LAPORAN 20 SOLVER

DESAIN & ANALISA ALGORITMA

DISUSUN OLEH :

Michael Johanes Johansyah L0123081

Muhammad Firman Ghani L0123087

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2024

BAB I

PENJABARAN PROBLEM

20 Solver adalah permainan dengan mengkombinasikan beberapa angka dan operasi yang memiliki hasil akhir 20. Pada permainan ini, kita harus mencari solusi sebanyak mungkin dengan mengkombinasi operasi matematika seperti penambahan (+), pengurangan (-), perkalian (*), pembagian (/) dan tanda kurung (‘(‘ dan ‘)’) pada angka yang dimasukkan dengan urutan angka sesuai urutan saat menginput. Urutan angka dan jenis operasi yang digunakan dapat bervariasi sehingga akan ada banyak kemungkinan kombinasi yang terjadi. dan urutan serta jenis operasinya dapat bervariasi..

Solusi yang digunakan untuk menyelesaikan masalah ini adalah algoritma Brute-Force, yaitu mencoba semua kemungkinan solusi untuk menemukan hasil yang valid. Kompleksitas dalam menentukan semua kemungkinan kombinasi angka dan operasi menjadi tantangan pada permainan ini. Sebab, semakin banyak angka yang digunakan, semakin besar jumlah kombinasi yang mungkin dihasilkan.

BAB II

PENJELASAN IMPLEMENTASI

<table>
<tr><th>No.</th><th>Implementasi</th><th>Penjelasan</th></tr>
<tr><td>1.</td><td>Menerima Input</td><td>

```
document.querySelectorAll("input[type='number']").forEach(function (input) {
  input.addEventListener("input", function () {
    if (input.value < 1 || input.value > 30) {
      input.value = 1;
    }
  });
});
```

Kode ini bertujuan untuk menerima input yang dimasukkan pada tampilan web, dan membatasi nilainya hanya dari nilai 1 sampai 30, dengan nilai awal yaitu 1.

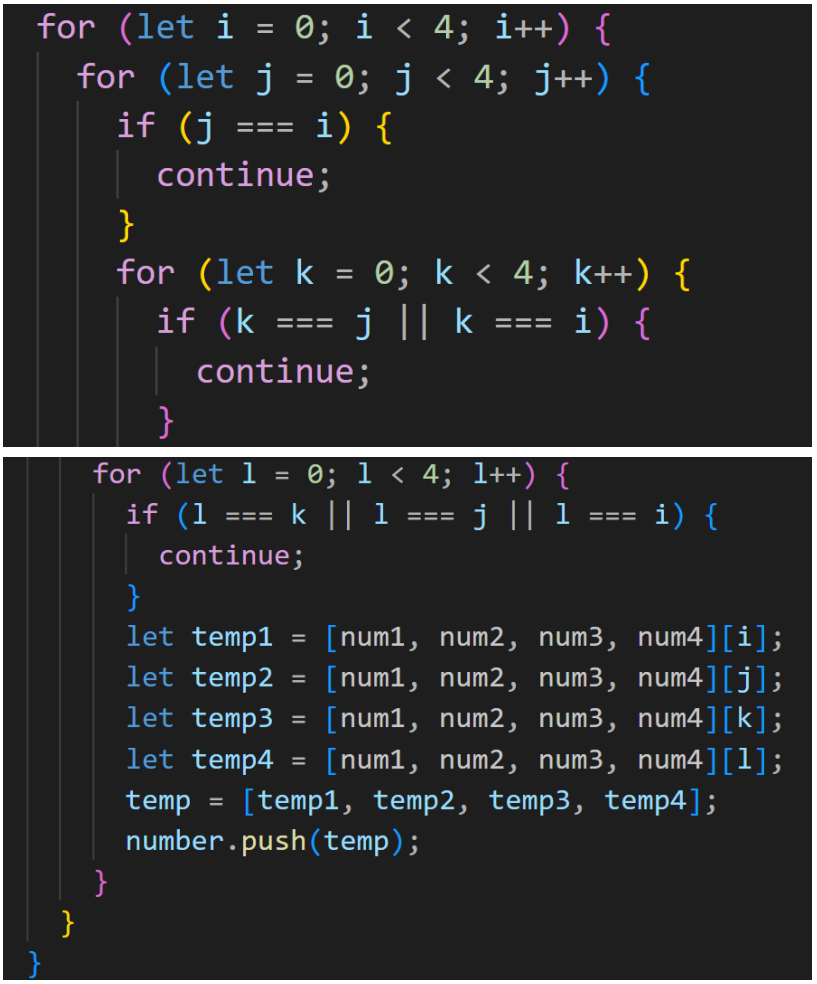
```
const num1 = parseInt(document.getElementById("num1").value);
const num2 = parseInt(document.getElementById("num2").value);
const num3 = parseInt(document.getElementById("num3").value);
const num4 = parseInt(document.getElementById("num4").value);
```

Setelah itu, input yang diterima akan disimpan nilainya dengan menggunakan fungsi const, yang berfungsi sebagai variabel.
</td></tr>
<tr><td>2.</td><td>Menginisialisasi Array dan Variabel</td><td>

