UAS Struktur Data

Nama : Monica Marcellina Fadzrin

Kelas : TIF RM 24A

NIM : 24552011249

Soal 1: Binary Tree (Inorder Traversal)

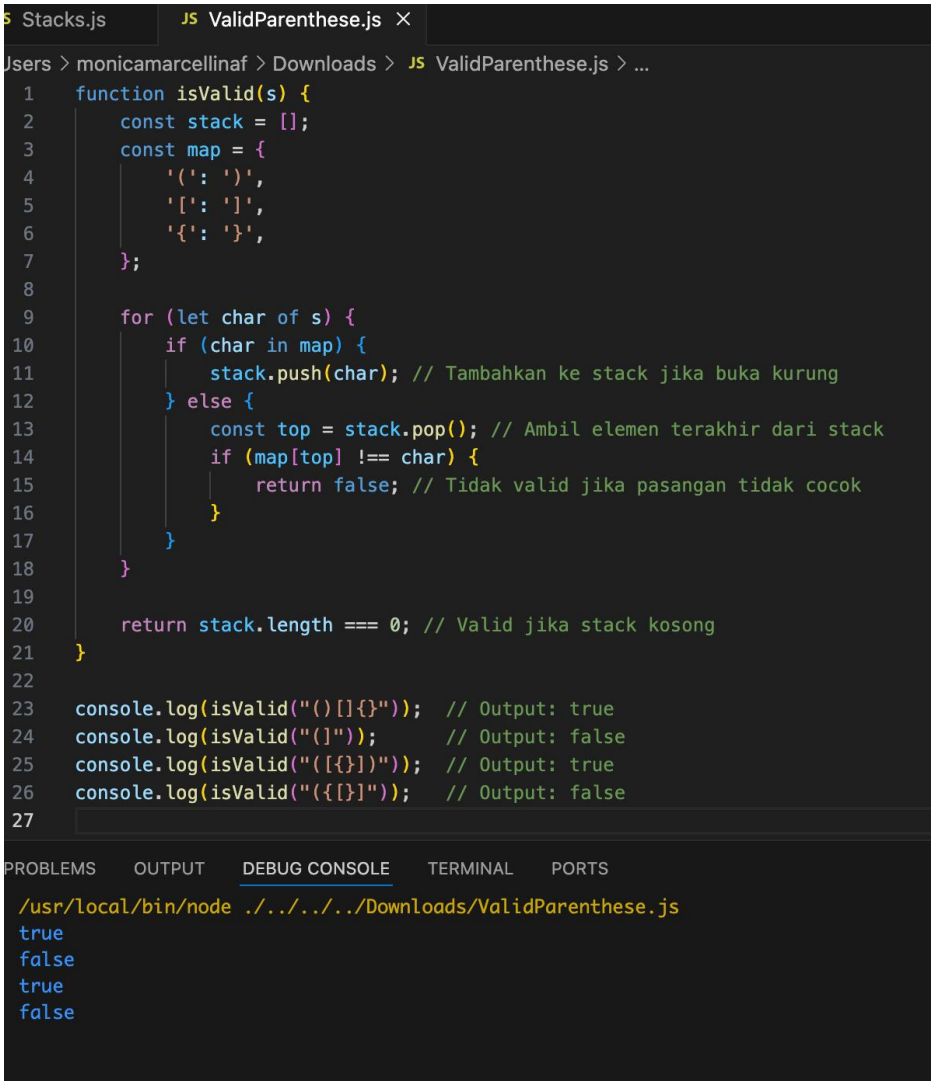
Kode Jawaban

Penjelasan :

1. Fungsi `inorderTraversal` menggunakan rekursi untuk melakukan traversal binary tree.
2. Traversal dilakukan dalam urutan inorder: kiri, akar, kanan.
3. Hasil traversal disimpan di array `result` dan dikembalikan.

