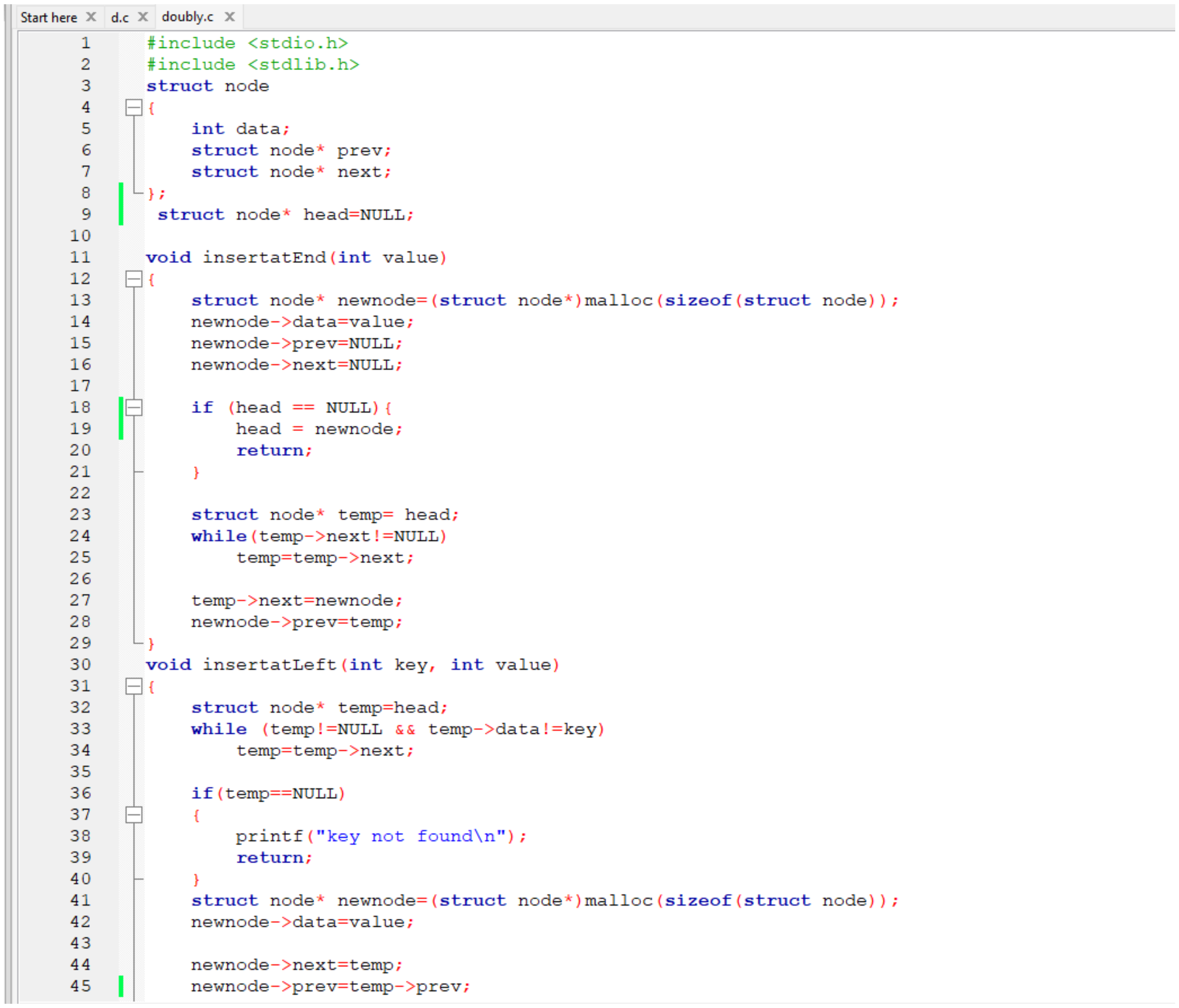