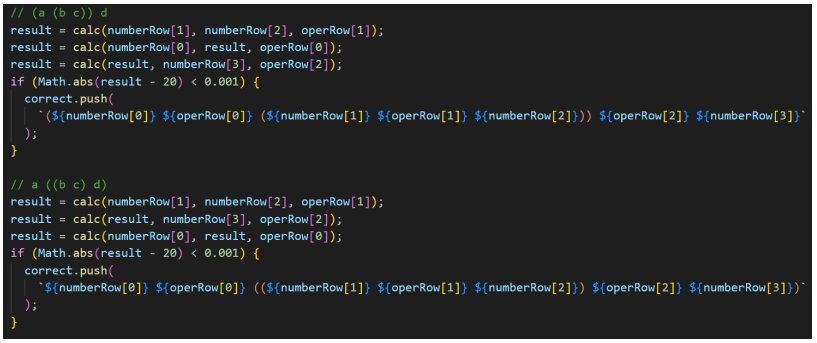
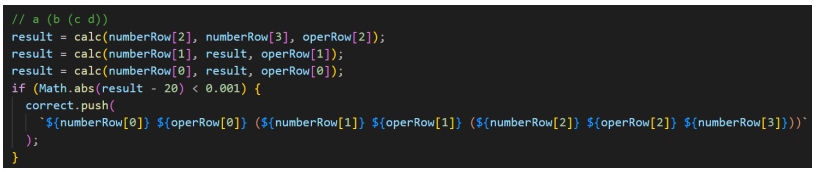
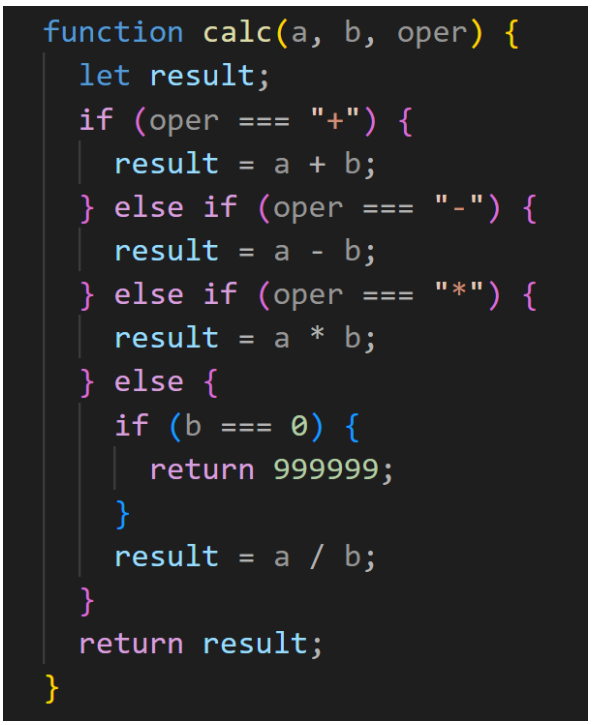
```
let temp = [0, 0, 0, 0];
let operatorList = ["+", "-", "*", "/"];
let operator = [];
let number = [];
let correct = [];
let solutionCount = 0;
```

Adapun sebelum memulai program, kita perlu untuk menyediakan beberapa variabel dan array yang akan digunakan kedepannya. Berikut ini penjelasan secara rincinya:

1. temp, yaitu suatu array yang terdiri dari 4 elemen, yang nantinya akan dimasukkan nilai dari 4 angka yang diterima pada tampilan web. Untuk nilai awalnya, sementara diisi dengan nilai 0.
2. operatorList, yaitu suatu array yang terdiri dari 4 elemen juga, namun kali ini berisi keempat simbol operasi aritmatika yang akan digunakan, yaitu +, -, *, dan /.
3. operator, yaitu suatu array yang kosong, yang nantinya akan menerima salah satu dari isi operatorList, untuk menjalankan operasi tertentu
4. number, yaitu array kosong yang akan menyimpan
</td></tr>
</table>

		angka tertentu pada suatu solusi dan membantu operasi ekspresi. 5. correct, yaitu array kosong yang akan menyimpan banyaknya solusi yang valid. 6. solutionCount, yaitu suatu variabel yang akan menyimpan jumlah dari solusi yang ditemukan, dan akan diperlihatkan pada tampilan web ketika semua operasi sudah dijalankan.
3.	Permutasi Angka	 Untuk mencegah terjadinya duplikasi angka input yang diterima, yaitu ke-4 angka yang sebelumnya sudah diambil, diperlukan permutasi angka. Kode ini menghasilkan semua kemungkinan permutasi untuk 4 angka (nums1, nums2, nums3 dan nums4), dengan jumlah total 4! = 24 kemungkinan. Berikut ini penjabaran logikanya: - Melakukan iterasi pada semua kemungkinan kombinasi dari keempat angka yang diterima, dan melewati kombinasi yang menggunakan angka yang sudah digunakan sebelumnya - Membuat array permutasi yang akan disimpan ke hasil array, dan mengulangi lagi prosesnya sampai semua permutasi dihasilkan.

