DSA LAB

Assignment 8

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

Doubly Linked List

AIM: To study doubly linked linear list and implement various operations on it – Insert, Delete, Reverse, Sorting, Locate.

Create a self-referential structure, node to represent a node of a doubly linked linear list.
Implement the routines to

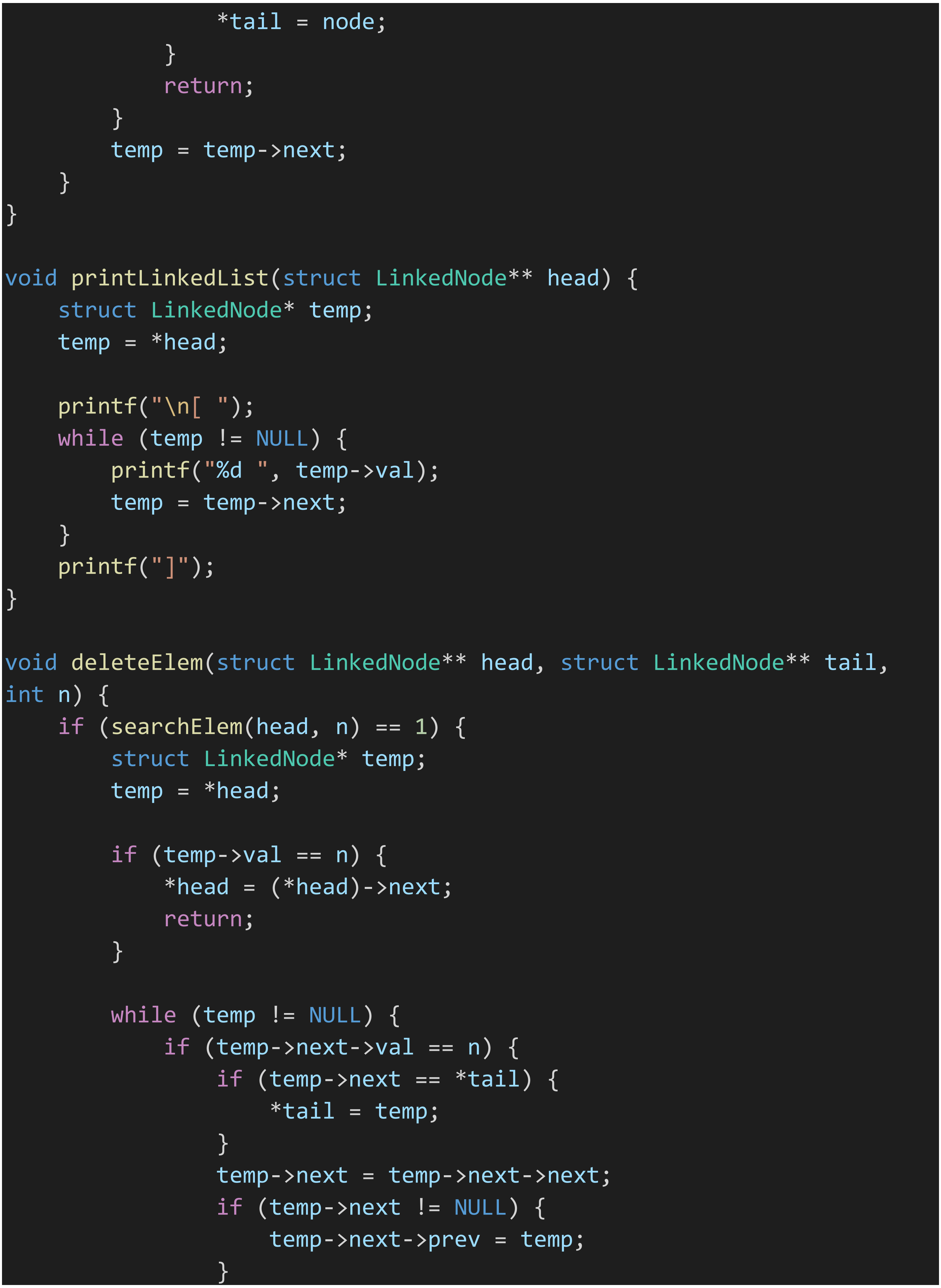
(1) Find length of the list
(2) Create a list
(3) Insert an element
  - at the beginning
  - at the end
  - at a specified position in the list
(4) Delete an element
  - the beginning
  - end
  - a specified position at the list.
(5) Reverse the list.
(6) Search the list.
(7) Sort the list.

Create a menu-driven program to test these routines.

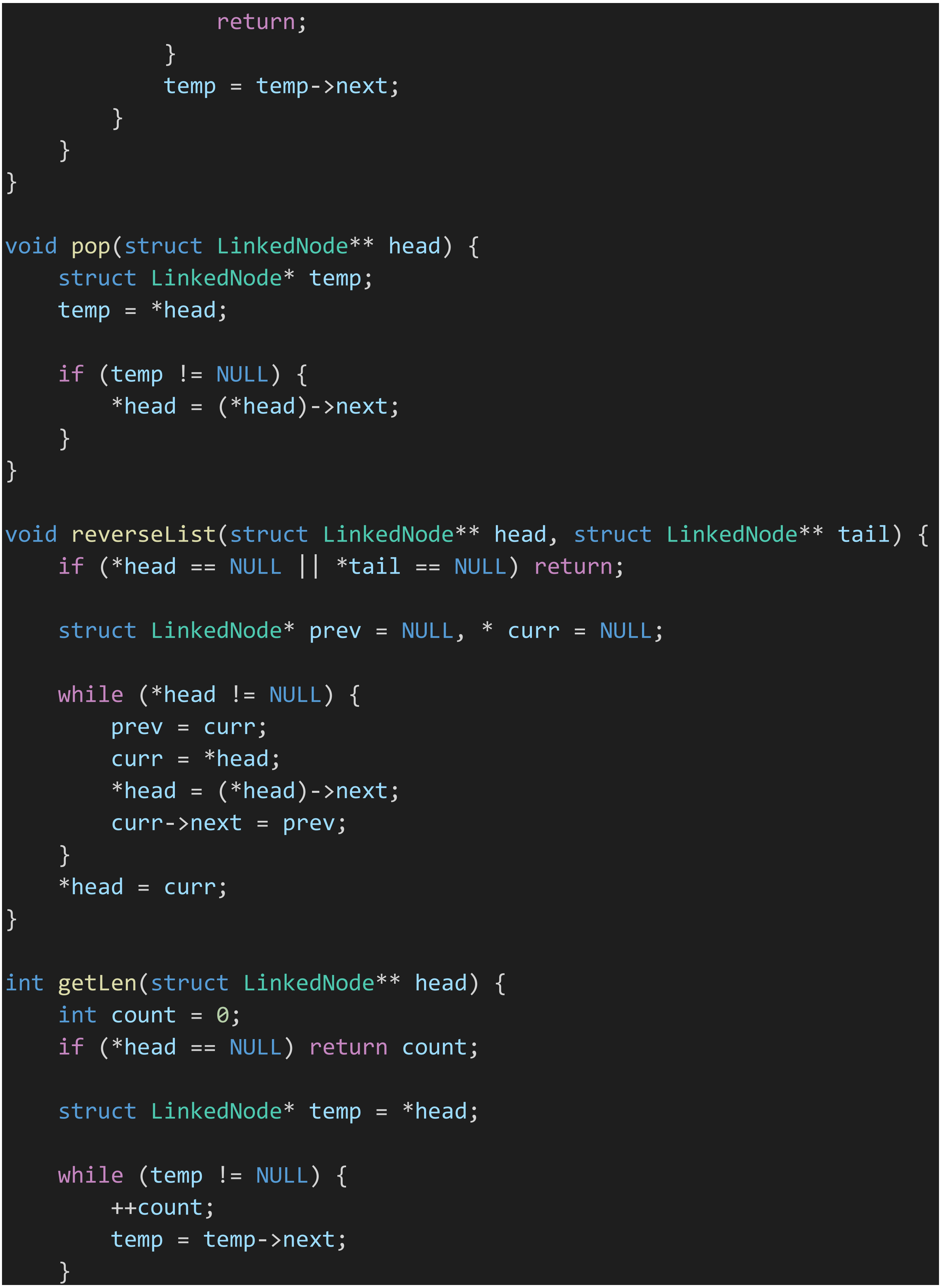
Code:

