Laporan Dokumentasi Tugas Solver
Desain Analisis & Algoritma

Disusun oleh :

1. **M Irfan Dhia Ulhaq (L0123078)**
2. **Muhammad Febrian Jamaludin (L0123094)**

Dosen Pengampu :
Fajar Muslim S.T., M.T

Program Studi Informatika
Fakultas Teknologi Informasi dan Sains Data
Universitas Sebelas Maret

2024

PENJABARAN PROBLEM

20 4-Solver adalah sebuah solusi untuk tantangan matematis yang melibatkan penggunaan berbagai macam kombinasi empat angka dan lima operator matematika (+, -, *, /, dan ()) untuk mencapai hasil akhir bernilai 20. Pengguna bebas memilih keempat angka tersebut, yang kemudian menciptakan dua skenario potensial:

1. Terdapat satu atau lebih kombinasi dari angka-angka dan operator-operator yang dapat menghasilkan 20.

2. Tidak ada kombinasi yang memungkinkan untuk mencapai hasil 20.

Program yang dirancang untuk menyelesaikan masalah ini akan mengeksplorasi semua kemungkinan kombinasi dan menampilkan seluruh kombinasi yang berhasil. Namun, jika tidak ada solusi yang ditemukan, program akan memberi tahu pengguna bahwa tidak ada penyelesaian yang mungkin dengan angka-angka yang telah dipilih.

Tantangan ini menggabungkan kreativitas matematis dengan kemampuan komputasi untuk menemukan solusi atau menentukan ketidakmungkinan solusi berdasarkan input yang diberikan.

DEMONSTRASI PROGRAM (DEMO)

Tutorial 20 4-solver web

1. Masuk ke link https://ryndearu.github.io/ atau jalankan file index.html
2. Masukkan 4 angka ke solver
3. Klik solve dan akan muncul semua solusi yang menghasilkan 20
4. Apabila angka yang diinput tidak memiliki solusi maka akan muncul output “Ini ga mungkin”

Penjelasan Implementasi

No.	Implementasi	PERAN
1.	Struktur HTML dan CSS dasar	<!DOCTYPE html> <html lang="id"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Solver 20</title> <style> Bagian ini mendefinisikan struktur dasar dokumen HTML. Ini menetapkan bahasa dokumen ke bahasa Indonesia, mengatur pengodean karakter ke UTF-8, mengoptimalkan tampilan untuk perangkat mobile, dan menetapkan judul halaman "Solver 20". Bagian <style> membuka bagian untuk mendefinisikan gaya CSS.
2.	Tampilan teks	body { font-family: Arial, sans-serif; max-width: 800px; margin: 0 auto; padding: 20px; text-align: center; } h1, h3 { color: #333; } Kode CSS ini mengatur tampilan dasar halaman web seperti menetapkan jenis font, lebar maksimum konten yaitu 800px, perataan teks ke tengah, dan warna untuk judul. Tujuannya adalah menciptakan tata letak yang rapi dan mudah dibaca untuk aplikasi Solver 20.
3.	Input fields	input { width: 50px; margin: 5px; padding: 5px; font-size: 16px; } Kode CSS ini mengatur tampilan kotak input angka yang akan berguna untuk user memasukkan input angka sehingga dari angka tersebut akan dicari solusinya. Mengatur lebar, margin, padding, dan ukuran font untuk memastikan input terlihat jelas dan mudah digunakan oleh user, baian ini penting untuk memudahkan pengguna memasukkan empat angka yang diperlukan.

4.	Tombol “solve”	```css button { margin: 10px; padding: 10px 20px; font-size: 16px; background-color: #4CAF50; color: white; border: none; cursor: pointer; } button:hover { background-color: #45a049; } ``` Kode ini mendesain tombol "Solve". Mengatur warna latar, ukuran, dan efek hover. Tujuannya membuat tombol mudah dilihat dan menarik untuk diklik, meningkatkan interaktivitas pengguna dengan aplikasi.
5.	Area hasil	```css #result { margin-top: 20px; border: 1px solid #ddd; padding: 10px; text-align: left; } ``` CSS ini mengatur tampilan area hasil dari solver. Memberikan batas, padding, dan mengatur teks ke kiri sehingga semua hasil dari solver akan ditampilkan dengan baik dan memisahkan hasil perhitungan dari elemen lain, membuatnya mudah dibaca oleh pengguna.
6.	Kerangka web	``` </style> </head> <body> <h1>Solver 20</h1> <div> <input type="number" id="num1" min="0" max="99"> <input type="number" id="num2" min="0" max="99"> <input type="number" id="num3" min="0" max="99"> <input type="number" id="num4" min="0" max="99"> </div> <button onclick="solveProblem()">Solve</button> <div id="result"></div> ``` Ini adalah kerangka utama halaman web. Menyusun elemen-elemen penting seperti judul bernama “Solver 20”, 4 buah input angka dengan minimal angka 0 dan maksimal angka 99, tombol "Solve" untuk mendapat output hasil, dan area hasil. Struktur ini membentuk antarmuka pengguna untuk aplikasi Solver 20.