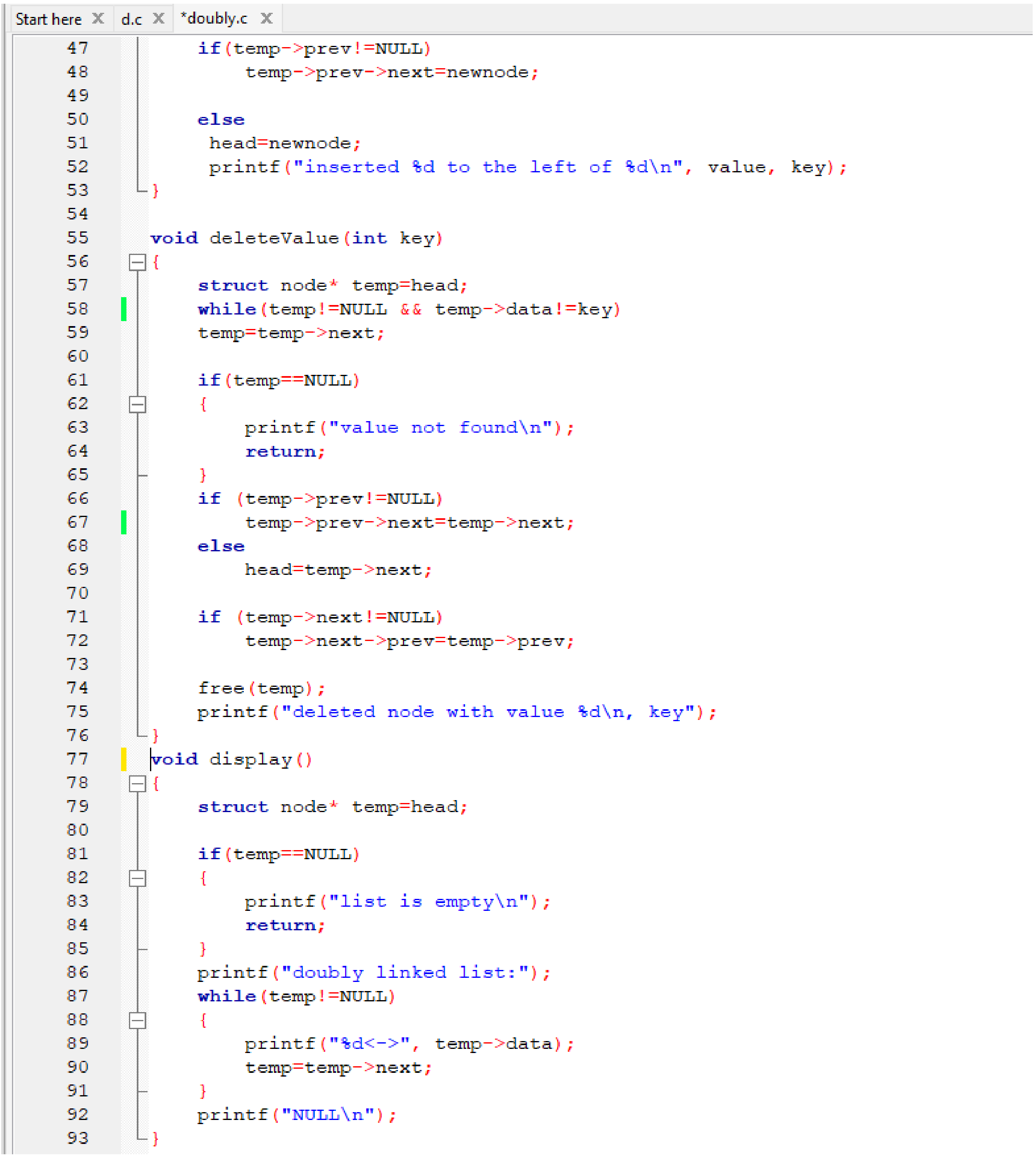