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

struct LinkedNode {
    int val;
    struct LinkedNode* next;
    struct LinkedNode* prev;
};
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;
    node->prev = NULL;

    if (*head == NULL) {
        *tail = node;
    }
    else {
        (*head)->prev = 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;
    node->prev = *tail;
    (*tail)->next = node;
    *tail = node;
}
```

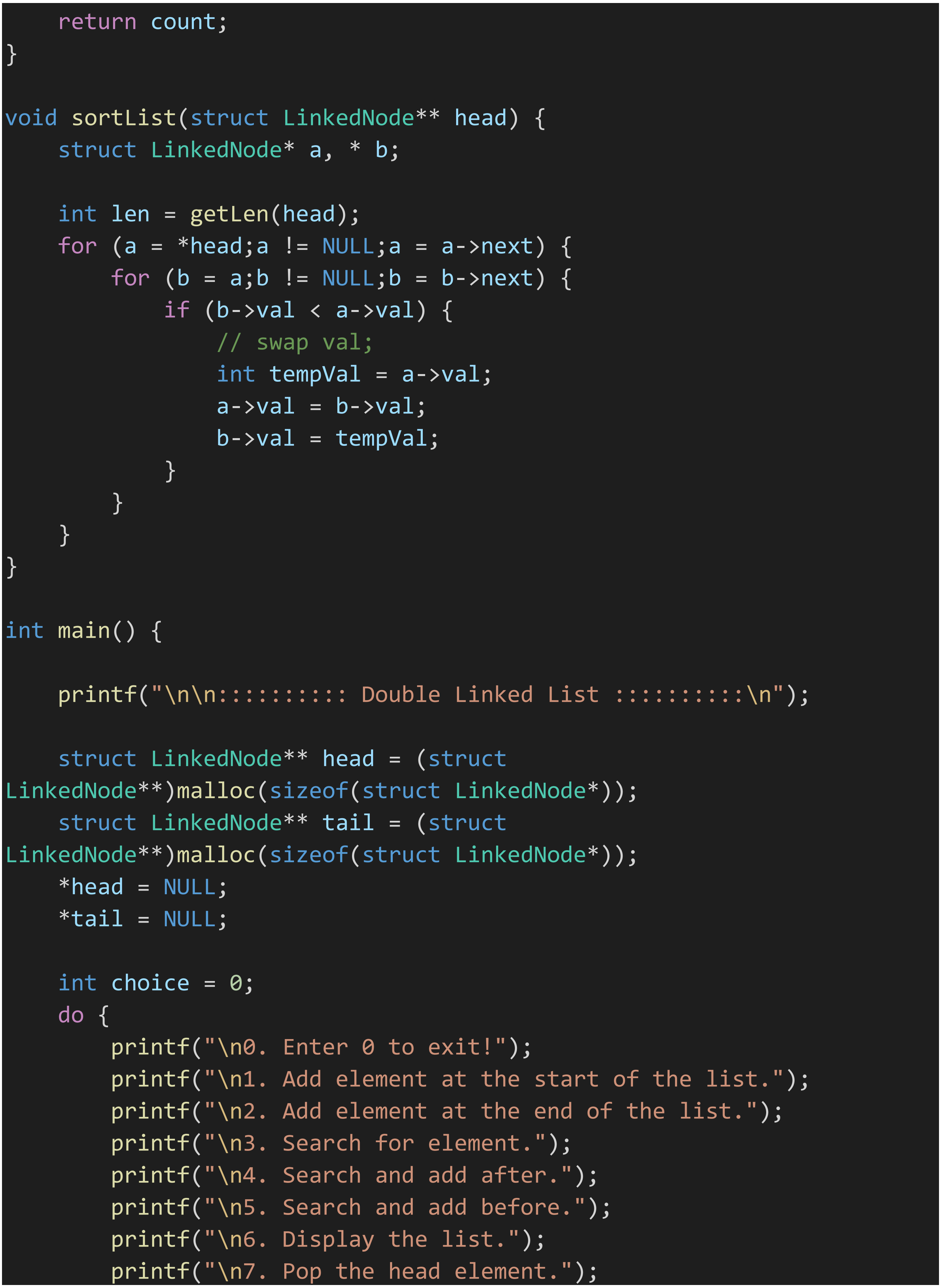
```
void searchAdd(struct LinkedNode** head, struct LinkedNode** tail, int
s, int n, int order) {
    struct LinkedNode* temp;
    temp = *head;

    //     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;
            node->prev = temp;
            temp->next = node;
        }
        else {
            node->next = *head;
            node->prev = NULL;
            temp->prev = node;
            *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;
            node->prev = temp;
            temp->next = node;
            temp->next->prev = node;
            if (node->next == NULL) {
```



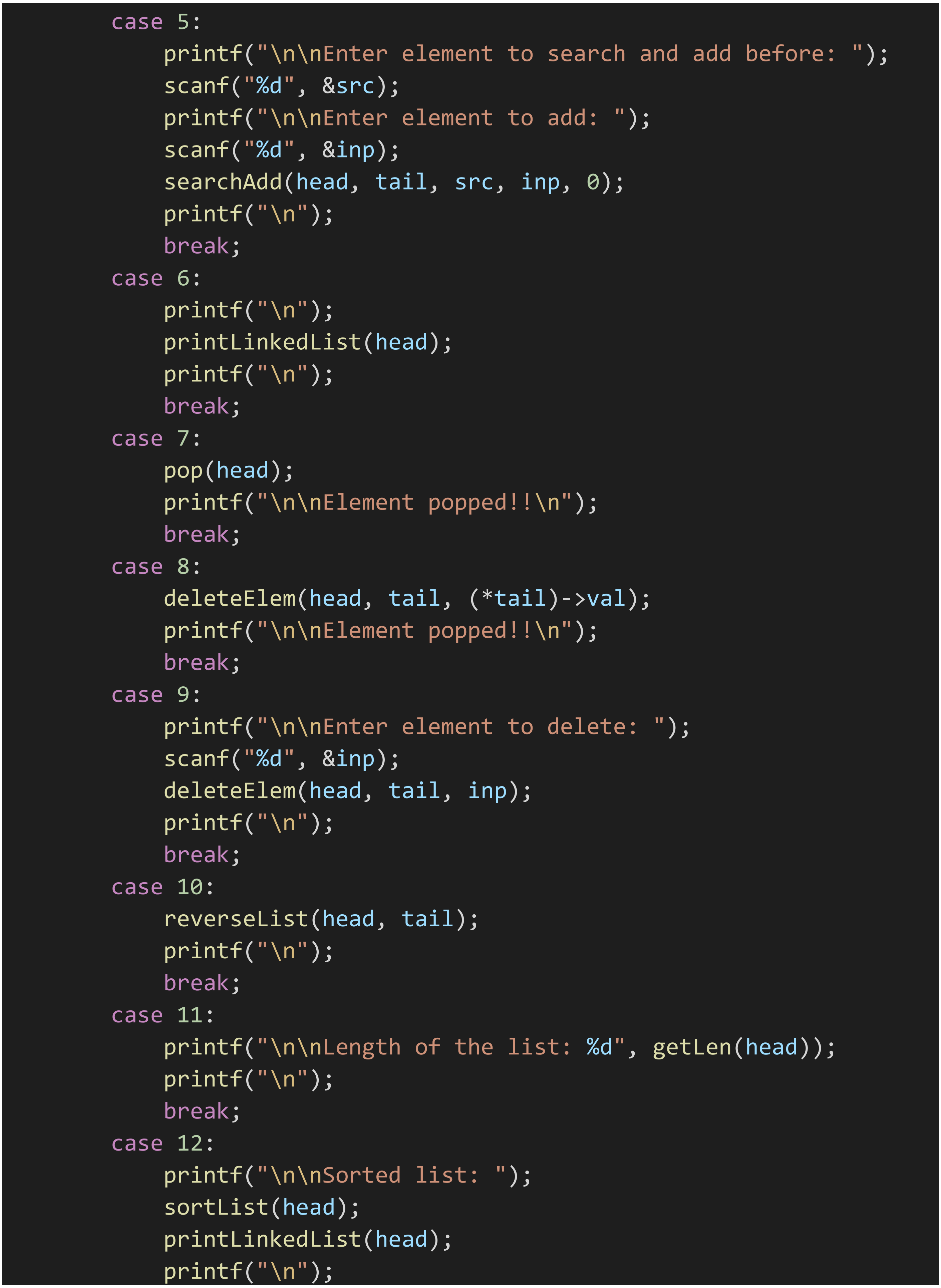
```
                *tail = node;
            }
            return;
        }
        temp = temp->next;
    }
}

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;
                if (temp->next != NULL) {
                    temp->next->prev = temp;
                }
```



