DSA LAB

Assignment 5

Name: Madhav Jha
Roll no.: E3-48
Branch: CSE (AI & ML)

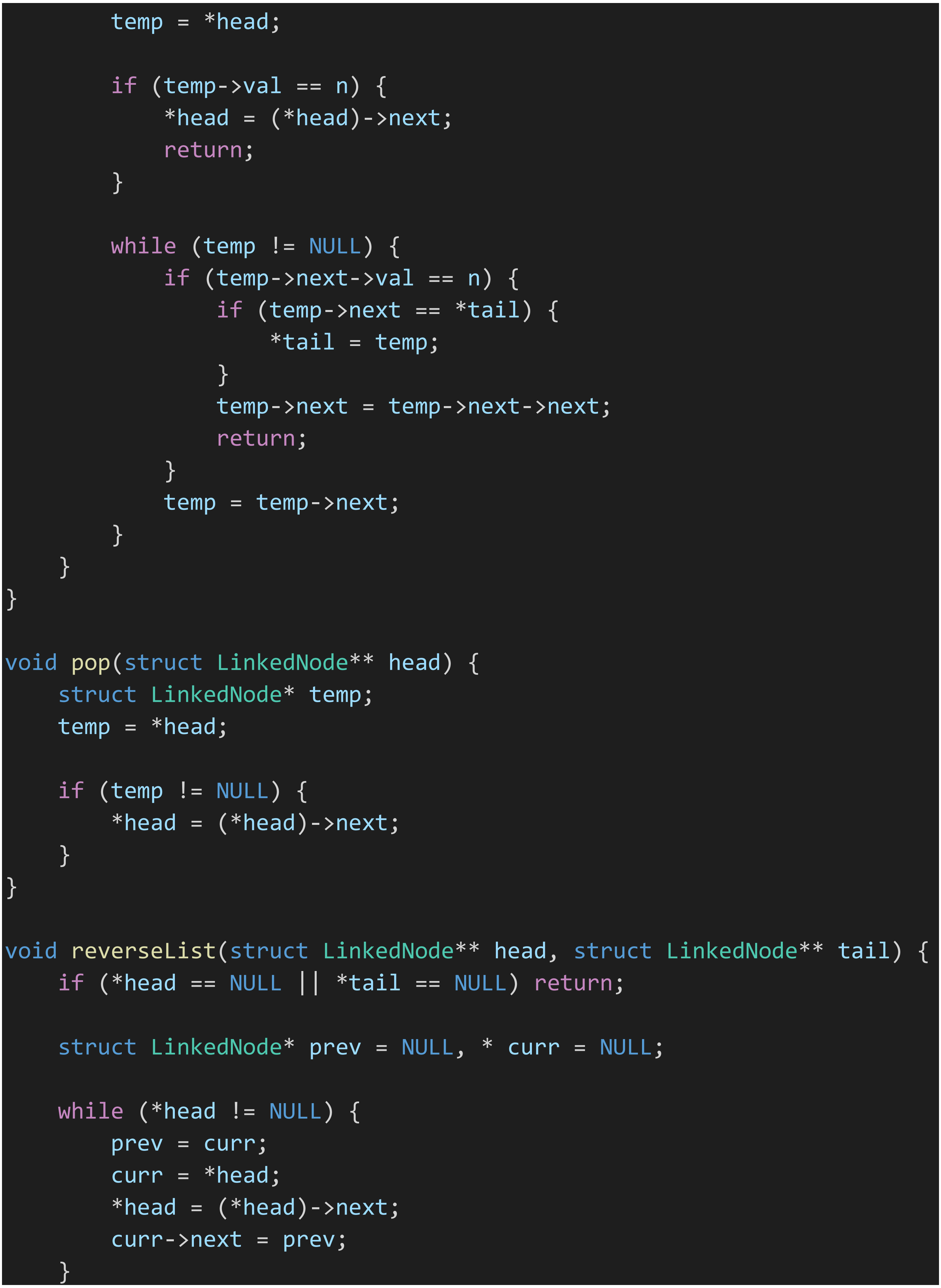
Linked List in C

AIM: “To study single linked lists and implement various operations on it – Insert, Delete, Update, Display.”

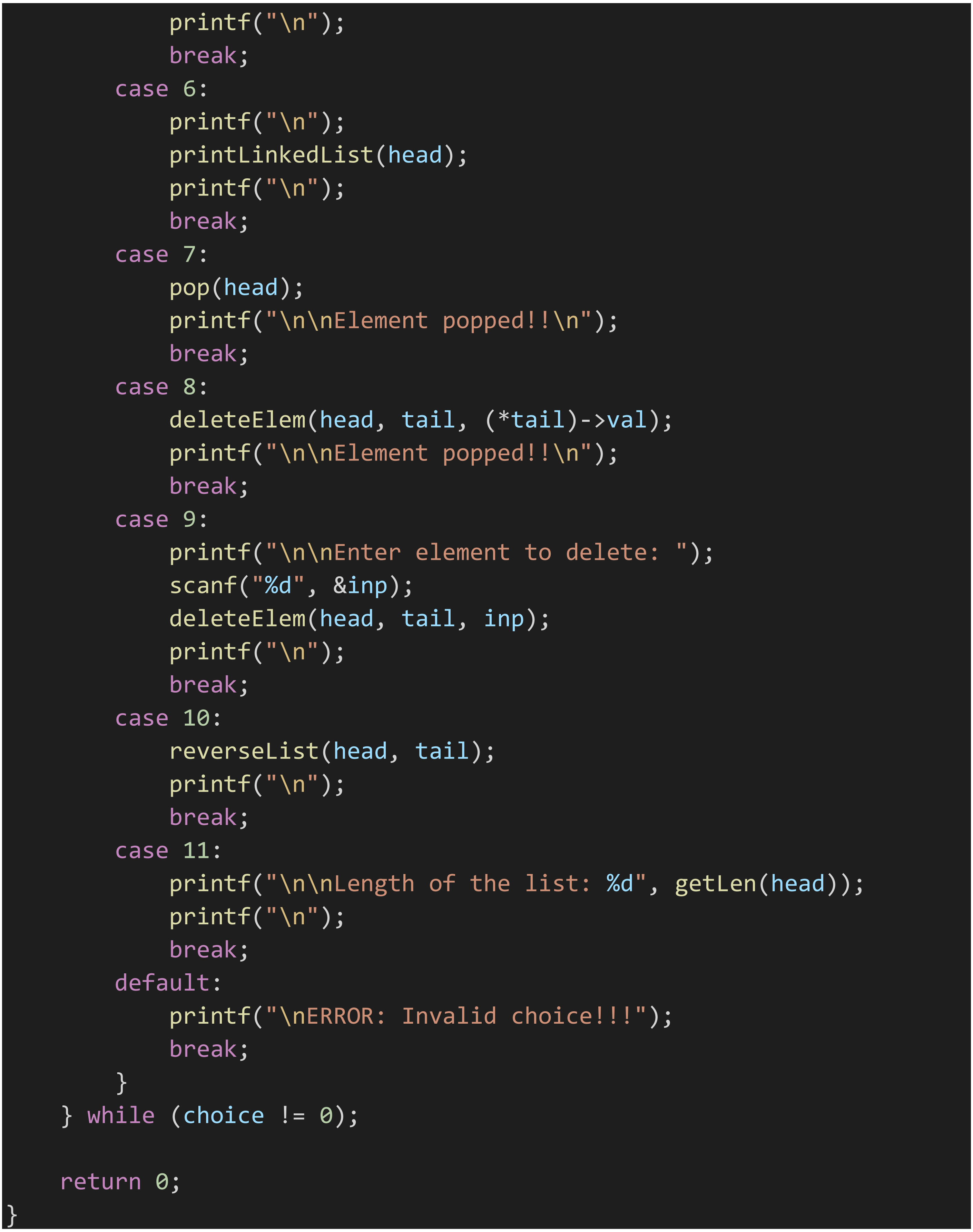
Problem Definition Create a self-referential structure, NODE to represent a node of a singly linked linear list. Implement the functions 1. create a list 2. insert an element – at the beginning, at the end and at a specified position in the list, in ordered list 3. delete an element from the front, rear, or a specified position at the list 4. reverse the list 5. find length of the list 6. search the list Create a menu-driven program to test all the functions.

