

UAS Struktur Data

Nama : Monica Marcellina Fadzrin

Kelas : TIF RM 24A

NIM : 24552011249

Soal 1: Binary Tree (Inorder Traversal)

Kode Jawaban

```
JS Stacks.js JS InorderTraversal.js X
Users > monicamarcellinaf > Downloads > JS InorderTraversal.js > TreeNode
1  class TreeNode {
2      constructor(val) {
3          this.val = val;
4          this.left = null;
5          this.right = null;
6      }
7  }
8
9  function inorderTraversal(root) {
10     const result = [];
11     function traverse(node) {
12         if (!node) return;
13         traverse(node.left); // Kunjungi anak kiri
14         result.push(node.val); // Kunjungi akar
15         traverse(node.right); // Kunjungi anak kanan
16     }
17     traverse(root);
18     return result;
19 }
20
21 // Membangun binary tree
22 const root = new TreeNode(1);
23 root.right = new TreeNode(2);
24 root.right.left = new TreeNode(3);
25
26 console.log(inorderTraversal(root)); // Output: [1, 3, 2]
27
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
/usr/local/bin/node ../../../../Downloads/InorderTraversal.js
> (3) [1, 3, 2]
```

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

```
Stacks.js  JS ValidParenthese.js X
Users > monicamarcellinaf > Downloads > JS ValidParenthese.js > ...
1  function isValid(s) {
2      const stack = [];
3      const map = {
4          '(': ')',
5          '[': ']',
6          '{': '}',
7      };
8
9      for (let char of s) {
10         if (char in map) {
11             stack.push(char); // Tambahkan ke stack jika buka kurung
12         } else {
13             const top = stack.pop(); // Ambil elemen terakhir dari stack
14             if (map[top] !== char) {
15                 return false; // Tidak valid jika pasangan tidak cocok
16             }
17         }
18     }
19
20     return stack.length === 0; // Valid jika stack kosong
21 }
22
23 console.log(isValid("()[]{}")); // Output: true
24 console.log(isValid("(")); // Output: false
25 console.log(isValid("([{}])")); // Output: true
26 console.log(isValid("({[]})")); // Output: false
27
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
/usr/local/bin/node ../../../../Downloads/ValidParenthese.js
true
false
true
false
```

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

```
JS Stacks.js JS QueueImplementasiMenggunakanStack.js •
Users > monicamarcellini > Downloads > JS QueueImplementasiMenggunakanStack.js > MyQueue > pop
1 class MyQueue {
2   constructor() {
3     this.stack1 = [];
4     this.stack2 = [];
5   }
6   push(x) {
7     this.stack1.push(x); // Tambahkan elemen ke stack1
8   }
9   pop() {
10    if (this.stack2.length === 0) {
11      while (this.stack1.length > 0) {
12        this.stack2.push(this.stack1.pop()); // Pindahkan elemen dari stack1 ke stack2
13      }
14    }
15    return this.stack2.pop(); // Ambil elemen dari stack2
16  }
17   peek() {
18     if (this.stack2.length === 0) {
19       while (this.stack1.length > 0) {
20         this.stack2.push(this.stack1.pop());
21       }
22     }
23     return this.stack2[this.stack2.length - 1]; // Elemen terdepan ada di stack2
24   }
25   empty() {
26     return this.stack1.length === 0 && this.stack2.length === 0; // Kosong jika kedua stack kosong
27   }
28 }
29 // Penggunaan
30 const queue = new MyQueue();
31 queue.push(1);
32 queue.push(2);
33 console.log(queue.peek()); // Output: 1
34 console.log(queue.pop()); // Output: 1
35 console.log(queue.empty()); // Output: false
36
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/local/bin/node ../../../../Downloads/QueueImplementasiMenggunakanStack.js
1
1
false
```

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  JS LinkedList.js X
Users > monicamarcellinaf > Downloads > JS 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 JS DoubleLinkedList.js
Users > monicamarcellinaf > Downloads > JS DoubleLinkedList.js > removeElements
1 class DoublyListNode {
2   constructor(val, prev = null, next = null) {
3     this.val = val;
4     this.prev = prev;
5     this.next = next;
6   }
7 }
8 function removeElements(head, val) {
9   let current = head;
10  while (current !== null) {
11    if (current.val === val) {
12      if (current.prev) {
13        current.prev.next = current.next; // Hubungkan prev ke next
14      }
15      if (current.next) {
16        current.next.prev = current.prev; // Hubungkan next ke prev
17      }
18      if (current === head) {
19        head = current.next; // Perbarui head jika node pertama dihapus
20      }
21    }
22    current = current.next; // Pindah ke node berikutnya
23  }
24  return head;
25 }
26 // Membangun double linked list
27 const head = new DoublyListNode(1);
28 head.next = new DoublyListNode(2, head);
29 head.next.next = new DoublyListNode(3, head.next);
30 head.next.next.next = new DoublyListNode(2, head.next.next);
31 head.next.next.next.next = new DoublyListNode(4, head.next.next.next);
32
33 const newHead = removeElements(head, 2);
34 let current = newHead;
35 while (current !== null) {
36   console.log(current.val); // Output: 1, 3, 4
37   current = current.next;
38 }
39
```

```
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.