```
                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;
    }
```

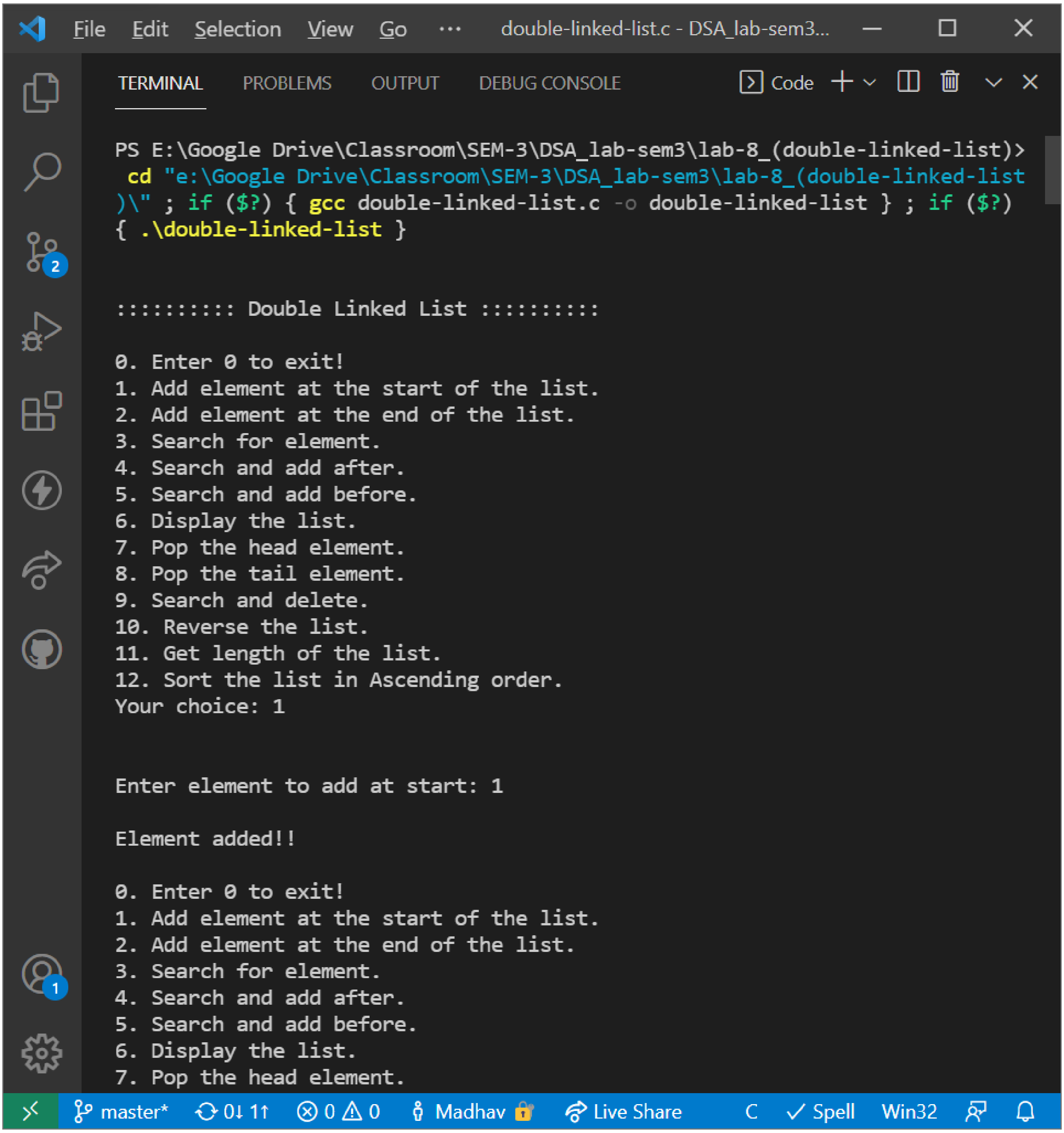


```
    return count;
}

void sortList(struct LinkedNode** head) {
    struct LinkedNode* a, * b;

    int len = getLen(head);
    for (a = *head;a != NULL;a = a->next) {
        for (b = a;b != NULL;b = b->next) {
            if (b->val < a->val) {
                // swap val;
                int tempVal = a->val;
                a->val = b->val;
                b->val = tempVal;
            }
        }
    }
}

int main() {

    printf("\n\n:::::::::: Double Linked List ::::::::::\n");

    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("\n12. Sort the list in Ascending order.");

        printf("\nYour choice: ");
        scanf("%d", &choice);

        int inp, src;
        switch (choice) {
        case 0:
            printf("\n\nExit...!\n\n- by Madhav Jha\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;
```

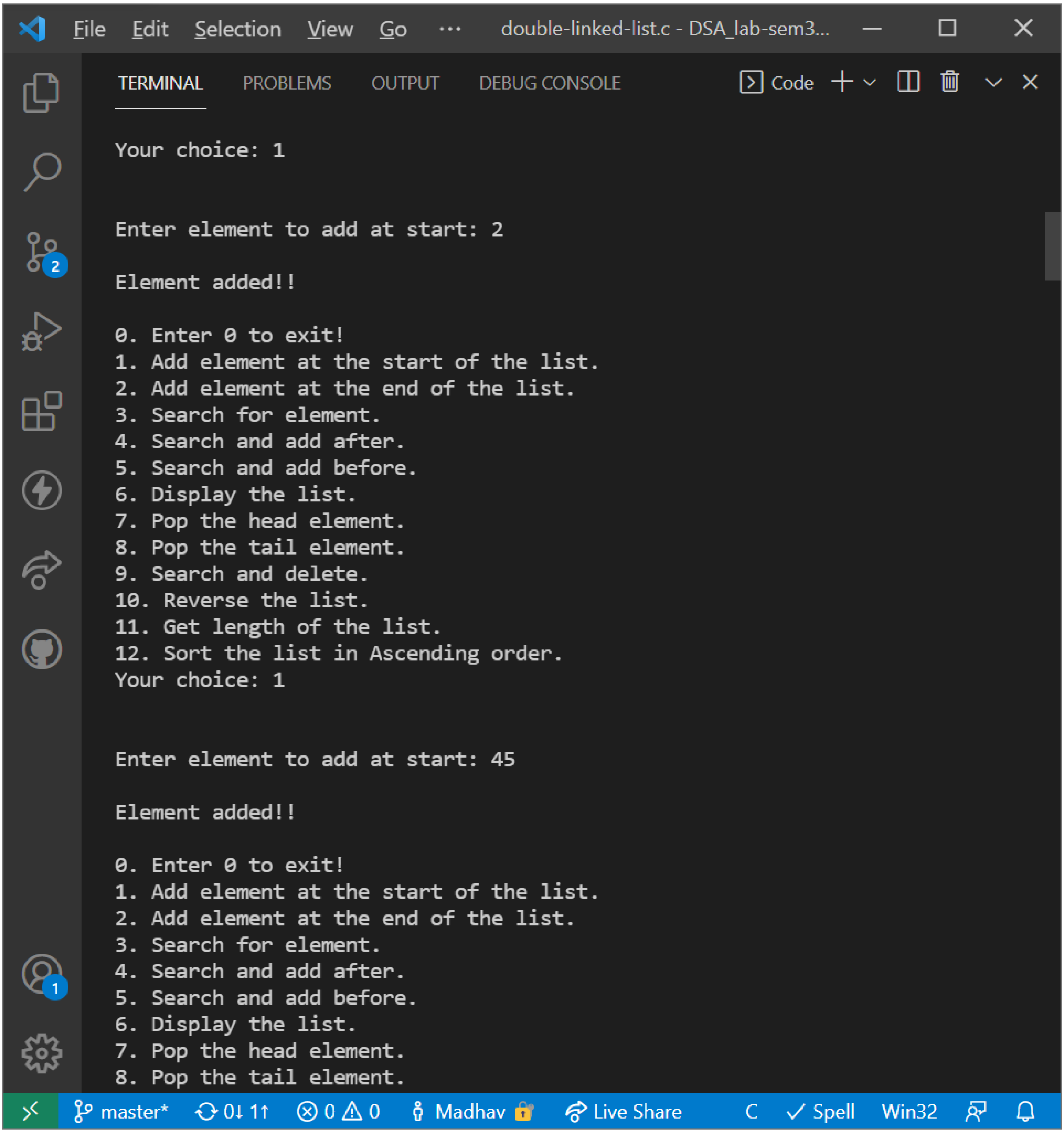


```
        case 5:
            printf("\n\nEnter element to search and add before: ");
            scanf("%d", &src);
            printf("\n\nEnter element to add: ");
            scanf("%d", &inp);
            searchAdd(head, tail, src, inp, 0);
            printf("\n");
            break;
        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;
        case 12:
            printf("\n\nSorted list: ");
            sortList(head);
            printLinkedList(head);
            printf("\n");
            break;
        default:
            printf("\nERROR: Invalid choice!!!");
            break;
        }
    } while (choice != 0);