Code:

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

typedef struct LinkedNode {
    int val;
    struct LinkedNode* next;
};
struct LinkedNode* head = NULL;
struct LinkedNode* tail = NULL;

int searchElem(struct LinkedNode** head, int n) {
    struct LinkedNode* temp;
    temp = *head;
    while (temp != NULL) {
        if (temp->val == n) {
            return 1;
        }
        temp = temp->next;
    }
    return 0;
}

void add(struct LinkedNode** head, struct LinkedNode** tail, int num)
{
    struct LinkedNode* node;
    node = (struct LinkedNode*)malloc(sizeof(struct LinkedNode));
    node->val = num;
    node->next = *head;
    if (*head == NULL) {
        *tail = node;
    }
    *head = node;
}

void addToEnd(struct LinkedNode** head, struct LinkedNode** tail, int
num) {
    if (*head == NULL || *tail == NULL) {
        add(head, tail, num);
        return;
    }
    struct LinkedNode* node;
    node = (struct LinkedNode*)malloc(sizeof(struct LinkedNode));
    node->val = num;
    node->next = NULL;
    (*tail)->next = node;
    *tail = node;
}

void searchAdd(struct LinkedNode** head, struct LinkedNode** tail, int
s, int n, int order) {
    struct LinkedNode* temp;
    temp = *head;

    //      order = 1 (after) 0 (before)
    if (temp != NULL && temp->val == s) {
        struct LinkedNode* node;
        node = (struct LinkedNode*)malloc(sizeof(struct LinkedNode));
        node->val = n;

        if (order == 1) {
            node->next = temp->next;
            temp->next = node;
        }
        else {
            node->next = *head;
            *head = node;
        }
        return;
    }

    while (temp != NULL) {
        if (temp->next->val == s) {
            struct LinkedNode* node;
            node = (struct LinkedNode*)malloc(sizeof(struct
LinkedNode));
            node->val = n;

            if (order == 1) {
                temp = temp->next;
            }
            node->next = temp->next;
            temp->next = node;
            if (node->next == NULL) {
                *tail = node;
            }
            return;
        }
        temp = temp->next;
    }
}

void printLinkedList(struct LinkedNode** head) {
    struct LinkedNode* temp;
    temp = *head;

    printf("\n[ ");
    while (temp != NULL) {
        printf("%d ", temp->val);
        temp = temp->next;
    }
    printf("]");
}

void deleteElem(struct LinkedNode** head, struct LinkedNode** tail,
int n) {
    if (searchElem(head, n) == 1) {
        struct LinkedNode* temp;
```

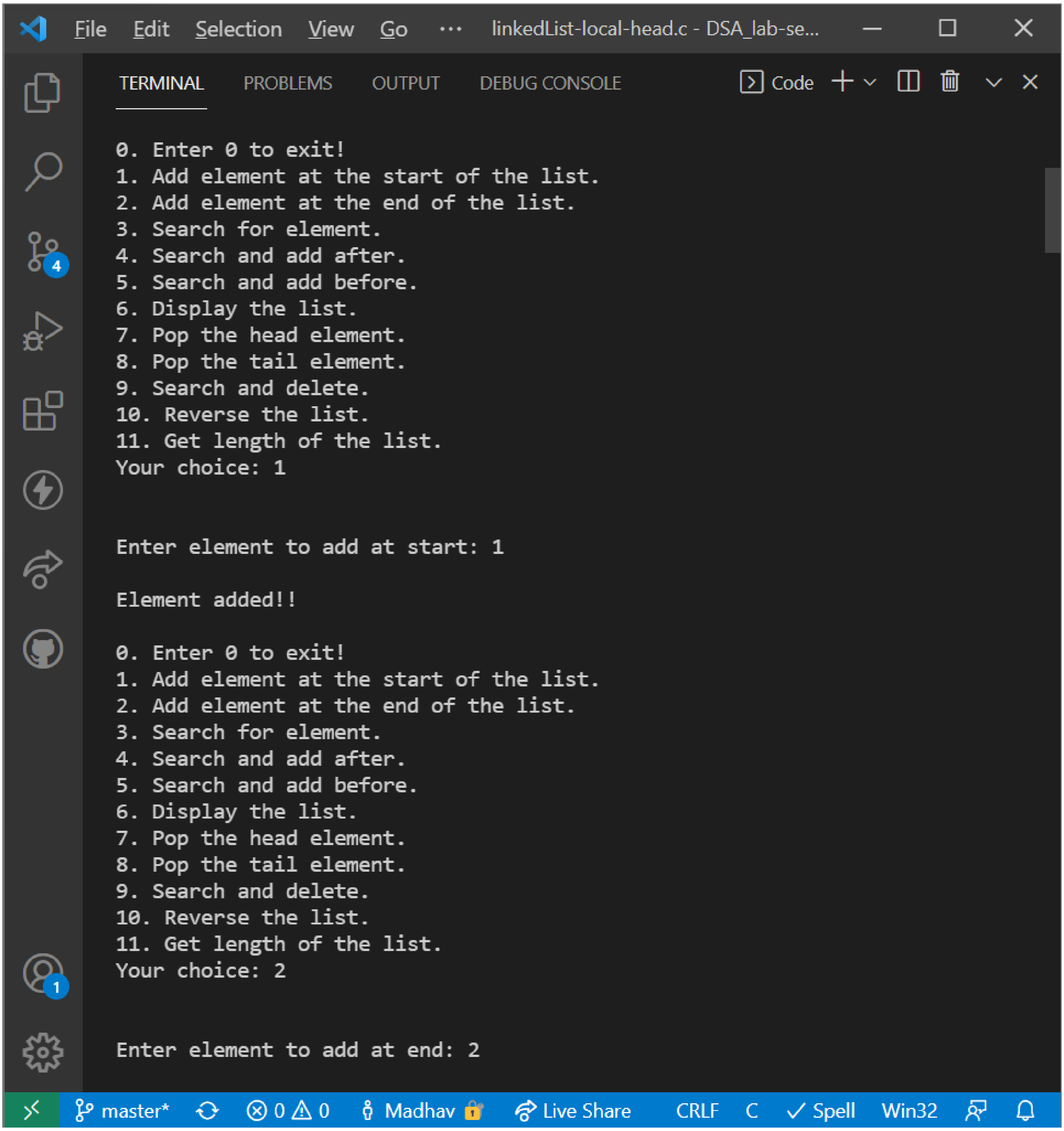