Soal 2: Stack (Valid Parentheses)

Kode Jawaban

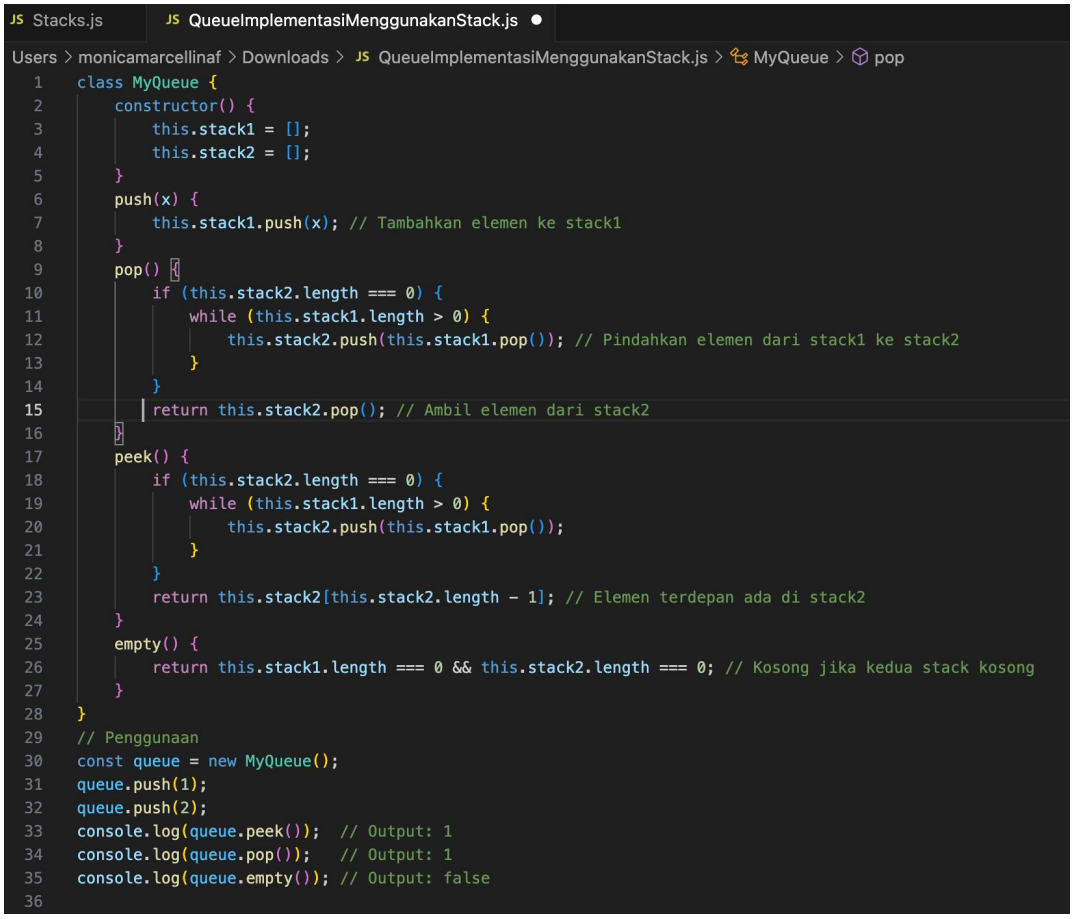


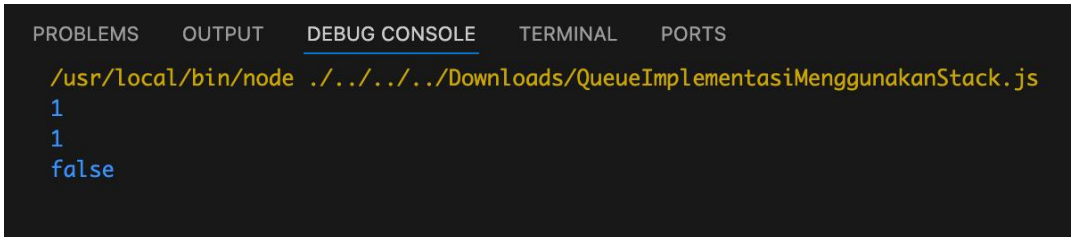
Penjelasan

1. Menggunakan stack untuk melacak pasangan kurung.
2. Jika menemukan kurung buka, tambahkan ke stack.
3. Jika menemukan kurung tutup, cocokkan dengan elemen di puncak stack.
4. String valid jika stack kosong setelah proses selesai.