    return 0;
}
```

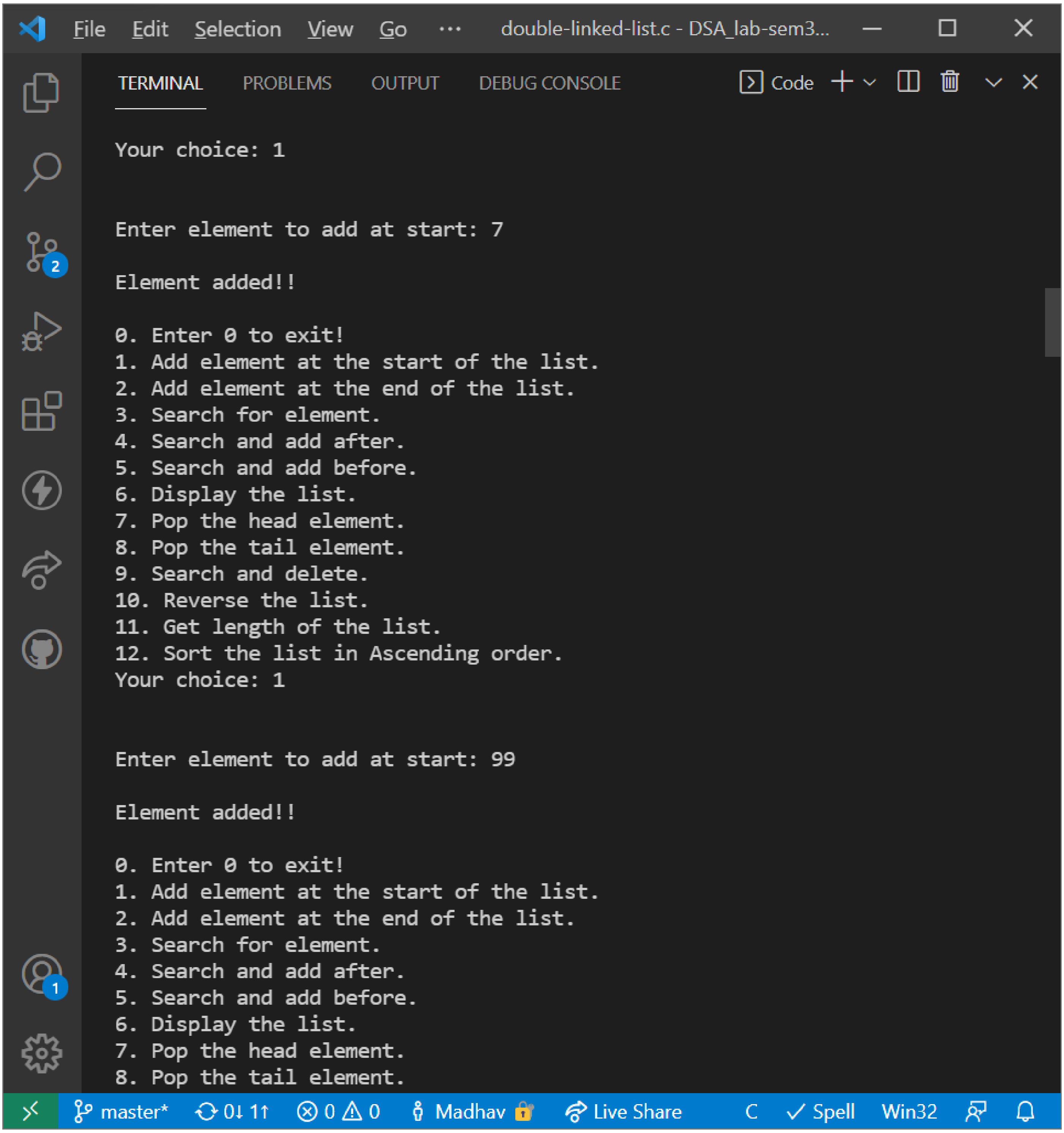
Output:



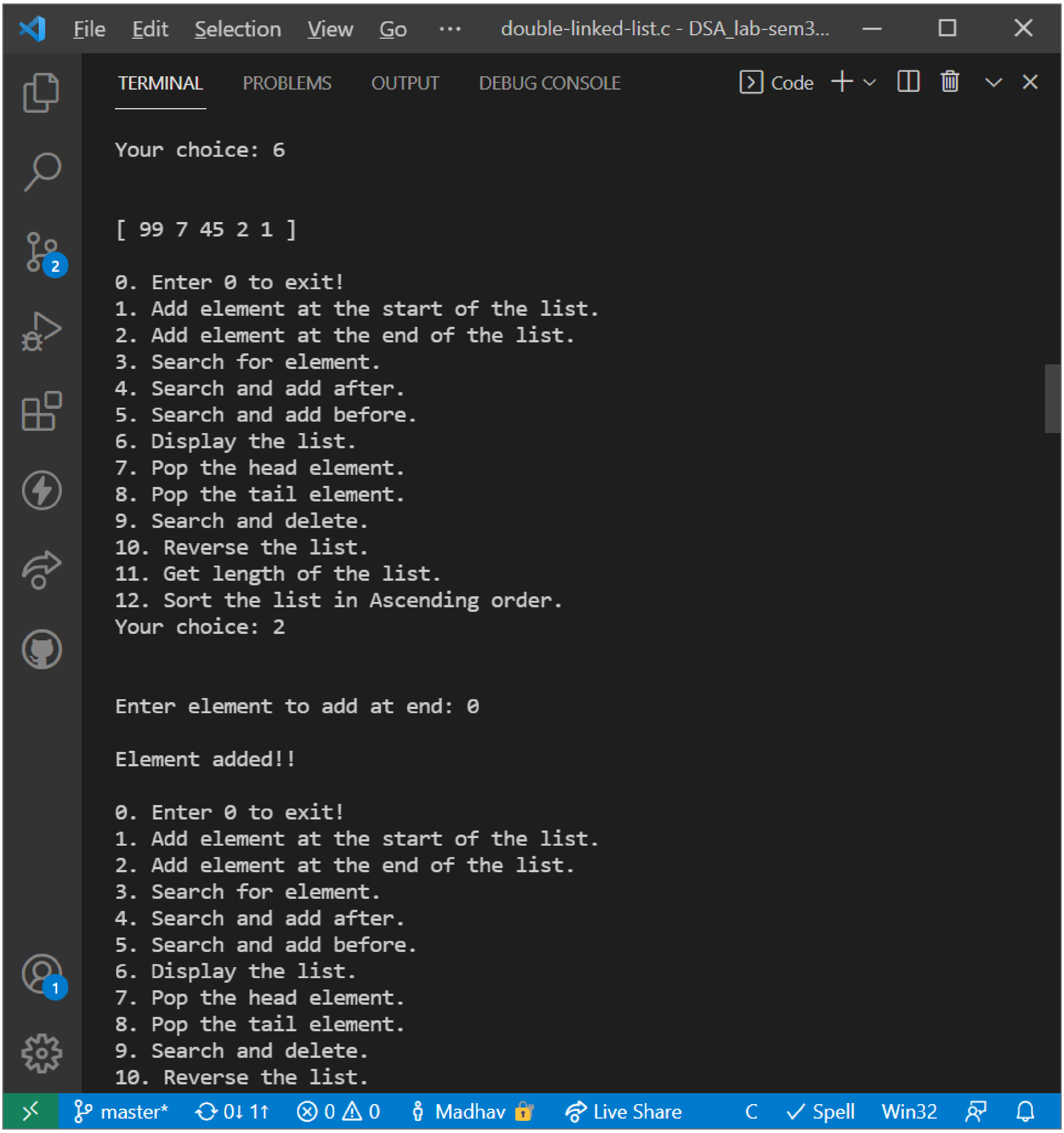
```
PS E:\Google Drive\Classroom\SEM-3\DSA_lab-sem3\lab-8_(double-linked-list)>
 cd "e:\Google Drive\Classroom\SEM-3\DSA_lab-sem3\lab-8_(double-linked-list
)\" ; if ($?) { gcc double-linked-list.c -o double-linked-list } ; if ($?)
{ .\double-linked-list }


:::::::::: Double Linked List ::::::::::

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.
12. Sort the list in Ascending order.
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.
```



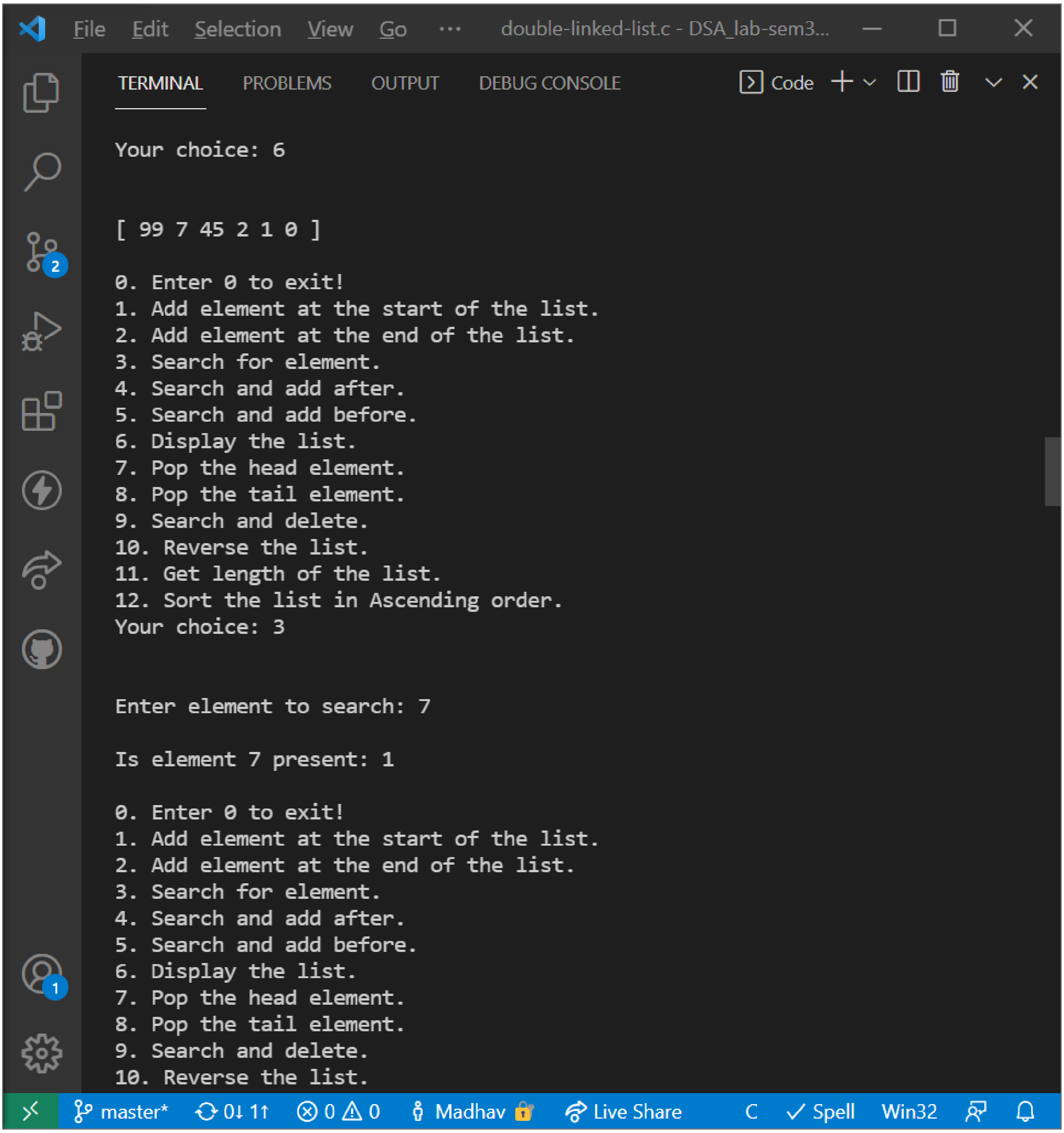
```
Your choice: 1