```
        temp = *head;

        if (temp->val == n) {
            *head = (*head)->next;
            return;
        }

        while (temp != NULL) {
            if (temp->next->val == n) {
                if (temp->next == *tail) {
                    *tail = temp;
                }
                temp->next = temp->next->next;
                return;
            }
            temp = temp->next;
        }
    }
}

void pop(struct LinkedNode** head) {
    struct LinkedNode* temp;
    temp = *head;

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

void reverseList(struct LinkedNode** head, struct LinkedNode** tail) {
    if (*head == NULL || *tail == NULL) return;

    struct LinkedNode* prev = NULL, * curr = NULL;

    while (*head != NULL) {
        prev = curr;
        curr = *head;
        *head = (*head)->next;
        curr->next = prev;
    }
    *head = curr;
}

int getLen(struct LinkedNode** head) {
    int count = 0;
    if (*head == NULL) return count;

    struct LinkedNode* temp = *head;

    while (temp != NULL) {
        ++count;
        temp = temp->next;
    }
    return count;
}

int main() {

    struct LinkedNode** head = (struct
LinkedNode**)malloc(sizeof(struct LinkedNode*));
    struct LinkedNode** tail = (struct
LinkedNode**)malloc(sizeof(struct LinkedNode*));
    *head = NULL;
    *tail = NULL;

    int choice = 0;
    do {
        printf("\n0. Enter 0 to exit!");
        printf("\n1. Add element at the start of the list.");
        printf("\n2. Add element at the end of the list.");
        printf("\n3. Search for element.");
        printf("\n4. Search and add after.");
        printf("\n5. Search and add before.");
        printf("\n6. Display the list.");
        printf("\n7. Pop the head element.");
        printf("\n8. Pop the tail element.");
        printf("\n9. Search and delete.");
        printf("\n10. Reverse the list.");
        printf("\n11. Get length of the list.");
        printf("\nYour choice: ");
        scanf("%d", &choice);

        int inp, src;
        switch (choice) {
        case 0:
            printf("\n\nExit...!\n\n");
            break;
        case 1:
            printf("\n\nEnter element to add at start: ");
            scanf("%d", &inp);
            add(head, tail, inp);
            printf("\nElement added!!\n");
            break;
        case 2:
            printf("\n\nEnter element to add at end: ");
            scanf("%d", &inp);
            addToEnd(head, tail, inp);
            printf("\nElement added!!\n");
            break;
        case 3:
            printf("\n\nEnter element to search: ");
            scanf("%d", &inp);
            printf("\nIs element %d present: %d\n", inp,
searchElem(head, inp));
            break;
        case 4:
            printf("\n\nEnter element to search and add after: ");
            scanf("%d", &src);
            printf("\n\nEnter element to add: ");
            scanf("%d", &inp);
            searchAdd(head, tail, src, inp, 1);
            printf("\n");
            break;
        case 5:
            printf("\n\nEnter element to search and add before: ");
            scanf("%d", &src);
            printf("\n\nEnter element to add: ");
            scanf("%d", &inp);
            searchAdd(head, tail, src, inp, 0);
```

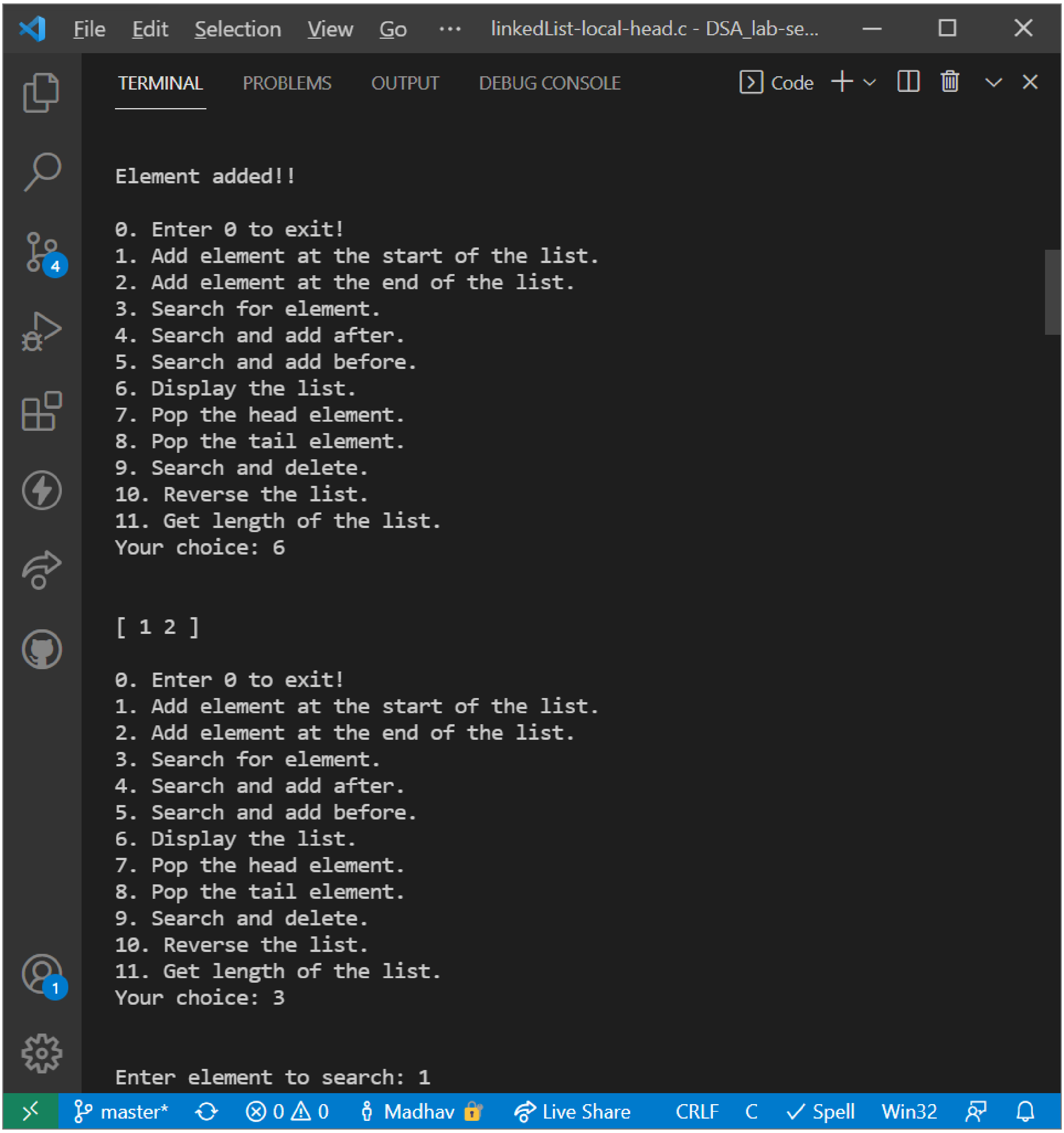