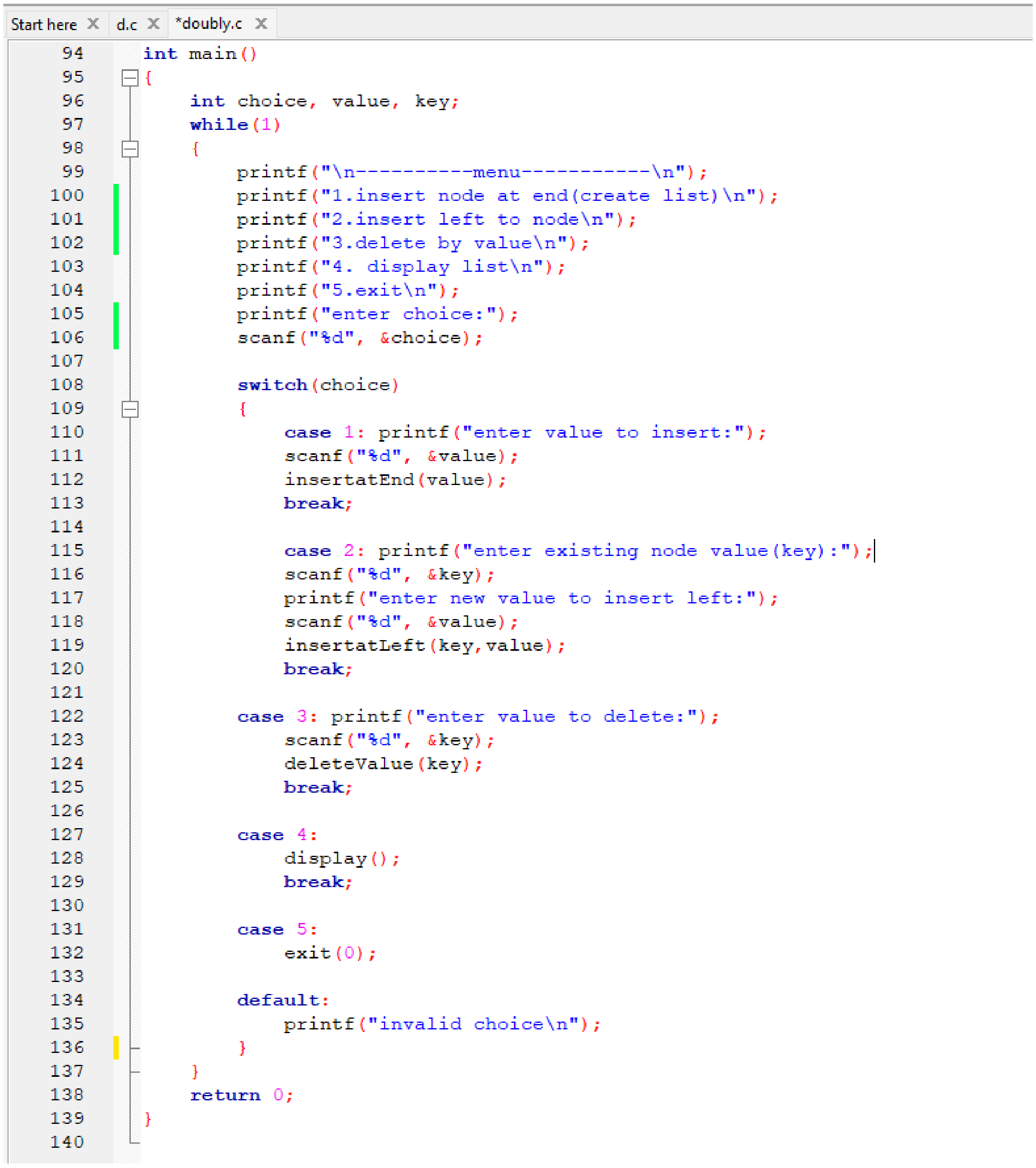
DS WEEK-8