Enter element to add at start: 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.
12. Sort the list in Ascending order.
Your choice: 1


Enter element to add at start: 45

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.
```



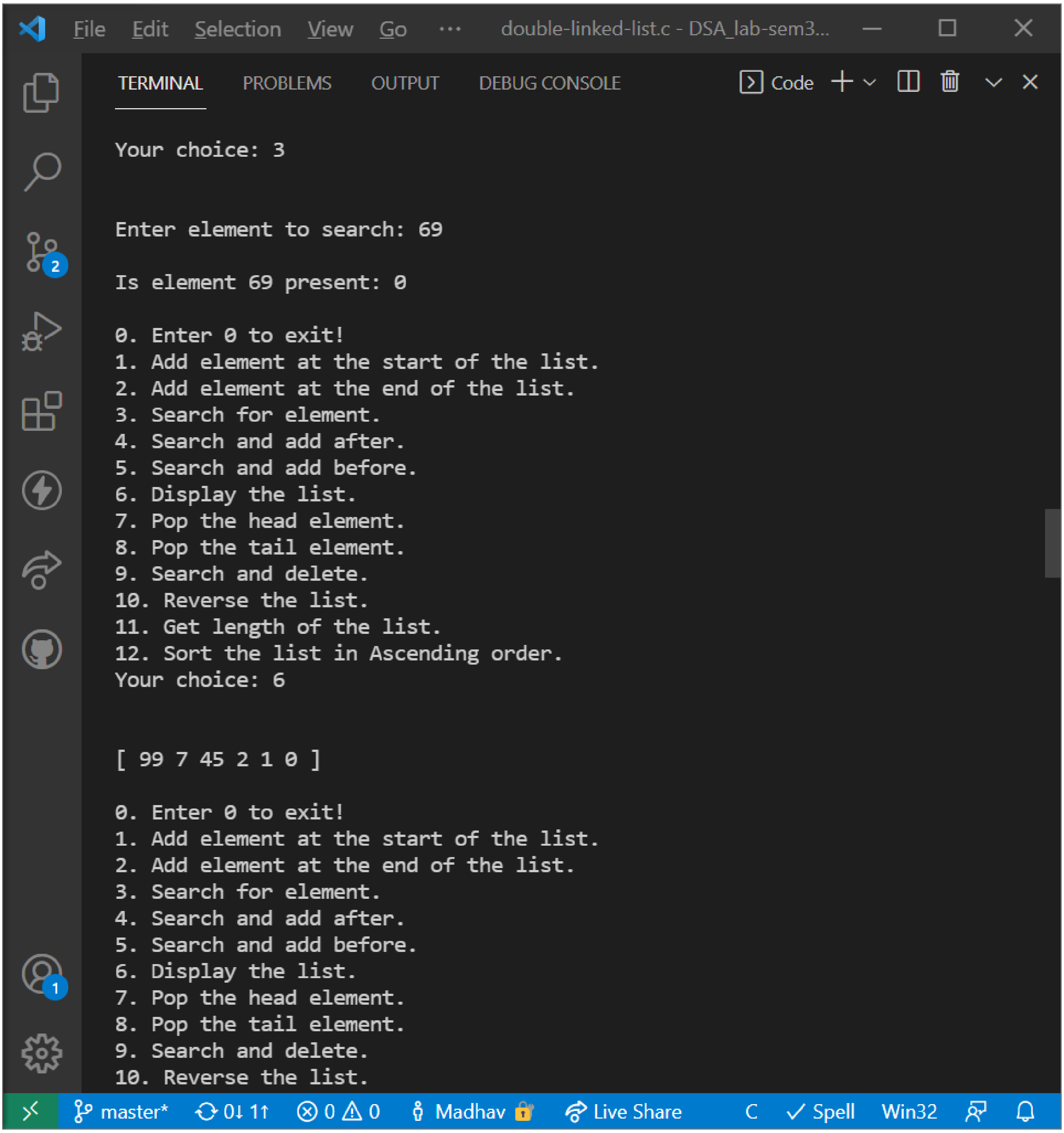
```
Your choice: 1


Enter element to add at start: 7

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.
12. Sort the list in Ascending order.
Your choice: 1


Enter element to add at start: 99

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.
```



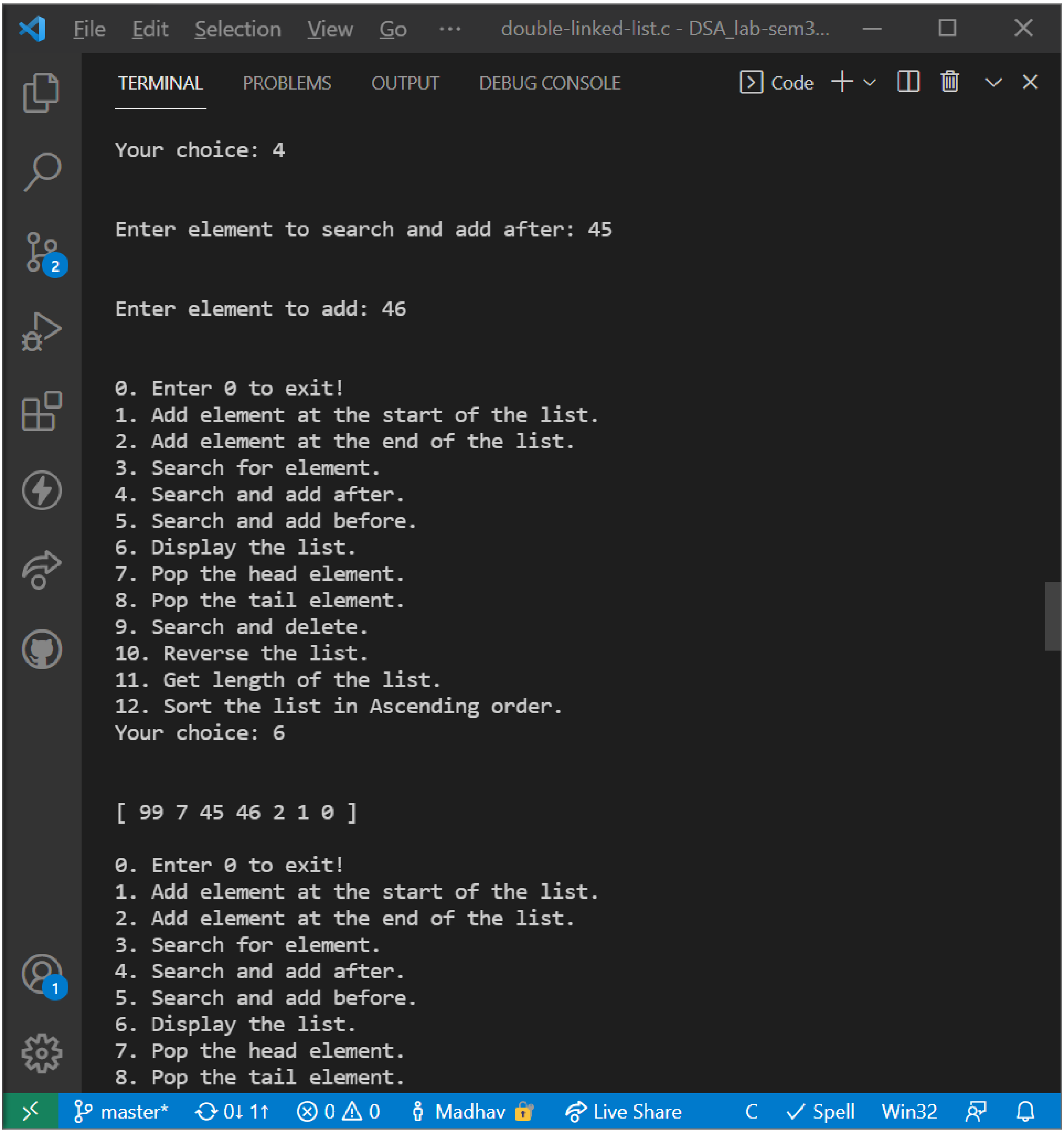
```
Your choice: 6


[ 99 7 45 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.
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.
12. Sort the list in Ascending order.
Your choice: 2