```
            printf("\n");
            break;
        case 6:
            printf("\n");
            printLinkedList(head);
            printf("\n");
            break;
        case 7:
            pop(head);
            printf("\n\nElement popped!!\n");
            break;
        case 8:
            deleteElem(head, tail, (*tail)->val);
            printf("\n\nElement popped!!\n");
            break;
        case 9:
            printf("\n\nEnter element to delete: ");
            scanf("%d", &inp);
            deleteElem(head, tail, inp);
            printf("\n");
            break;
        case 10:
            reverseList(head, tail);
            printf("\n");
            break;
        case 11:
            printf("\n\nLength of the list: %d", getLen(head));
            printf("\n");
            break;
        default:
            printf("\nERROR: Invalid choice!!!");
            break;
        }
    } while (choice != 0);

    return 0;
}
```

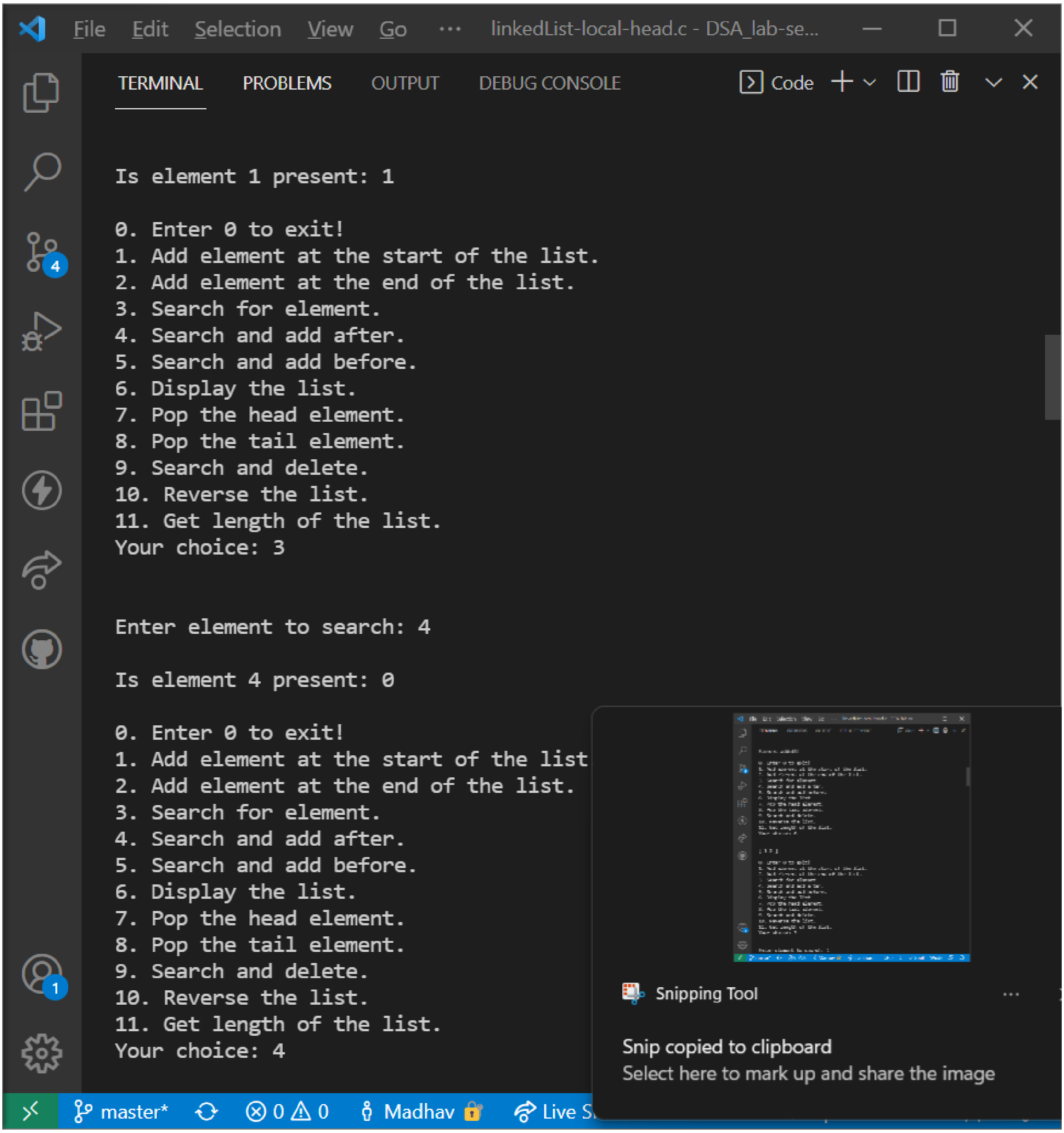
Output:



```
0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 1


Enter element to add at start: 1

Element added!!

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 2


Enter element to add at end: 2
```

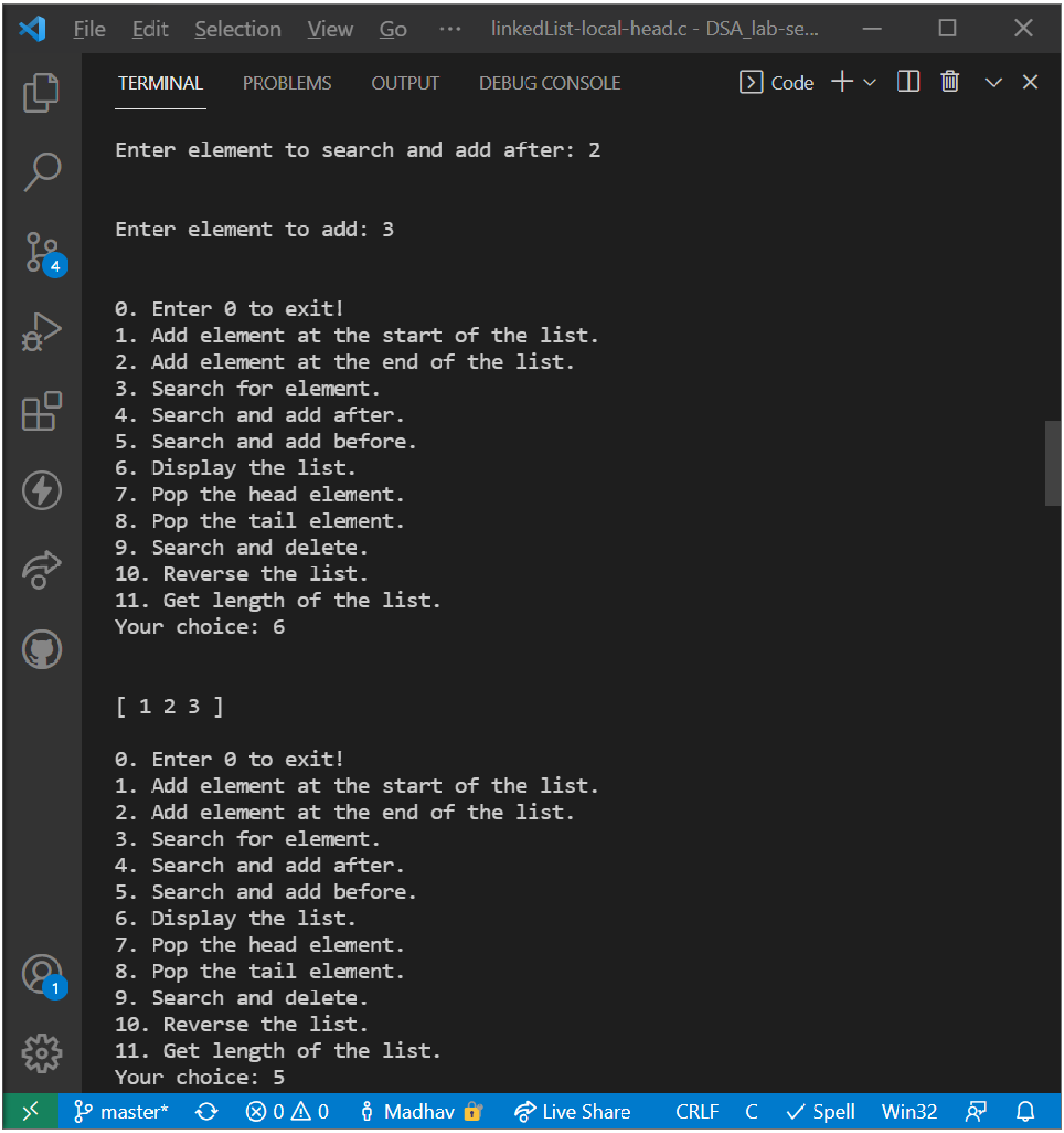


```
Element added!!

