LAB - 06 05/02/2024.

1. Double Linked List operations:

```
# include <stdio.h>
# include <stdlib.h>
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};
struct Node* create (int value) {
    struct Node* newnode = (struct Node*) malloc (sizeof(struct Node));
    newnode -> data = value;
    newnode -> prev = NULL;
    newnode -> next = NULL;
    struct newnode;
};
void display() {
    struct Node* curr = head;
    while (curr != NULL) {
        printf(" %d ->", curr->data);
        curr = curr->next;
    }
    printf(" NULL \n");
}
void insertleft (int value, int target) {
    struct Node* newnode = create (value);
    struct Node* curr = head;
    if ( curr == NULL || curr -> data == target) {
        newnode -> next = curr;
        if (curr != NULL) {
            curr -> prev = newnode;
        }
        head = newnode;
        return;
    }
    while (curr != NULL && curr -> data != target) {
        curr = curr -> next;
    }
    if (curr != NULL) {
        newnode → next = curr;
        newnode → prev = curr → prev;
        if (curr → prev != NULL) {
            curr → prev → next = newnode;
        }
        else {
            printf("Not found! \n");
            free(newnode);
        }
    }
}
void deletenode(int value) {
    struct Node* curr = head;
    while (curr != NULL && curr → data != value) {
        curr = curr → next;
    }
    if (curr != NULL) {
        if (curr → prev != NULL) {
            curr → prev → next = curr → next;
        }
        else {
            head = curr → next;
        }
        if (curr → next != NULL) {
            curr → next → prev = curr → prev;
        }
        free(curr);
        print("Deleted \n");
    }
    else {
        printf("Not found! \n");
    }
}
int main() {
    int choice, value, target;
    printf("\n --- Double Linked List --- \n");
    print(" 1) Create  2) Insert   3) Delete    4) Exit \n");
    while(1) {
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
switch (choice) {
case 1:
    printf(" Enter first value: ");
    scanf("%d", &value);
    head = create (value);
    display();
    break;
case 2:
    printf(" Enter value to be inserted: ");
    scanf (" %d", &value);
    printf(" Enter target node: ");
    scanf ("%d", &target);
    insertleft (value, target);
    display( );
    break;
case 3:
    printf ("Enter value to be deleted! ");
    scanf(" %d", &value);
    deletenode (value);
    display(~~head~~);
    break;
case 4:
    printf(" exiting...\n");
    exit(0);
default:
    printf(" Invalid input \n");
  }
}
return 0;
}
```

Output:

```
----Double Linked List operation Menu----
1) Create 2) Insert a new node to the left 3) Delete a node 4) Exit
Enter your choice: 3
Enter value to be deleted: 1
Not found!
NULL
Enter your choice: 1
Enter first value: 20
20 → NULL
Enter your choice: 2
Enter value to be inserted: 30
Enter target node: 20
30 → 20 → NULL
Enter your choice: 2
Enter ~~your~~ value to be inserted: 40
30 → 40 → 20 → NULL
Enter your choice: 3
Enter value to be deleted: 30
Deleted
40 → 20 → NULL
Enter your choice: 4
Exiting...
```

[illegible]