Enter element to add at end: 0

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.
```



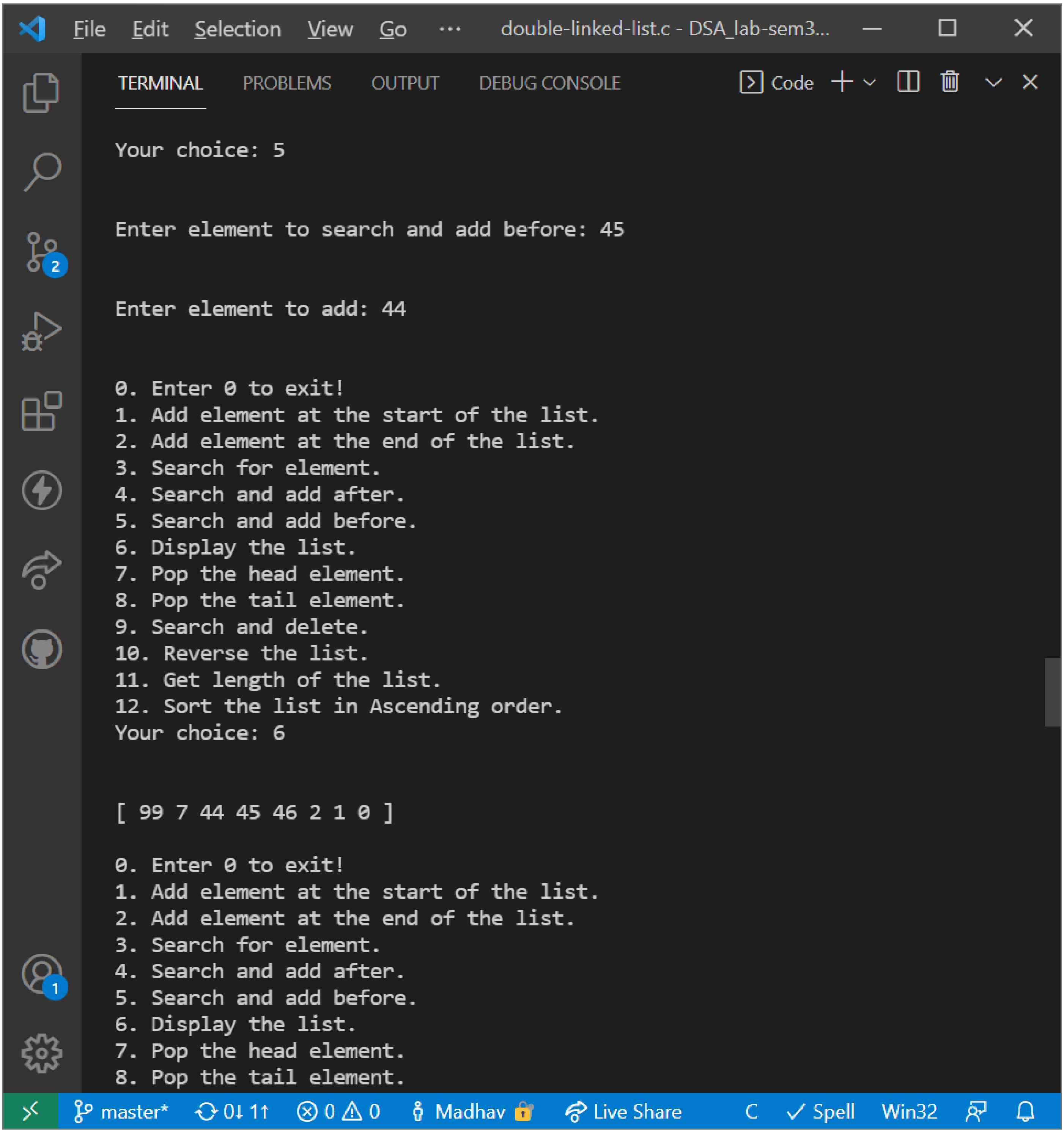
```
Your choice: 6


[ 99 7 45 2 1 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.
12. Sort the list in Ascending order.
Your choice: 3


Enter element to search: 7

Is element 7 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.
```



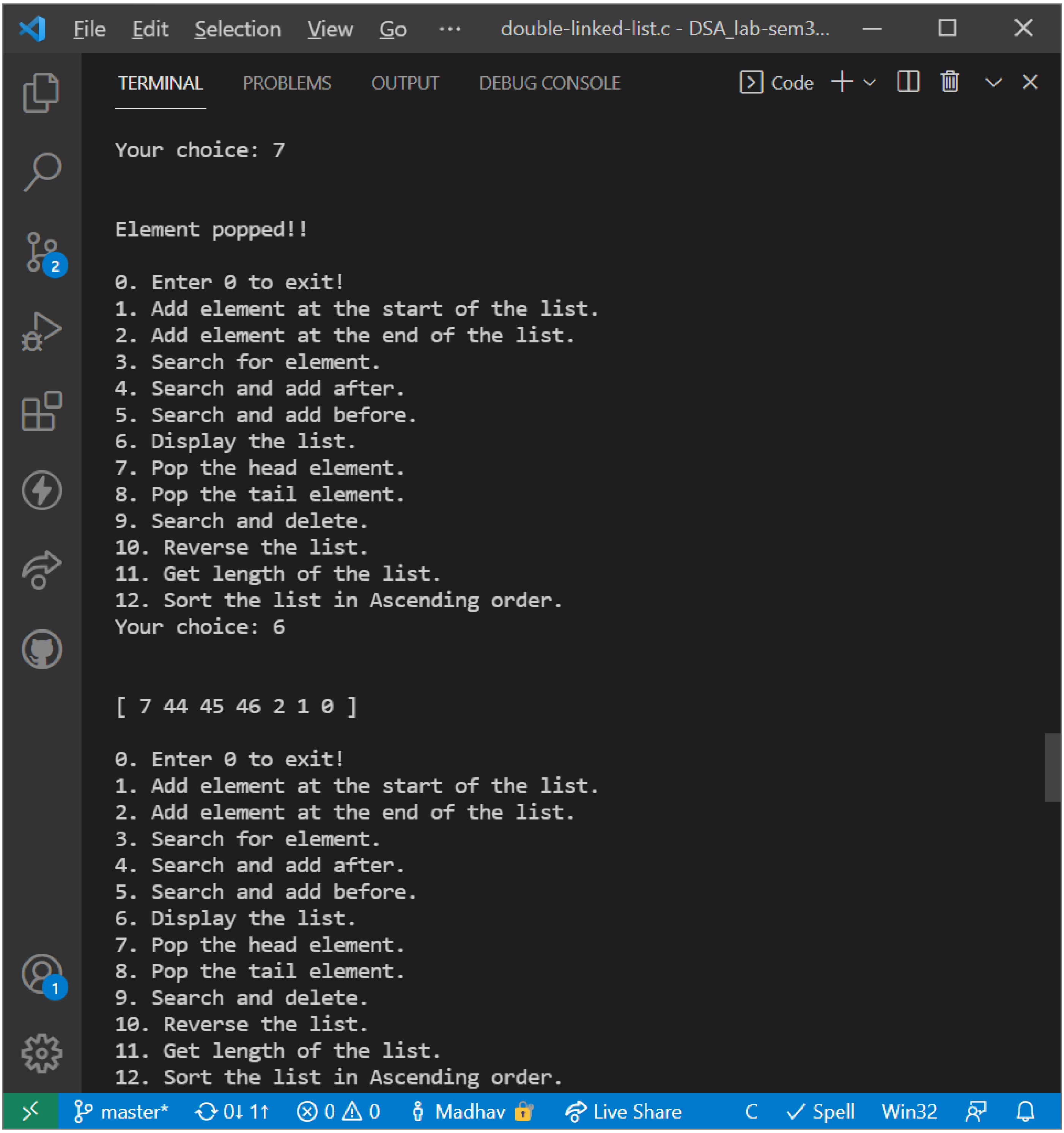
```
Your choice: 3


Enter element to search: 69

Is element 69 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.
12. Sort the list in Ascending order.
Your choice: 6


[ 99 7 45 2 1 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.
```



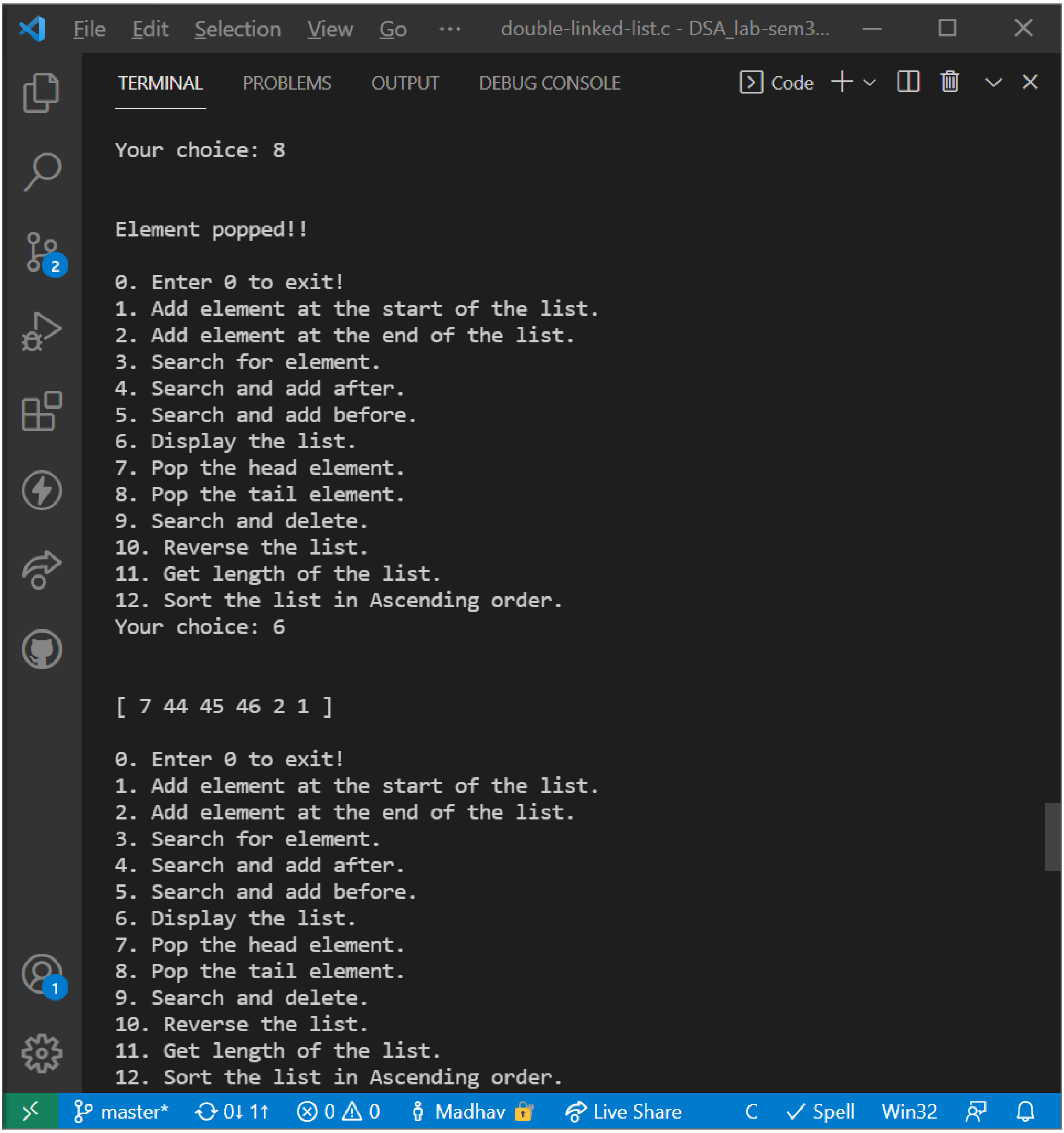
```
Your choice: 4