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 6


[ 1 2 ]

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 3


Enter element to search: 1
```



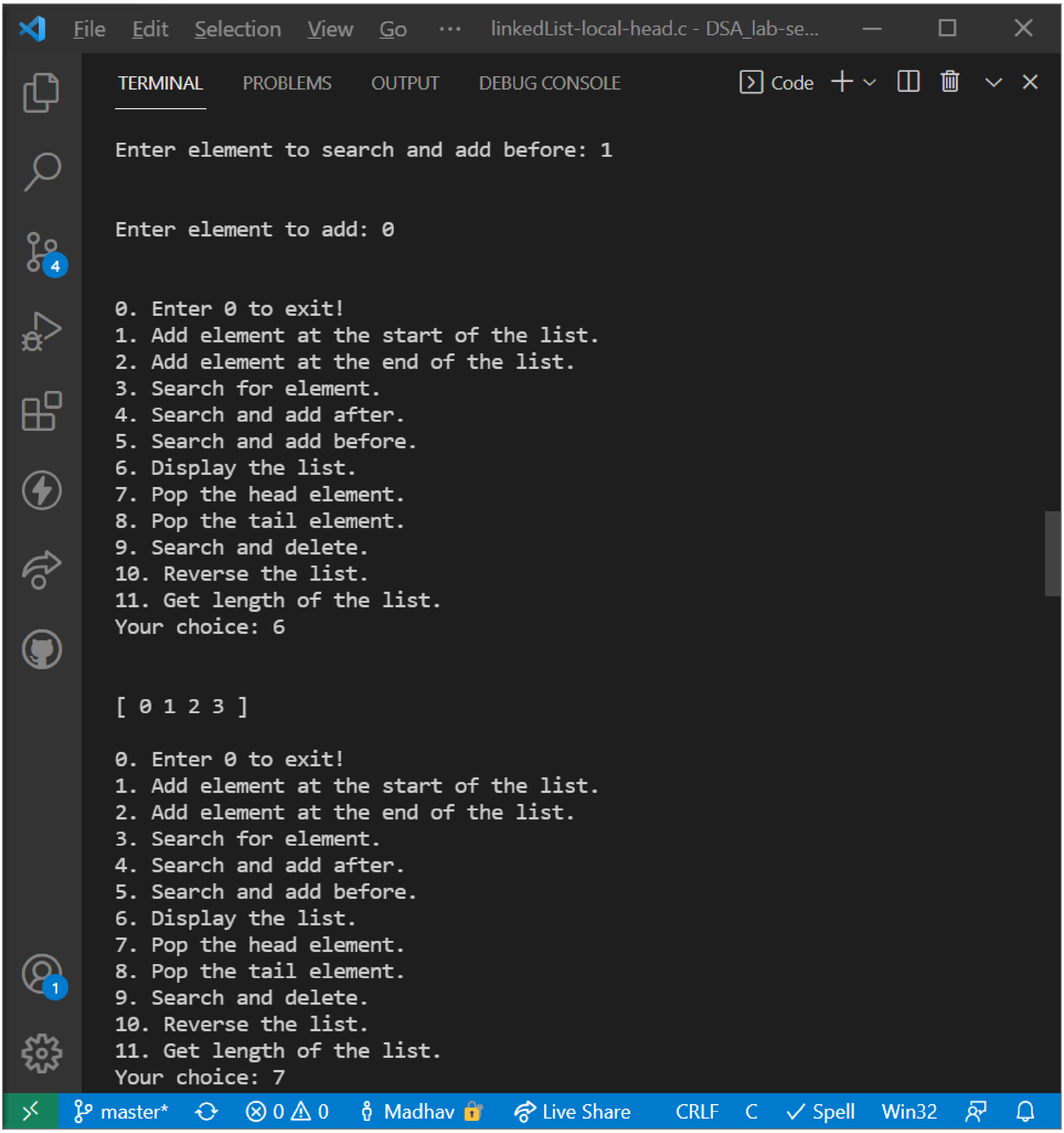
```
Is element 1 present: 1

