

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

void createLinkedList(struct Node** head, int values[], int size) {
    struct Node* temp = NULL;
    for (int i = 0; i < size; i++) {
        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = values[i];
        newNode->next = NULL;

        if (*head == NULL) {
            *head = newNode;
        } else {
            temp = *head;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }
}

void insertAtBeginning(struct Node** head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}

void insertAtEnd(struct Node** head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    struct Node* temp = *head;
```

```

newNode->data = data;
newNode->next = NULL;

if (*head == NULL) {
    *head = newNode;
} else {
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}
}

void display(struct Node* head) {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }

    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;

    int values[] = {10, 20, 30};
    createLinkedList(&head, values, 3);
    printf("Linked List after creation:\n");
    display(head);

    insertAtBeginning(&head, 5);
    printf("Linked List after inserting 5 at the beginning:\n");
    display(head);
}

```

```
insertAtEnd(&head, 40);
printf("Linked List after inserting 40 at the end:\n");
display(head);

return 0;
}
```

Output

```
Linked List after creation:
10 20 30
Linked List after inserting 5 at the beginning:
5 10 20 30
Linked List after inserting 40 at the end:
5 10 20 30 40
```

Leetcode problem 20 solution

The screenshot shows a Leetcode IDE interface. The top half displays a code editor with a dark theme, containing C code for validating parentheses. The bottom half shows the results panel with tabs for 'Testcase' and 'Test Result'. The 'Testcase' tab is selected, showing 'Accepted' status and 'Runtime: 0 ms'. Below it, four test cases are listed: Case 1 (passed), Case 2 (passed), Case 3 (passed), and Case 4 (passed).

```
Code
C Auto
1 #include <stdbool.h>
2 #include <string.h>
3
4 bool isValid(char* s) {
5     int n = strlen(s);
6     char stack[n];
7     int top = -1;
8     int i = 0;
9     while (i < n)
10    {
11        char c = s[i];
12        if (c == '(' || c == '{' || c == '[')
13            stack[++top] = c;
14        else
15        {
16            if (top == -1)
17                return false;
18            char topChar = stack[top--];
19            if ((c == ')' && topChar != '(') ||
20                (c == '}' && topChar != '{') ||
21                (c == ']' && topChar != '['))
22                return false;
23        }
24        i++;
25    }
26    return top == -1;
27 }
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

- Case 1
- Case 2
- Case 3
- Case 4