$() \rightarrow 1$

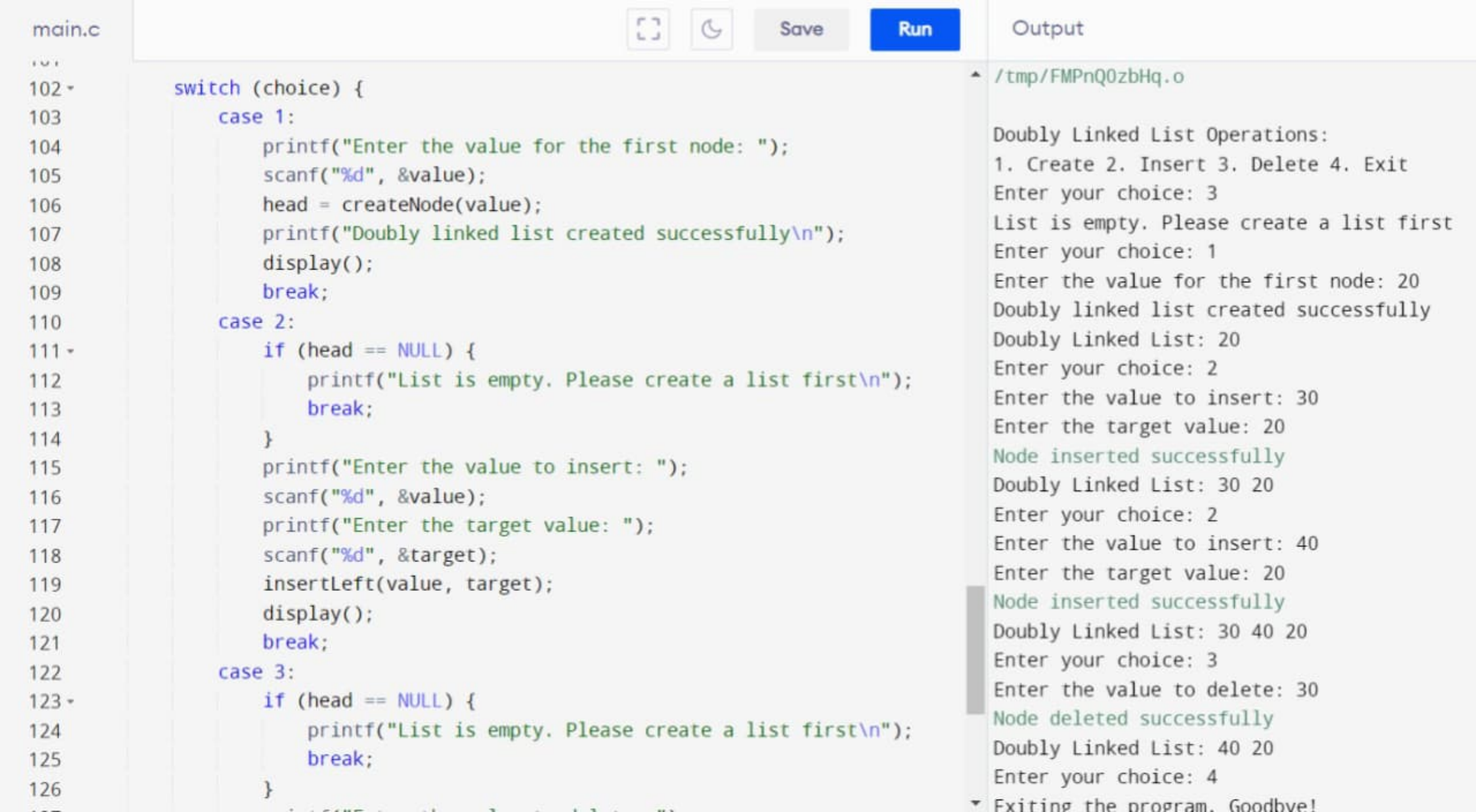
$()() \rightarrow 2$

$()()() \rightarrow 3$

$(()) \rightarrow 2$

$$\underbrace{(\underbrace{()}_{1}\underbrace{(\underbrace{()}_{1})}_{2})}_{3}$$

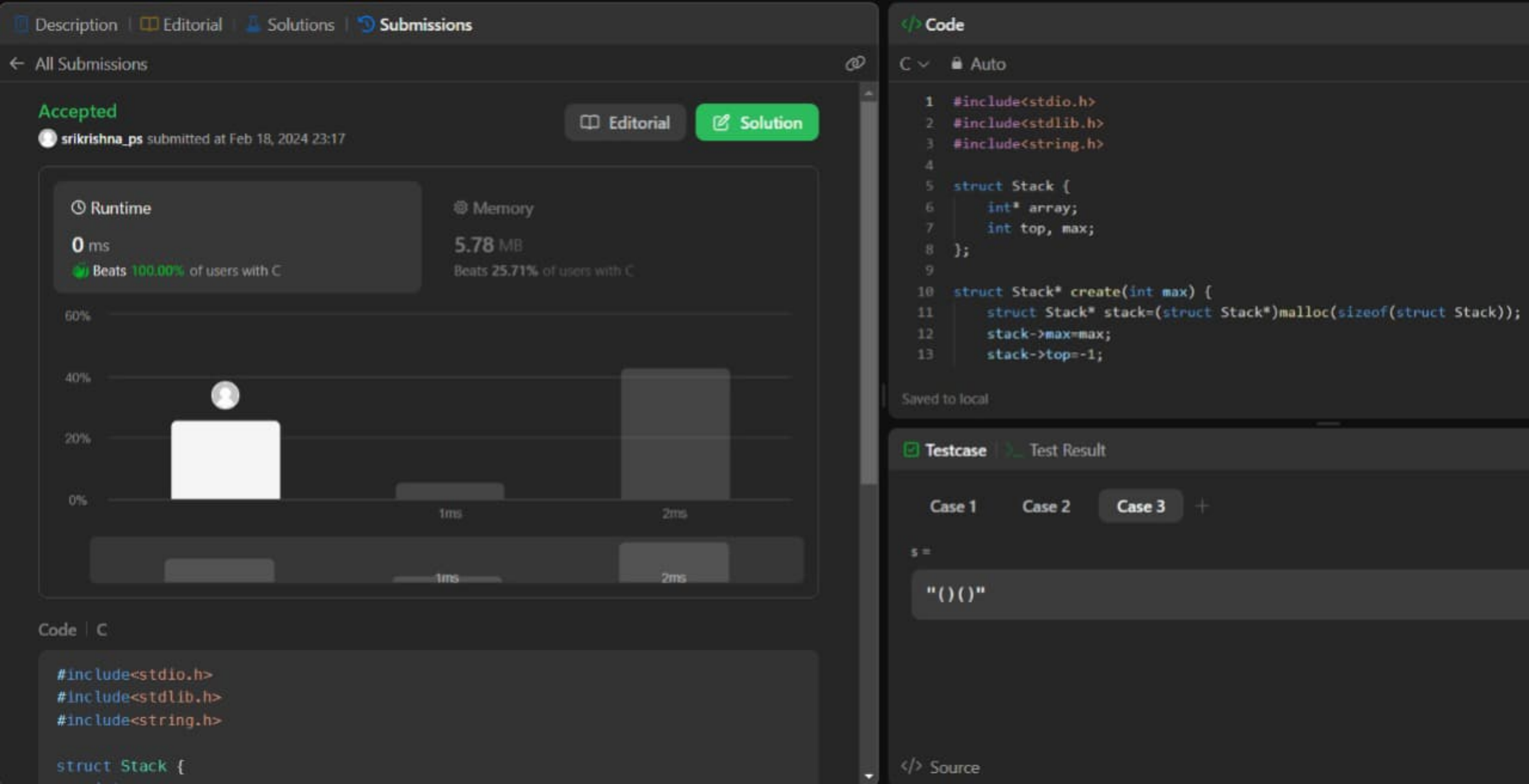
6



```c
    switch (choice) {
        case 1:
            printf("Enter the value for the first node: ");
            scanf("%d", &value);
            head = createNode(value);
            printf("Doubly linked list created successfully\n");
            display();
            break;
        case 2:
            if (head == NULL) {
                printf("List is empty. Please create a list first\n");
                break;
            }
            printf("Enter the value to insert: ");
            scanf("%d", &value);
            printf("Enter the target value: ");
            scanf("%d", &target);
            insertLeft(value, target);
            display();
            break;
        case 3:
            if (head == NULL) {
                printf("List is empty. Please create a list first\n");
                break;
            }
```

```
/tmp/FMPnQ0zbHq.o

Doubly Linked List Operations:
1. Create 2. Insert 3. Delete 4. Exit
Enter your choice: 3
List is empty. Please create a list first
Enter your choice: 1
Enter the value for the first node: 20
Doubly linked list created successfully
Doubly Linked List: 20
Enter your choice: 2
Enter the value to insert: 30
Enter the target value: 20
Node inserted successfully
Doubly Linked List: 30 20
Enter your choice: 2
Enter the value to insert: 40
Enter the target value: 20
Node inserted successfully
Doubly Linked List: 30 40 20
Enter your choice: 3
Enter the value to delete: 30
Node deleted successfully
Doubly Linked List: 40 20
Enter your choice: 4
Exiting the program. Goodbye!
```



Accepted

srikrishna_ps submitted at Feb 18, 2024 23:17

Runtime: 0 ms, Beats 100.00% of users with C

Memory: 5.78 MB, Beats 25.71% of users with C

Code | C

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

struct Stack {
    int* array;
    int top, max;
};

struct Stack* create(int max) {
    struct Stack* stack=(struct Stack*)malloc(sizeof(struct Stack));
    stack->max=max;
    stack->top=-1;
```

```
s =
"()()"
```