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 3


Enter element to search: 4

Is element 4 present: 0

0. Enter 0 to exit!
1. Add element at the start of the list
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 4
```

Snip copied to clipboard
Select here to mark up and share the image

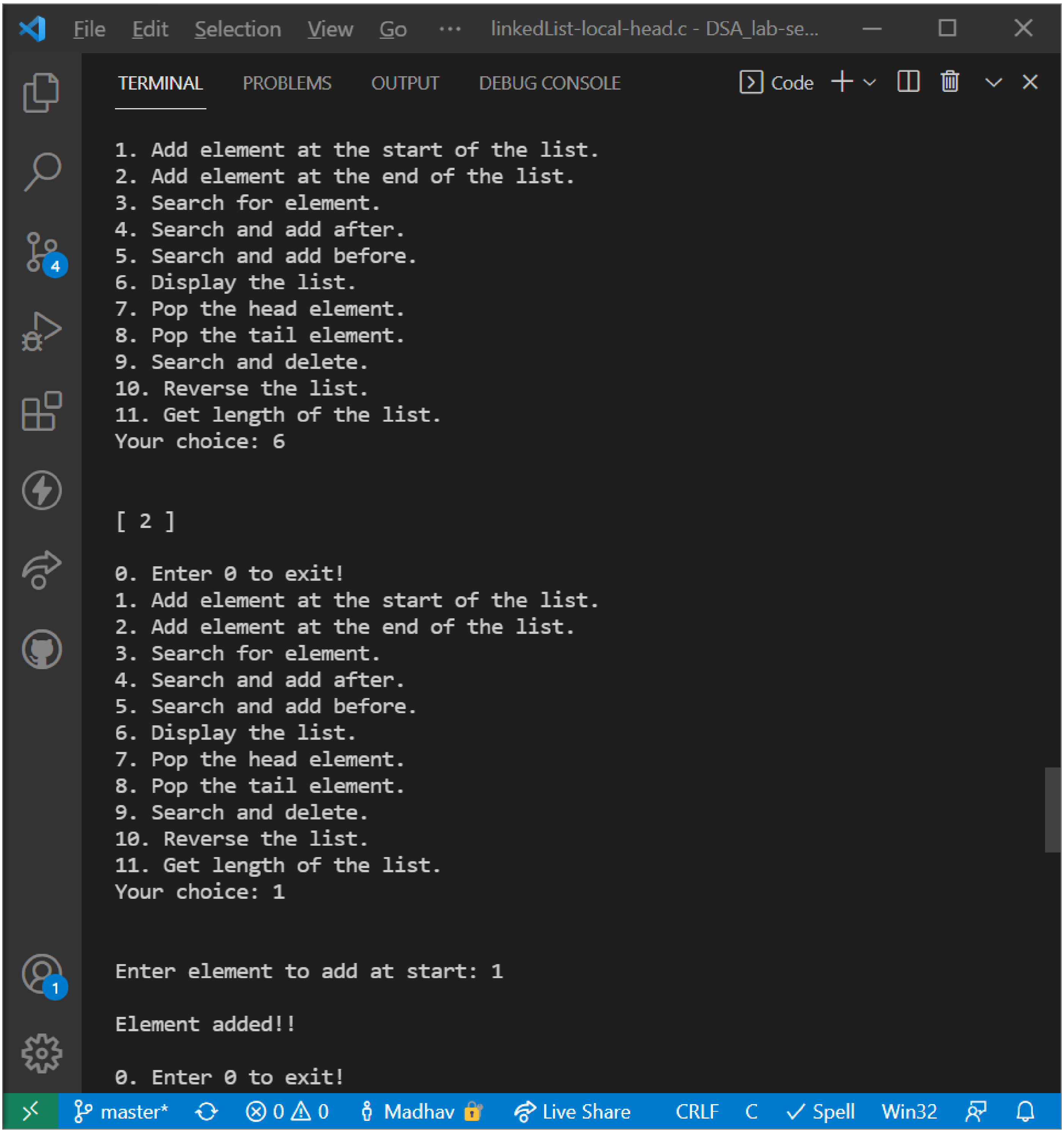


File Edit Selection View Go ··· linkedList-local-head.c - DSA_lab-se...
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
Code

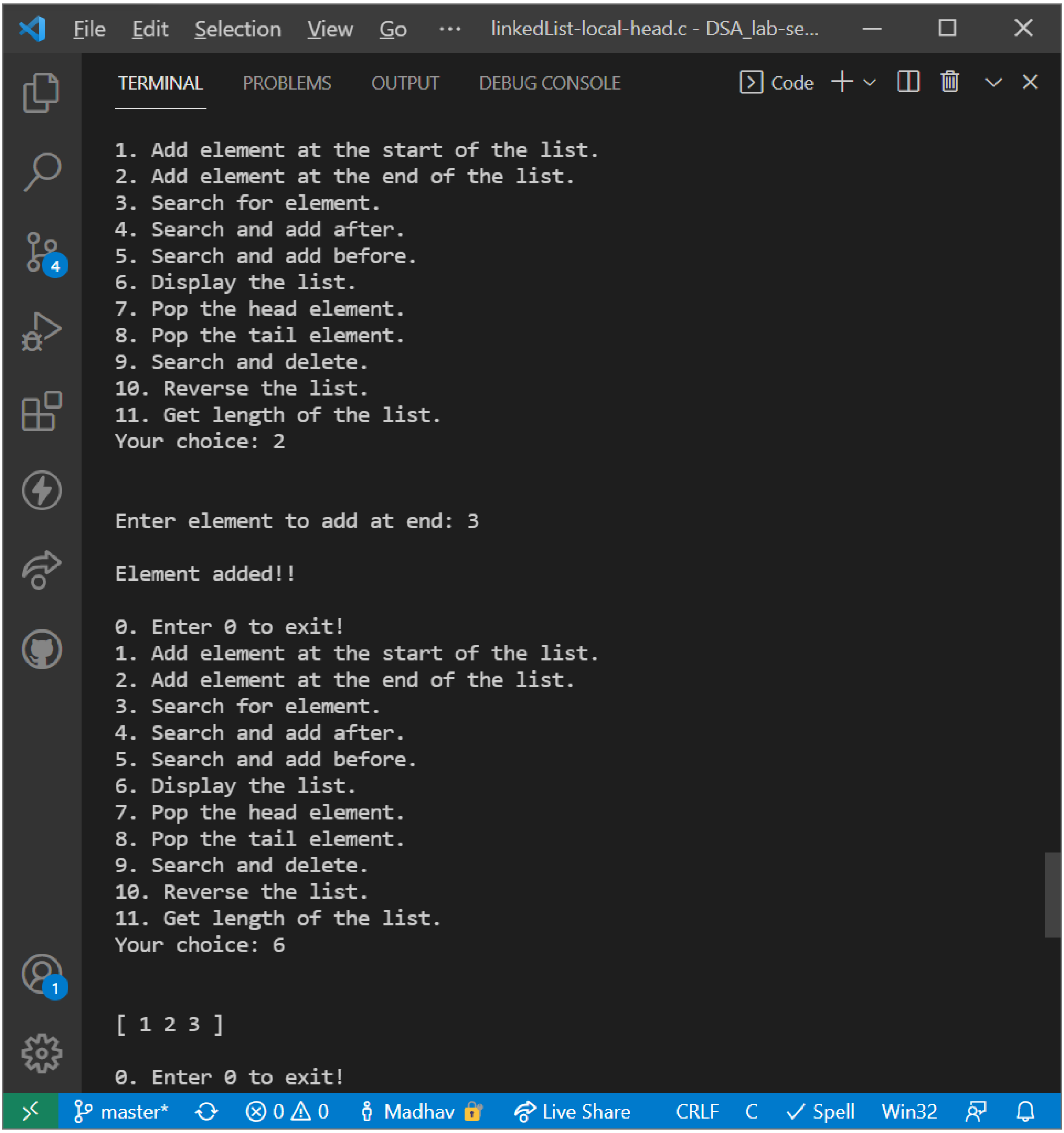
```
Enter element to search and add after: 2


Enter element to add: 3


0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 6


[ 1 2 3 ]