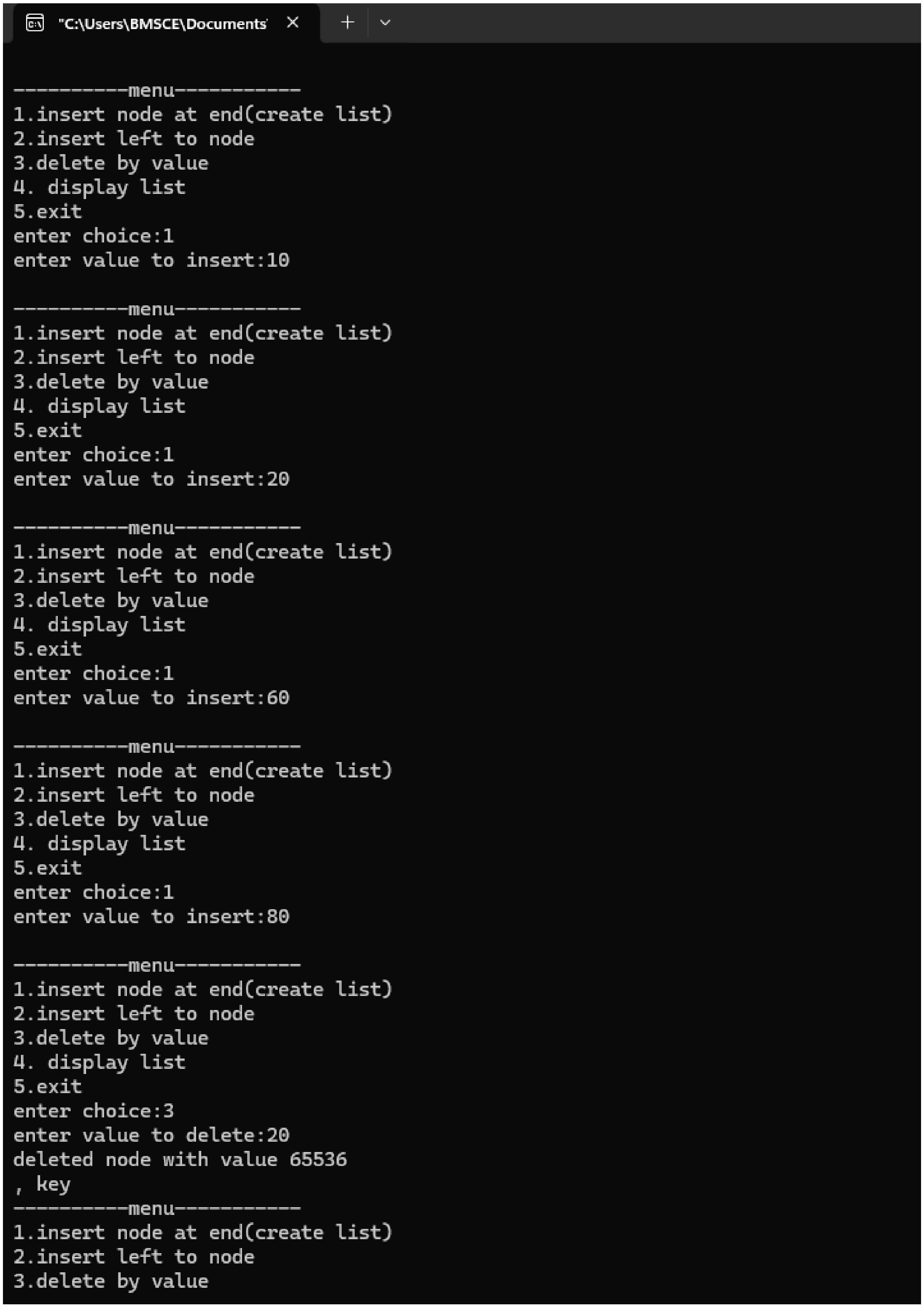
```c
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node* prev;
    struct node* next;
};
 struct node* head=NULL;

void insertatEnd(int value)
{
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=value;
    newnode->prev=NULL;
    newnode->next=NULL;

    if (head == NULL){
        head = newnode;
        return;
    }

    struct node* temp= head;
    while(temp->next!=NULL)
        temp=temp->next;

    temp->next=newnode;
    newnode->prev=temp;
}
void insertatLeft(int key, int value)
{
    struct node* temp=head;
    while (temp!=NULL && temp->data!=key)
        temp=temp->next;

    if(temp==NULL)
    {
        printf("key not found\n");
        return;
    }
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=value;

    newnode->next=temp;
    newnode->prev=temp->prev;
```