Enter element to search and add after: 45


Enter element to add: 46


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.
12. Sort the list in Ascending order.
Your choice: 6


[ 99 7 45 46 2 1 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.
```



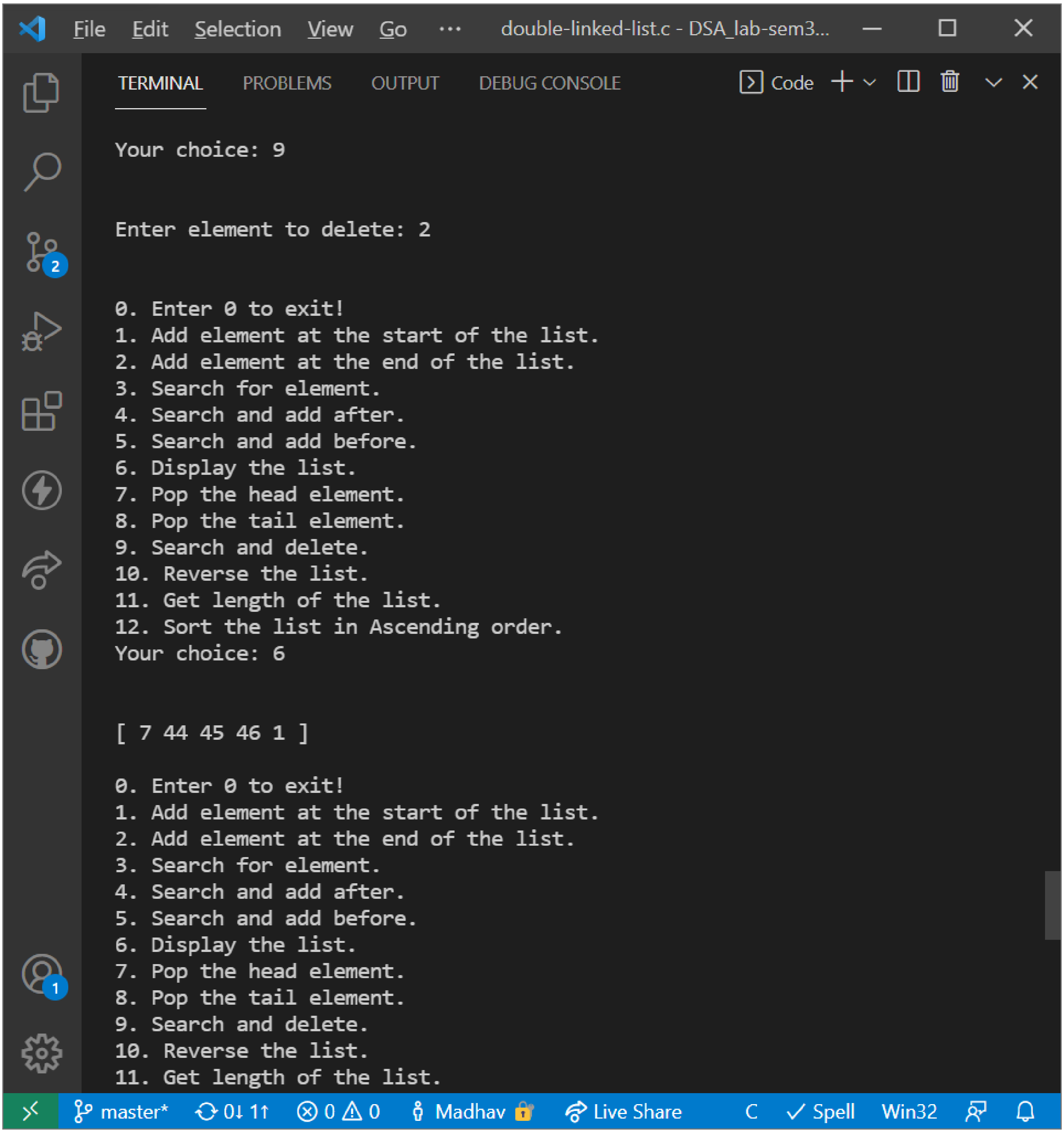
```
Your choice: 5


Enter element to search and add before: 45


Enter element to add: 44


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.
12. Sort the list in Ascending order.
Your choice: 6


[ 99 7 44 45 46 2 1 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.
```



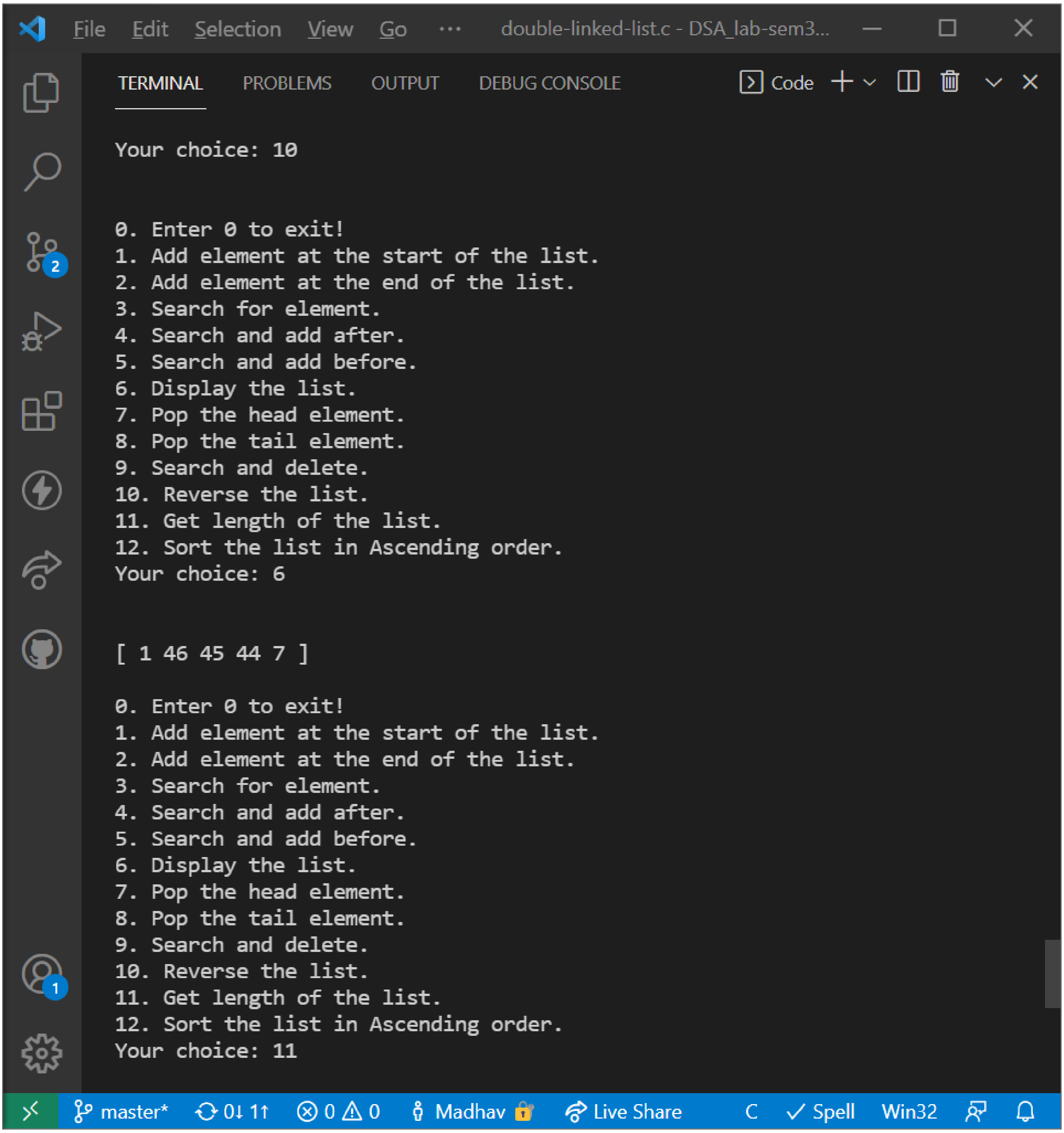
```
Your choice: 7