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 5
```



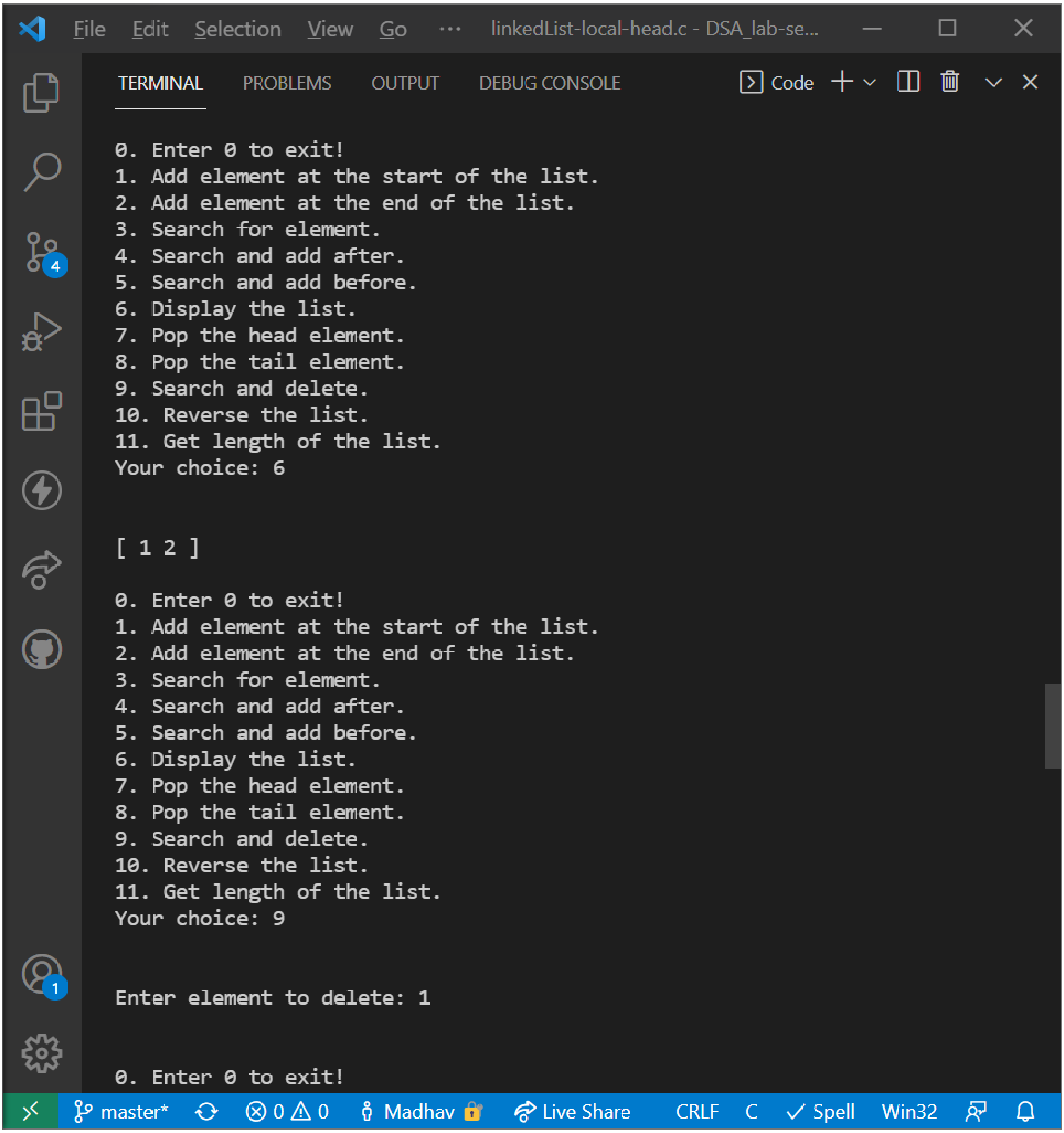
```
Enter element to search and add before: 1


Enter element to add: 0


0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 6


[ 0 1 2 3 ]

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 7
```



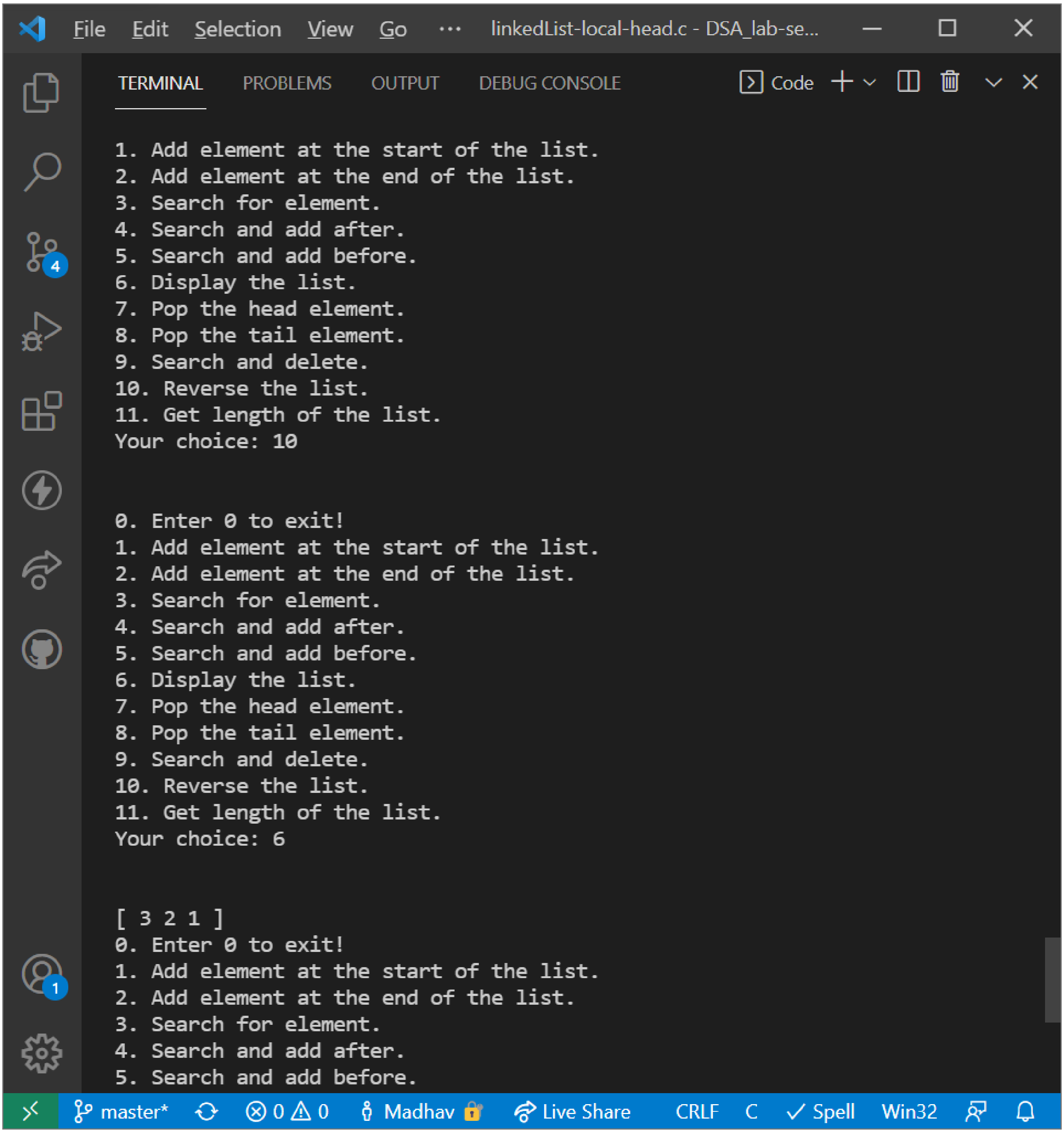
```
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 6


[ 2 ]

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 1


Enter element to add at start: 1

Element added!!