Start here ✕ d.c ✕ *doubly.c ✕

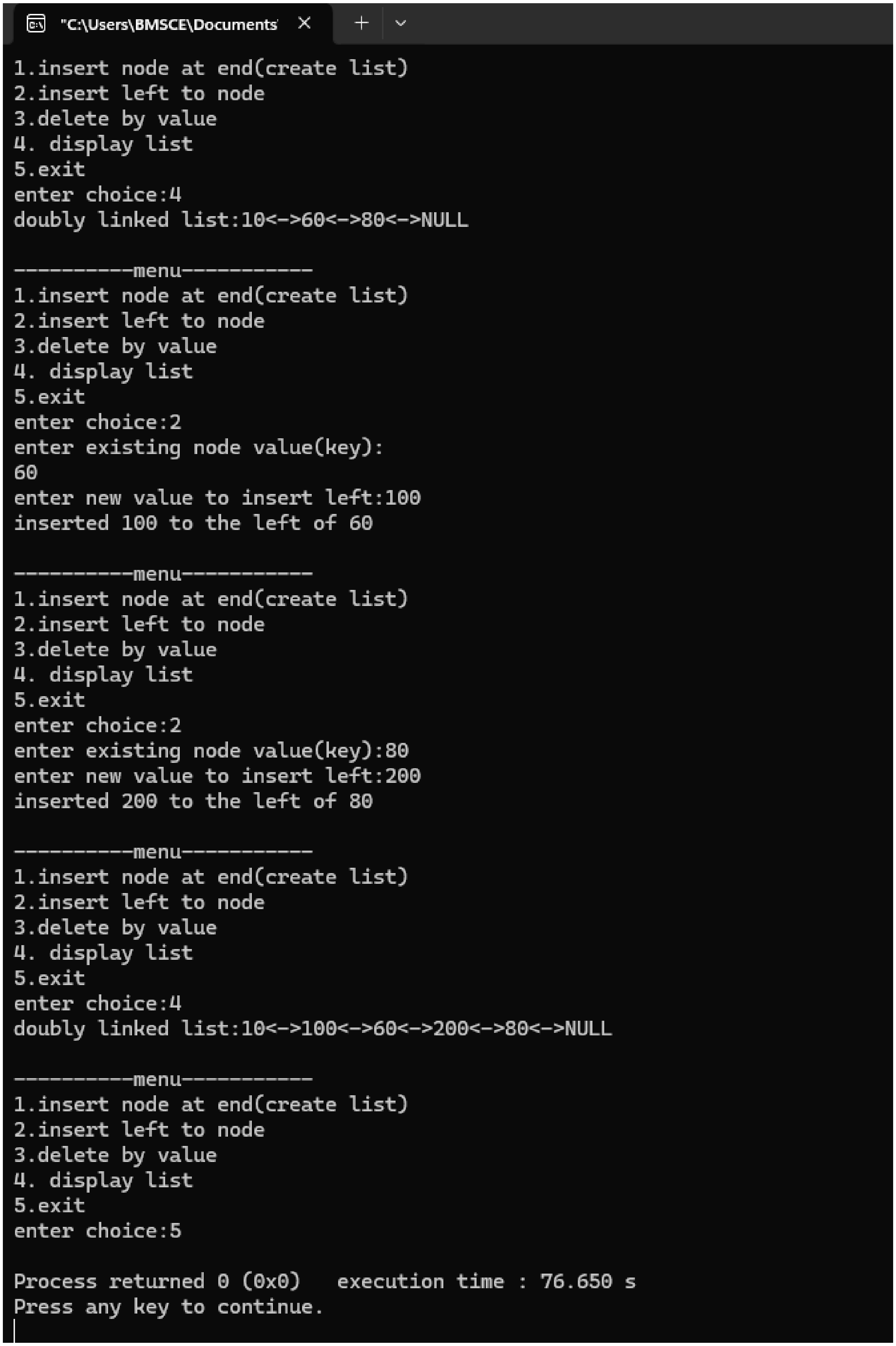
```c
    if(temp->prev!=NULL)
        temp->prev->next=newnode;

    else
     head=newnode;
     printf("inserted %d to the left of %d\n", value, key);
}

void deleteValue(int key)
{
    struct node* temp=head;
    while(temp!=NULL && temp->data!=key)
    temp=temp->next;

    if(temp==NULL)
    {
        printf("value not found\n");
        return;
    }
    if (temp->prev!=NULL)
        temp->prev->next=temp->next;
    else
        head=temp->next;

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

    free(temp);
    printf("deleted node with value %d\n, key");
}
void display()
{
    struct node* temp=head;

    if(temp==NULL)
    {
        printf("list is empty\n");
        return;
    }
    printf("doubly linked list:");
    while(temp!=NULL)
    {
        printf("%d<->", temp->data);
        temp=temp->next;
    }
    printf("NULL\n");
}
```