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.
12. Sort the list in Ascending order.
Your choice: 6


[ 7 44 45 46 2 1 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.
12. Sort the list in Ascending order.
```



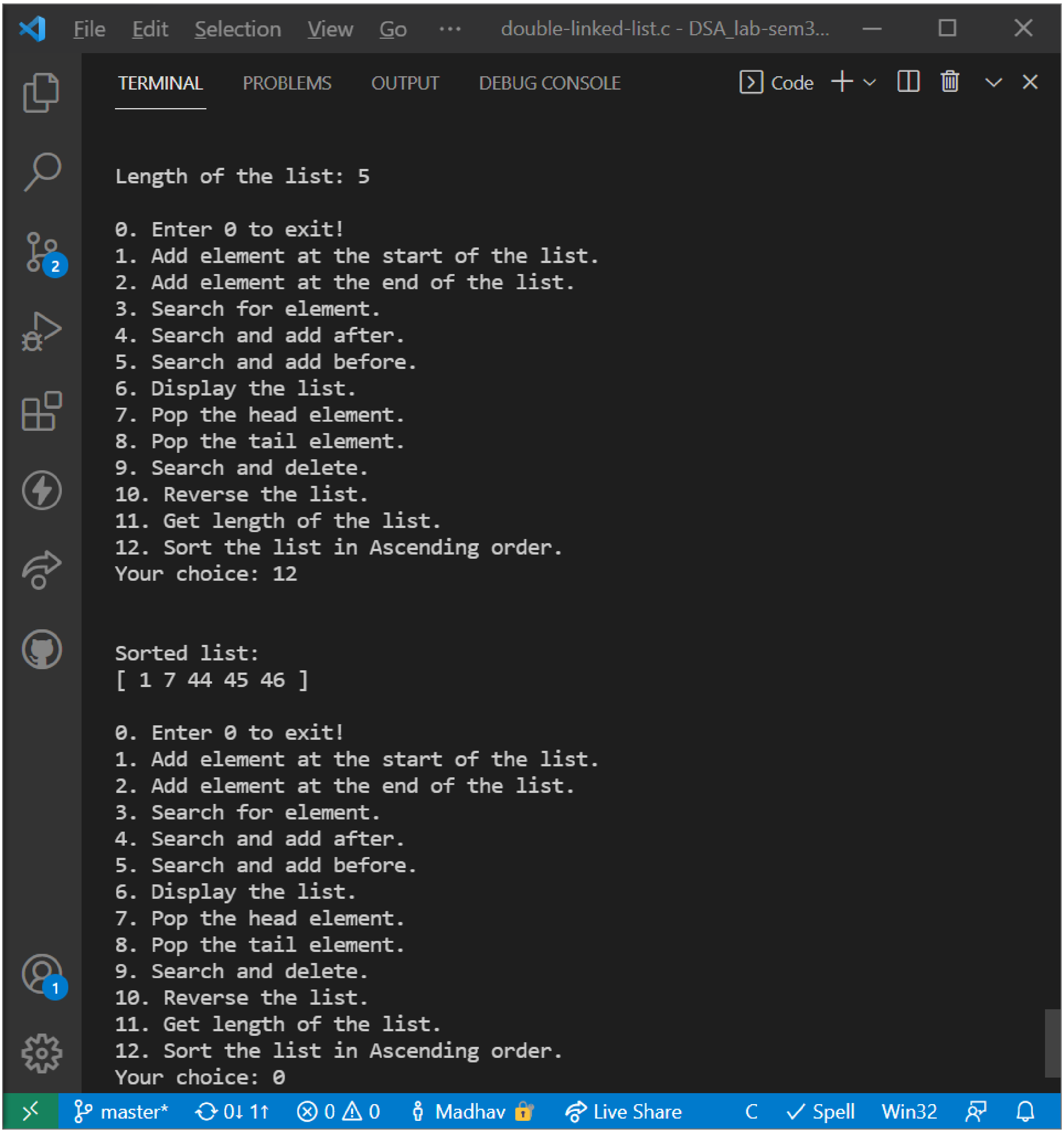
```
Your choice: 8


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.
12. Sort the list in Ascending order.
Your choice: 6


[ 7 44 45 46 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.
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.
12. Sort the list in Ascending order.
```



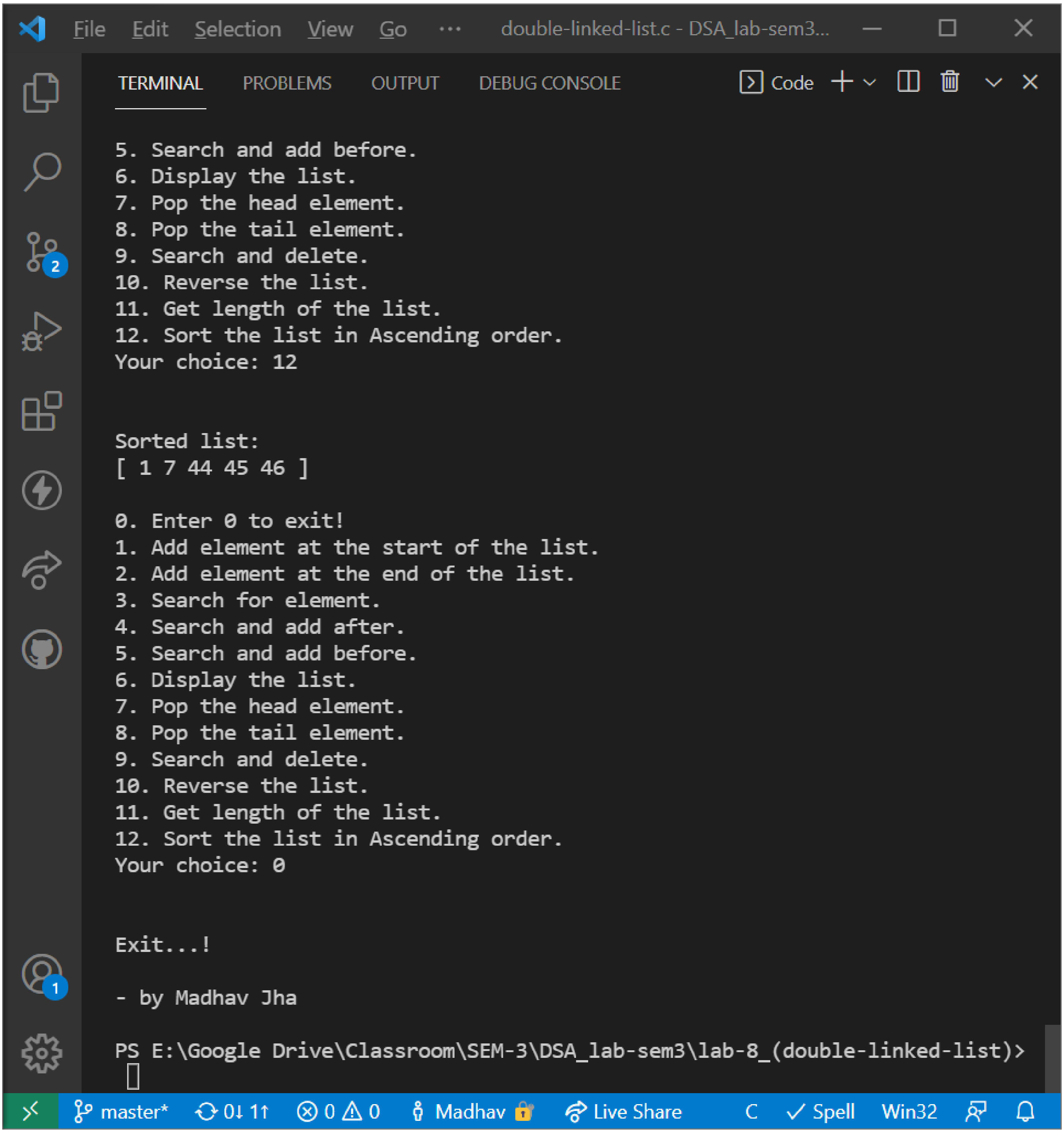
```
Your choice: 9


Enter element to delete: 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.
12. Sort the list in Ascending order.
Your choice: 6


[ 7 44 45 46 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.
12. Sort the list in Ascending order.
Your choice: 6


[ 1 46 45 44 7 ]

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.
12. Sort the list in Ascending order.
Your choice: 11
```



```
Length of the list: 5

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.
12. Sort the list in Ascending order.
Your choice: 12


Sorted list:
[ 1 7 44 45 46 ]

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.
12. Sort the list in Ascending order.
Your choice: 0
```



```
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.
12. Sort the list in Ascending order.
Your choice: 12


Sorted list:
[ 1 7 44 45 46 ]

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.
12. Sort the list in Ascending order.
Your choice: 0


Exit...!

- by Madhav Jha

PS E:\Google Drive\Classroom\SEM-3\DSA_lab-sem3\lab-8_(double-linked-list)>
```