0. Enter 0 to exit!
```



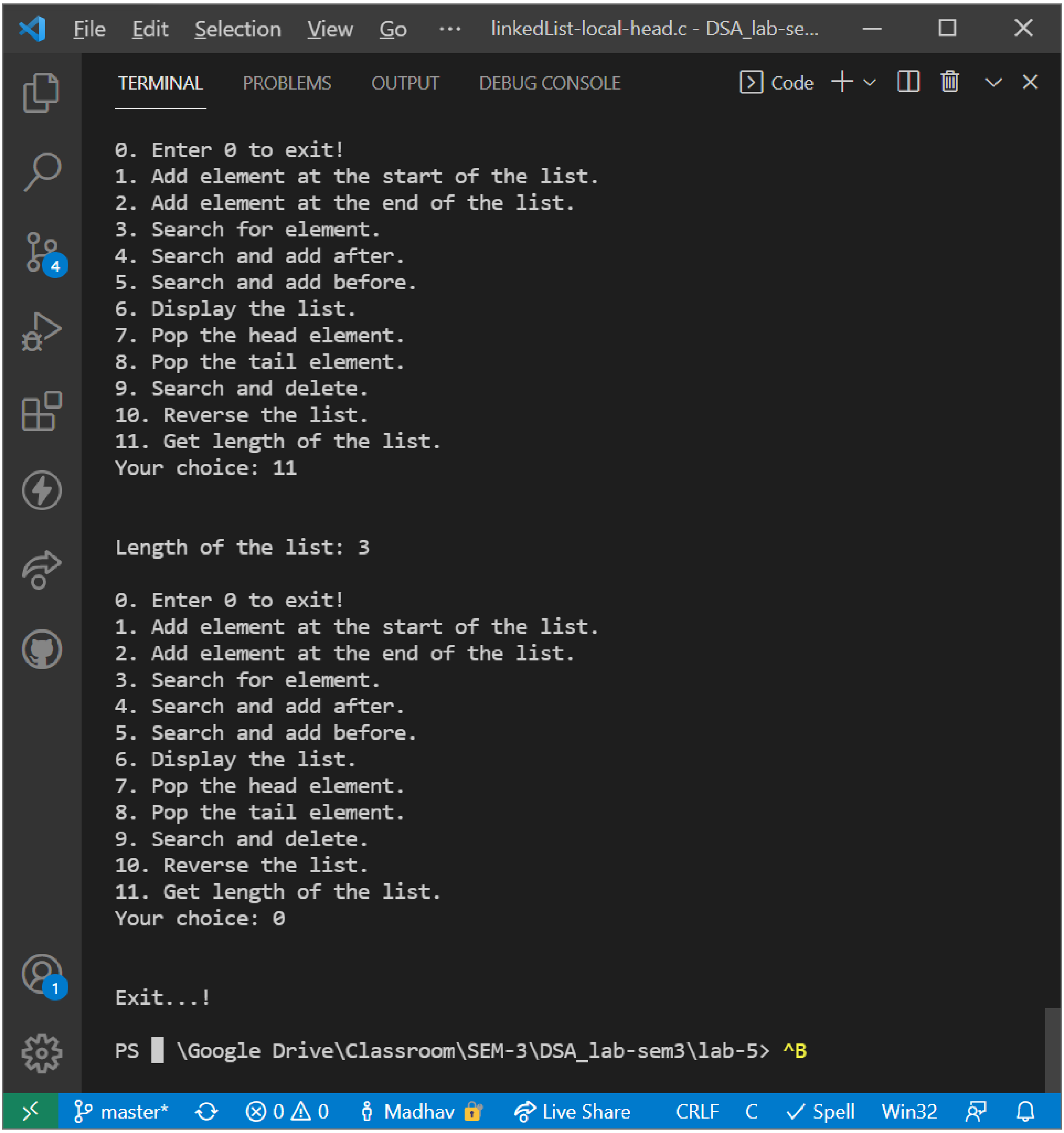
```
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 2


Enter element to add at end: 3

Element added!!

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 6


[ 1 2 3 ]

0. Enter 0 to exit!
```



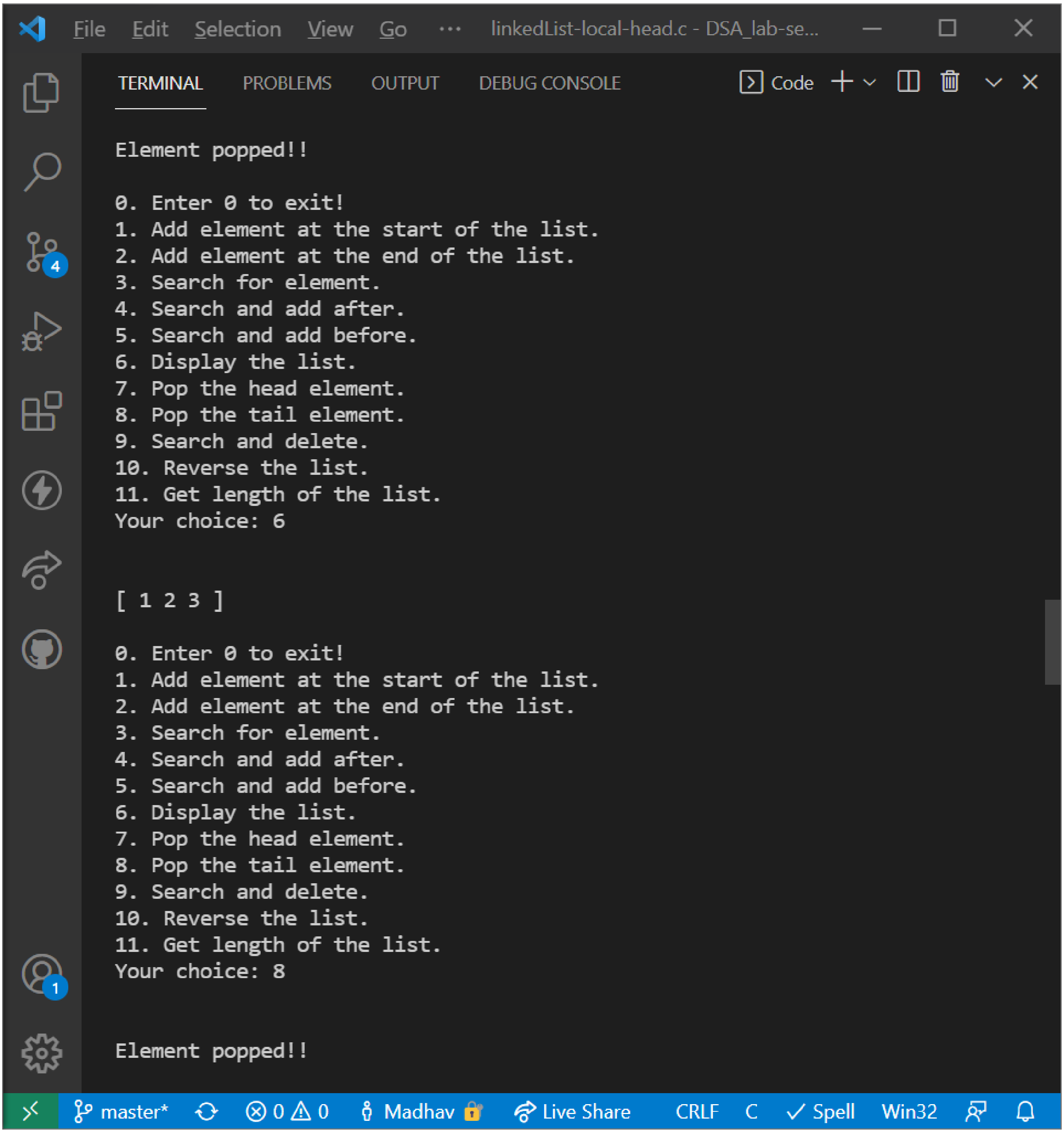
```
0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 6


[ 1 2 ]

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 9


Enter element to delete: 1


0. Enter 0 to exit!
```



```
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 10


0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 6


[ 3 2 1 ]
0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
```



```
0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 11


Length of the list: 3

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 0


Exit...!

PS [redacted] \Google Drive\Classroom\SEM-3\DSA_lab-sem3\lab-5> ^B
```



```
Element popped!!

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 6


[ 1 2 3 ]

0. Enter 0 to exit!
1. Add element at the start of the list.
2. Add element at the end of the list.
3. Search for element.
4. Search and add after.
5. Search and add before.
6. Display the list.
7. Pop the head element.
8. Pop the tail element.
9. Search and delete.
10. Reverse the list.
11. Get length of the list.
Your choice: 8


Element popped!!
```