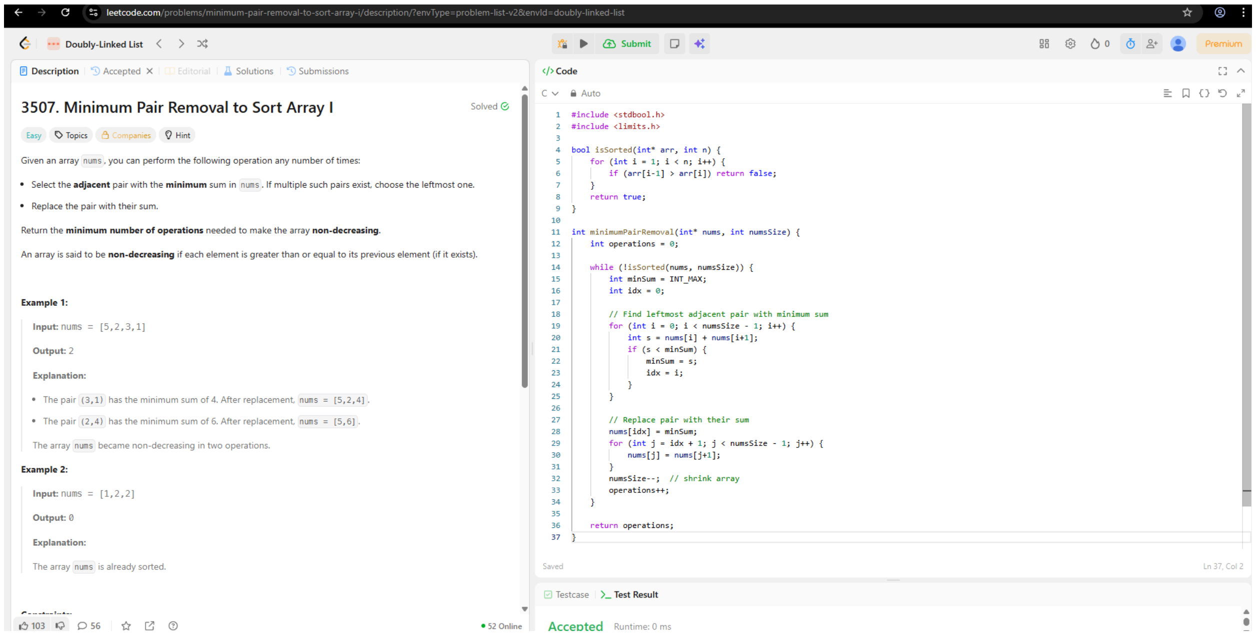
```c
int main()
{
    int choice, value, key;
    while(1)
    {
        printf("\n----------menu-----------\n");
        printf("1.insert node at end(create list)\n");
        printf("2.insert left to node\n");
        printf("3.delete by value\n");
        printf("4. display list\n");
        printf("5.exit\n");
        printf("enter choice:");
        scanf("%d", &choice);

        switch(choice)
        {
            case 1: printf("enter value to insert:");
            scanf("%d", &value);
            insertatEnd(value);
            break;

            case 2: printf("enter existing node value(key):");
            scanf("%d", &key);
            printf("enter new value to insert left:");
            scanf("%d", &value);
            insertatLeft(key,value);
            break;

        case 3: printf("enter value to delete:");
            scanf("%d", &key);
            deleteValue(key);
            break;

        case 4:
            display();
            break;

        case 5:
            exit(0);

        default:
            printf("invalid choice\n");
        }
    }
    return 0;
}
```



```
----------menu-----------
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:1
enter value to insert:10

----------menu-----------
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:1
enter value to insert:20

----------menu-----------
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:1
enter value to insert:60

----------menu-----------
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:1
enter value to insert:80

----------menu-----------
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:3
enter value to delete:20
deleted node with value 65536
, key
----------menu-----------
1.insert node at end(create list)
2.insert left to node
3.delete by value
```



```
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:4
doubly linked list:10<->60<->80<->NULL

----------menu-----------
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:2
enter existing node value(key):
60
enter new value to insert left:100
inserted 100 to the left of 60

----------menu-----------
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:2
enter existing node value(key):80
enter new value to insert left:200
inserted 200 to the left of 80

----------menu-----------
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:4
doubly linked list:10<->100<->60<->200<->80<->NULL

----------menu-----------
1.insert node at end(create list)
2.insert left to node
3.delete by value
4. display list
5.exit
enter choice:5

Process returned 0 (0x0)   execution time : 76.650 s
Press any key to continue.
```