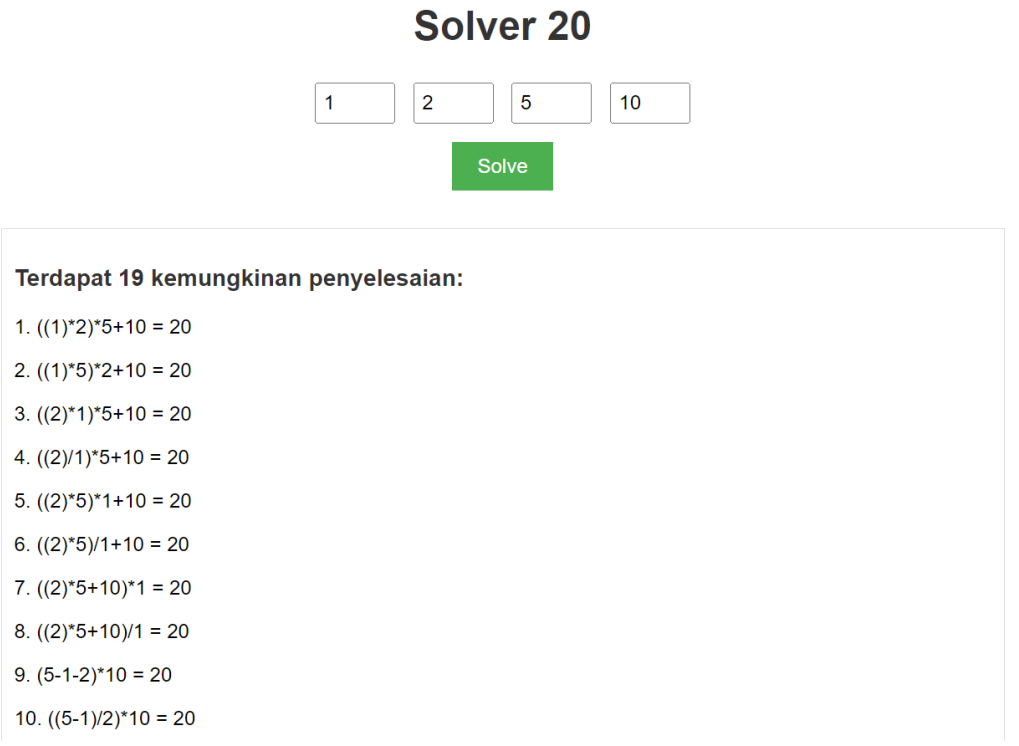
7.	Inti logika	<script> function findSolutions(numbers, target) { const operators = ['+', '-', '*', '/']; const solutions = []; function calculate(a, b, op) { switch (op) { case '+': return a + b; case '-': return a - b; case '*': return a * b; case '/': return b !== 0 ? a / b : NaN; } } Ini adalah inti logika solver. Fungsi findSolutions berperan penting untuk mencari semua kombinasi yang menghasilkan 20 menggunakan operator +, -, *, dan /, sementara calculate melakukan operasi matematika dasar pada setiap operator sehingga hasil dari kalkulasi tersebut sesuai dengan fungsi yang kita inginkan. Fungsi-fungsi ini penting untuk menemukan solusi dari angka-angka yang dimasukkan sehingga 20 4-solver tersebut dapat berjalan dengan baik dan memberikan hasil bernilai 20 menggunakan 4 angka yang akan diinput user.
8.	Fungsi addParentheses	function addParentheses(exp, op) { return op === '*' || op === '/' ? `(${exp})` : exp; } Fungsi ini menambahkan tanda kurung pada ekspresi matematika saat diperlukan untuk memastikan urutan operasi perhitungan matematika yang benar dalam ekspresi yang dihasilkan karena pada matematika dasar bilangan yang berada di dalam tanda kurung dikerjakan terlebih dahulu sebelum yang lain, dilanjutkan dengan perkalian atau pembagian, dan pertambahan dan pengurangan di akhir.
9.	Fungsi generateExpressions	function generateExpressions(nums, exp = '', usedCount = 0) { if (usedCount === 4) { if (Math.abs(eval(exp) - target) < 1e-6) { solutions.push(exp); } return; } for (let i = 0; i < nums.length; i++) { if (nums[i] !== null) { const newNums = [...nums]; const num = newNums[i]; newNums[i] = null; if (exp === '') { generateExpressions(newNums, `${num}`, usedCount + 1); } else { for (const op of operators) { const newExp = `${addParentheses(exp, op)}${op}${num}`; generateExpressions(newNums, newExp, usedCount + 1); } } } } } generateExpressions(numbers); return solutions; }