Soal 3: Queue (Implementasi dengan Stack)

Kode Jawaban





Penjelasan

1. Queue diimplementasikan menggunakan dua stack (`stack1` dan `stack2`).
2. Elemen didorong ke `stack1`, lalu dipindahkan ke `stack2` untuk mengakses elemen terdepan.
3. Operasi `pop` dan `peek` memanfaatkan elemen di `stack2`.

Soal 4: Linked List (Reverse Linked List)

Kode Jawaban

Stacks.js | LinkedList.js

Users > monicamarcellinaf > Downloads > LinkedList.js > ...

```
1   class ListNode {
2       constructor(val) {
3           this.val = val;
4           this.next = null;
5       }
6   }
7
8   function reverseList(head) {
9       let prev = null;
10      let current = head;
11      while (current !== null) {
12          const nextTemp = current.next; // Simpan referensi ke node berikutnya
13          current.next = prev; // Balikkan arah pointer
14          prev = current; // Geser prev ke current
15          current = nextTemp; // Geser current ke node berikutnya
16      }
17      return prev; // prev menjadi head baru
18  }
19
20  // Membangun linked list
21  const head = new ListNode(1);
22  head.next = new ListNode(2);
23  head.next.next = new ListNode(3);
24  head.next.next.next = new ListNode(4);
25  head.next.next.next.next = new ListNode(5);
26
27  const reversedHead = reverseList(head);
28  let current = reversedHead;
29  while (current !== null) {
30      console.log(current.val); // Output: 5, 4, 3, 2, 1
31      current = current.next;
32  }
33
```

```
/usr/local/bin/node ./../../../Downloads/LinkedList.js
5
4
3
2
1
```

Penjelasan

1. Fungsi `reverseList` menggunakan iterasi untuk membalik arah pointer pada linked list.
2. Variabel `prev` digunakan untuk melacak node sebelumnya.
3. Pada akhir proses, `prev` menjadi head dari linked list yang sudah dibalik.

Soal 5: Double Linked List (Hapus Node dengan Nilai Tertentu)

Kode Jawaban

Stacks.js | DoubleLinkedList.js

Users > monicamarcellinaf > Downloads > DoubleLinkedList.js > removeElements

```
class DoublyListNode {
    constructor(val, prev = null, next = null) {
        this.val = val;
        this.prev = prev;
        this.next = next;
    }
}
function removeElements(head, val) {
    let current = head;
    while (current !== null) {
        if (current.val === val) {
            if (current.prev) {
                current.prev.next = current.next; // Hubungkan prev ke next
            }
            if (current.next) {
                current.next.prev = current.prev; // Hubungkan next ke prev
            }
            if (current === head) {
                head = current.next; // Perbarui head jika node pertama dihapus
            }
        }
        current = current.next; // Pindah ke node berikutnya
    }
    return head;
}
// Membangun double linked list
const head = new DoublyListNode(1);
head.next = new DoublyListNode(2, head);
head.next.next = new DoublyListNode(3, head.next);
head.next.next.next = new DoublyListNode(2, head.next.next);
head.next.next.next.next = new DoublyListNode(4, head.next.next.next);

const newHead = removeElements(head, 2);
let current = newHead;
while (current !== null) {
    console.log(current.val); // Output: 1, 3, 4
    current = current.next;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/local/bin/node ./../../../Downloads/DoubleLinkedList.js
1
3
4
```

Penjelasan

1. Fungsi `removeElements` menghapus node dengan nilai tertentu.
2. Node sebelumnya (`prev`) dan berikutnya (`next`) dihubungkan untuk melewati node yang dihapus.
3. Jika node pertama dihapus, head diperbarui ke node berikutnya.