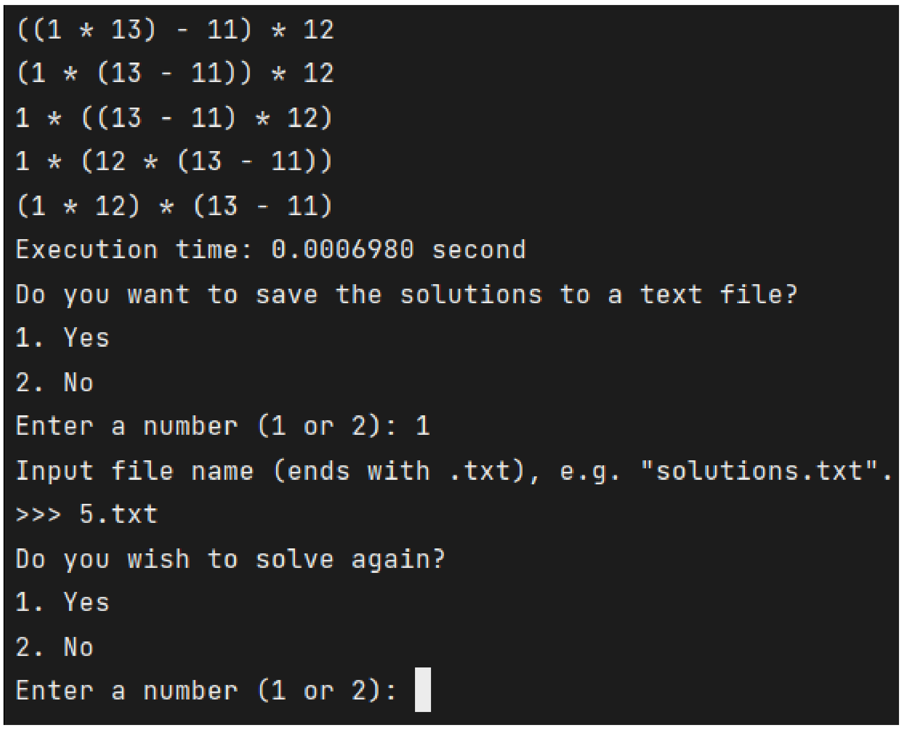
```
(4 * 8) - (6 + 2)
(4 + (6 * 2)) + 8
(4 * (6 + 2)) - 8
(4 * (6 - 2)) + 8
4 + ((6 * 2) + 8)
Execution time: 0.0038489 second
Do you want to save the solutions to a text file?
1. Yes
2. No
Enter a number (1 or 2): 1
Input file name (ends with .txt), e.g. "solutions.txt".
>>> 4.txt
Do you wish to solve again?
1. Yes
2. No
Enter a number (1 or 2):
```

Output file: `4.txt`

5. Kartu: K J Q A



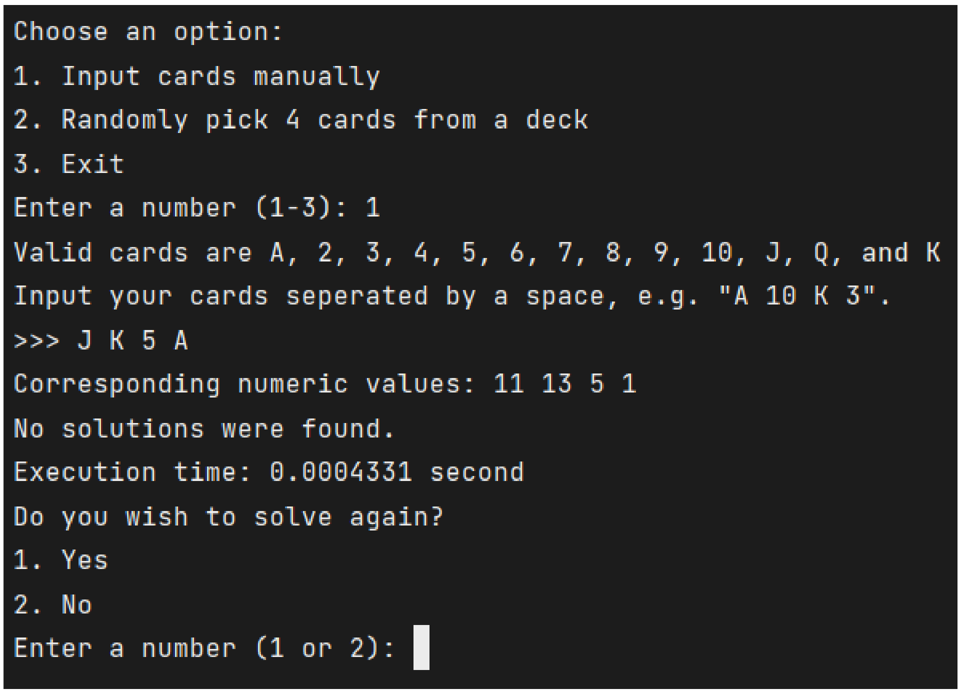
```
Enter a number (1 or 2): 1
Choose an option:
1. Input cards manually
2. Randomly pick 4 cards from a deck
3. Exit
Enter a number (1-3): 1
Valid cards are A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, and K
Input your cards seperated by a space, e.g. "A 10 K 3".
>>> K J Q A
Corresponding numeric values: 13 11 12 1
32 solution(s) found:
((13 - 11) * 12) * 1
((13 - 11) * 12) / 1
(13 - 11) * (12 * 1)
(13 - 11) * (12 / 1)
```



```
((1 * 13) - 11) * 12
(1 * (13 - 11)) * 12
1 * ((13 - 11) * 12)
1 * (12 * (13 - 11))
(1 * 12) * (13 - 11)
Execution time: 0.0006980 second
Do you want to save the solutions to a text file?
1. Yes
2. No
Enter a number (1 or 2): 1
Input file name (ends with .txt), e.g. "solutions.txt".
>>> 5.txt
Do you wish to solve again?
1. Yes
2. No
Enter a number (1 or 2): 
```

Output file: `5.txt`

6. Kartu: J K 5 A



```
Choose an option:
1. Input cards manually
2. Randomly pick 4 cards from a deck
3. Exit
Enter a number (1-3): 1
Valid cards are A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, and K
Input your cards seperated by a space, e.g. "A 10 K 3".
>>> J K 5 A
Corresponding numeric values: 11 13 5 1
No solutions were found.
Execution time: 0.0004331 second
Do you wish to solve again?
1. Yes
2. No
Enter a number (1 or 2):
```

Tidak memiliki *output file* karena tidak ditemukan solusi.

LAMPIRAN

Tautan *Remote Repository*

Berikut adalah tautan *remote repository* yang berisi *source code* program di tugas ini.
https://github.com/noelsimbolon/Tucil1_13521096

Checklist

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	✓	