		Fungsi ini bekerja secara rekursif untuk menghasilkan dan mengevaluasi semua kemungkinan ekspresi matematika. Dimulai dengan empat angka, fungsi ini secara sistematis membangun ekspresi dengan menggabungkan angka-angka tersebut menggunakan operator matematika (+, -, *, /). Setiap kali sebuah angka digunakan, fungsi memanggil dirinya sendiri dengan sisa angka yang belum digunakan. Proses ini berlanjut hingga semua empat angka telah digunakan dalam ekspresi. Ketika sebuah ekspresi lengkap terbentuk, fungsi mengevaluasinya dan, jika hasilnya sama dengan target, menyimpannya sebagai solusi. Dengan cara ini, fungsi mengeksplorasi semua kemungkinan kombinasi untuk menemukan solusi yang valid.. Fungsi ini krusial dalam mencari semua kombinasi yang mungkin untuk mencapai target 20.
10.	Fungsi solveProblem	`function solveProblem() {` `  const num1 = parseInt(document.getElementById('num1').value);` `  const num2 = parseInt(document.getElementById('num2').value);` `  const num3 = parseInt(document.getElementById('num3').value);` `  const num4 = parseInt(document.getElementById('num4').value);` `  const numbers = [num1, num2, num3, num4];` `  if (numbers.some(isNaN)) {` `  document.getElementById('result').innerHTML = "Mohon masukkan 4 angka valid.";` `  return;` `  }` `  const solutions = findSolutions(numbers, 20);` `  const resultDiv = document.getElementById('result');` `  if (solutions.length > 0) {` `  resultDiv.innerHTML = ` Terdapat ${solutions.length} kemungkinan penyelesaian: `;` `  solutions.forEach((solution, index) => {` `  resultDiv.innerHTML += ` ${index + 1}. ${solution} = 20 `;` `  });` `  } else {` `  resultDiv.innerHTML = "<p>Ini ga mungkin</p>";` `  }` `}` Fungsi ini menangani interaksi pengguna dengan program dengan mengambil input dari 4 angka yang akan digunakan oleh solver, memvalidasi, memanggil findSolutions untuk melakukan operasi matematika sehingga ditampilkan segala kemungkinan operasi dari 4 angka tersebut yang bernilai 20, dan menampilkan hasil dari semua operasi. Ini adalah jembatan antara input pengguna dan logika pemecahan masalah.
11.	Struktur penutup HTML	`  </script>` `</body>` `</html>` Bagian ini menutup struktur HTML, memastikan semua elemen terdefinisi dengan benar dan script JavaScript termasuk dalam dokumen. Ini penting untuk fungsi keseluruhan halaman web.

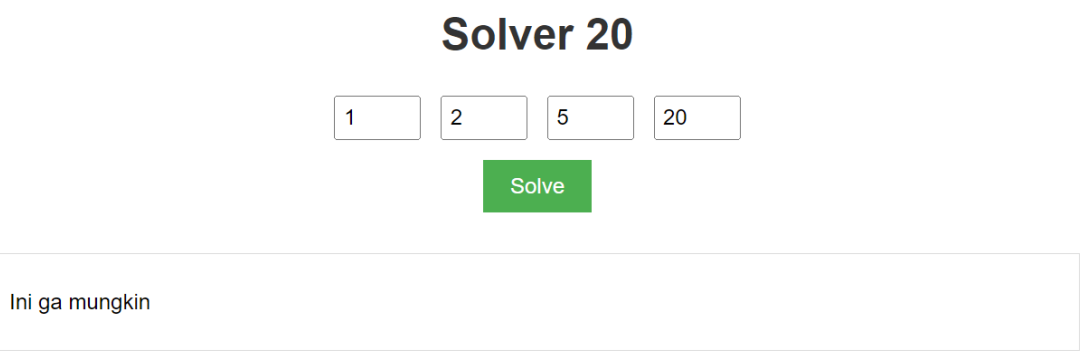
PENGUJIAN KODE

1. Tampilan UI

2. Program menemukan solusi



3. Ketika keempat angka tidak memenuhi



PERAN ANGGOTA KELOMPOK	
Anggota	Peran
M Irfan Dhia Ulhaq (L0123078)	Menyusun laporan
Muhamamd Febrian Jamaludin (L0123094)	Membuat keseluruhan kode dan membuat video penjelasan.