		- Hasil akhirnya adalah array ‘number’ yang berisi semua kemungkinan permutasi dari ke-4 angka input. Selain itu, kita juga perlu untuk mencari kombinasi dari operator yang akan digunakan. `for (let i = 0; i < 4; i++) { for (let j = 0; j < 4; j++) { for (let k = 0; k < 4; k++) { let temp1 = operatorList[i]; let temp2 = operatorList[j]; let temp3 = operatorList[k]; let operTemp = [temp1, temp2, temp3]; operator.push(operTemp); } } }` Karena terdapat 4 elemen pada operatorList, dan masing-masing dari elemen dapat digunakan 3 kali, total kombinasinya adalah 4^3 = 64.
4.	Kalkulasi dengan tanda kurung (bracket)	Tanda kurung juga sangat penting dalam operasi aritmatika, oleh karena itu, diperlukan program khusus untuk melakukan kalkulasi dengan tanda kurung, agar dapat memprioritaskan operasi yang ada didalam tanda kurung. Kita akan menggunakan array number dan operator yang sudah disimpan sebelumnya sebagai numberRow dan operRow. `for (let numberRow of number) { for (let operRow of operator) { let result = 0;` Berikut ini kelima jenis tanda kurung beserta kodenya: `// (a b) (c d) result = calc( calc(numberRow[0], numberRow[1], operRow[0], true), calc(numberRow[2], numberRow[3], operRow[2], true), operRow[1] ); if (Math.abs(result - 20) < 0.001) { correct.push( `(${numberRow[0]} ${operRow[0]} ${numberRow[1]}) ${operRow[1]} (${numberRow[2]} ${operRow[2]} ${numberRow[3]})` ); }` `// ((a b) c) d result = calc(numberRow[0], numberRow[1], operRow[0], true); result = calc(result, numberRow[2], operRow[1], true); result = calc(result, numberRow[3], operRow[2]); if (Math.abs(result - 20) < 0.001) { correct.push( `(${numberRow[0]} ${operRow[0]} ${numberRow[1]}) ${operRow[1]} ${numberRow[2]} ${operRow[2]} ${numberRow[3]}` ); }`

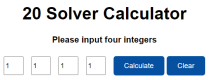
		 
5.	Kalkulasi Angka	 Fungsi ini akan melakukan operasi aritmatika, tergantung dengan simbol yang diterima. Operasi ini juga akan menghindari eror jika pembaginya adalah 0, dengan mengembalikan nilai sebesar 999999. Nilai tersebut hanyalah sebagai simbol eror, sehingga bisa juga digantikan dengan nilai lain.
6.	Fungsi Tambahan	Adapun berikut ini fungsi tambahan yang akan memperlihatkan jumlah solusi yang ditemukan berdasarkan operasi yang sudah dilakukan:

		`` solutionCount = correct.length; document.getElementById("solution-count").innerHTML = `Found ${solutionCount} solution(s)`; if (correct.length === 0) { resultContainer.innerHTML = "No solution found"; } else { const maxSolutionsPerColumn = 20; const columns = []; let currentColumn = []; for (let i = 0; i < correct.length; i++) { currentColumn.push(correct[i]); if (currentColumn.length >= maxSolutionsPerColumn) { columns.push(currentColumn); currentColumn = []; } } ``` Kode tersebut juga mengatur agar tampilan solusi dibatasi sebanyak 20 per kolom. Untuk hasil tampilan solusi akan diperlihatkan pada bagian Pengujian.
7.	Clear Button	``` document .getElementById("clear-button") .addEventListener("click", function () { // Reset angka agar kembali menjadi 1 document.getElementById("num1").value = 1; document.getElementById("num2").value = 1; document.getElementById("num3").value = 1; document.getElementById("num4").value = 1; }); ``` Kode ini bertujuan untuk me-reset angka yang sudah dimasukkan agar kembali menjadi 1. Hal ini diperlukan untuk memudahkan pengguna memasukkan input yang diinginkan.

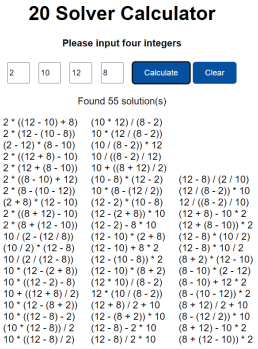
BAB III

PENGUJIAN

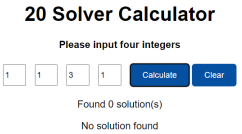
1. Tampilan Web



2. Tampilan ketika program menemukan solusi



3. Tampilan ketika program tidak menemukan solusi



BAB IV

PEMBAGIAN TUGAS

Kami mengerjakan tugas 20 Solver ini dengan rincian pembagian tugas sebagai berikut:

1. Michael Johanes Johansyah
    - Membuat source code sampai menjadi web.
    - Membuat laporan bagian penjelasan implementasi.
    - Membuat video dokumentasi.
2. Muhammad Firman Ghani
    - Membuat laporan bagian penjabaran problem dan pengujian.