leetcode.com/problems/minimum-pair-removal-to-sort-array-i/description/?envType=problem-list-v2&envId=doubly-linked-list

Doubly-Linked List

Description | Accepted | Editorial | Solutions | Submissions

3507. Minimum Pair Removal to Sort Array I

Solved

Easy · Topics · Companies · Hint

Given an array `nums`, you can perform the following operation any number of times:

- Select the **adjacent** pair with the **minimum** sum in `nums`. If multiple such pairs exist, choose the leftmost one.
- Replace the pair with their sum.

Return the **minimum number of operations** needed to make the array **non-decreasing**.

An array is said to be **non-decreasing** if each element is greater than or equal to its previous element (if it exists).

Example 1:

Input: nums = [5,2,3,1]

Output: 2

Explanation:

- The pair `(3,1)` has the minimum sum of 4. After replacement, `nums = [5,2,4]`.
- The pair `(2,4)` has the minimum sum of 6. After replacement, `nums = [5,6]`.

The array `nums` became non-decreasing in two operations.

Example 2:

Input: nums = [1,2,2]

Output: 0

Explanation:

The array `nums` is already sorted.

103 · 56 · 52 Online

Code

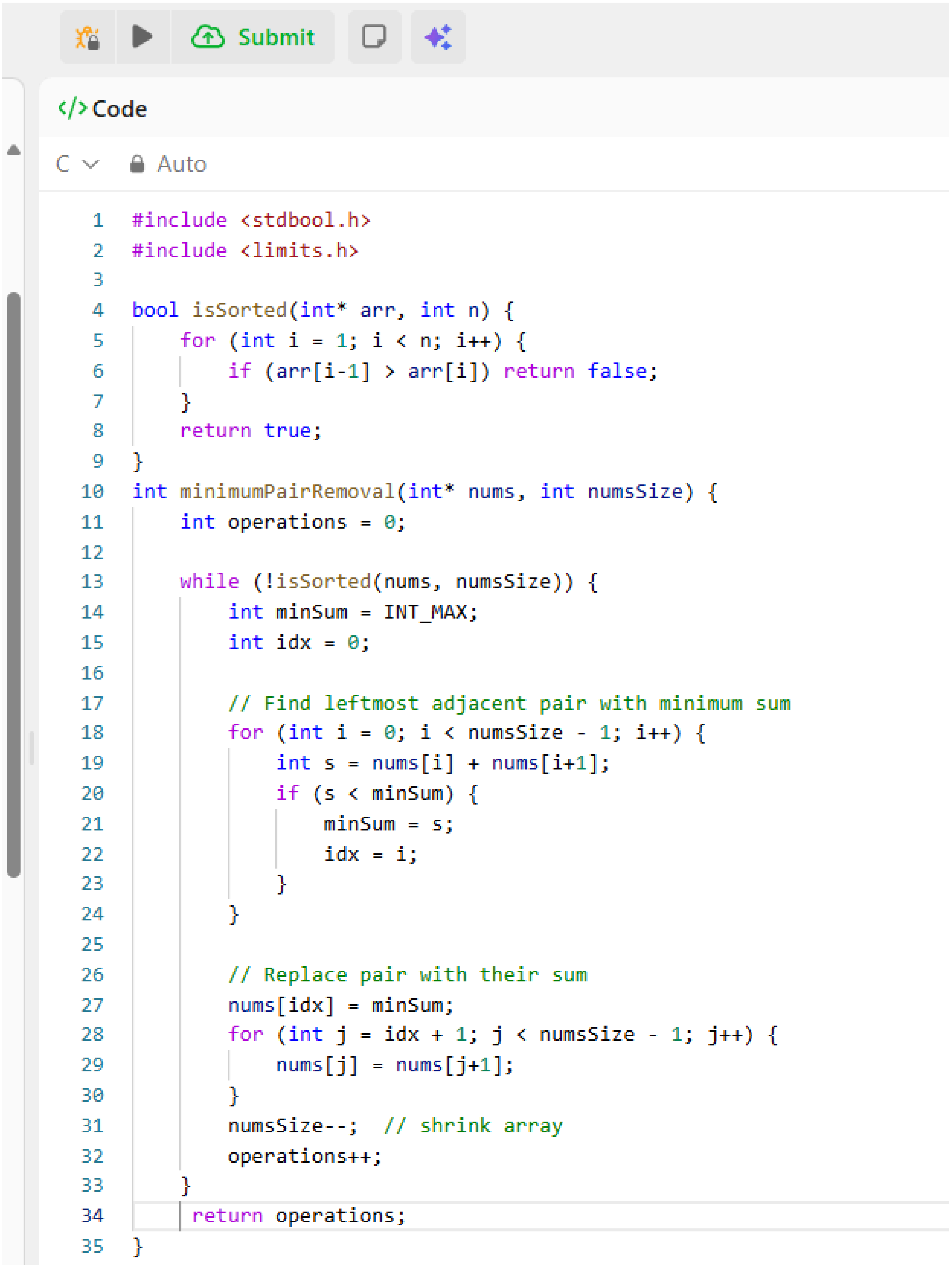
C · Auto

```c
#include <stdbool.h>
#include <limits.h>

bool isSorted(int* arr, int n) {
    for (int i = 1; i < n; i++) {
        if (arr[i-1] > arr[i]) return false;
    }
    return true;
}

int minimumPairRemoval(int* nums, int numsSize) {
    int operations = 0;

    while (!isSorted(nums, numsSize)) {
        int minSum = INT_MAX;
        int idx = 0;

        // Find leftmost adjacent pair with minimum sum
        for (int i = 0; i < numsSize - 1; i++) {
            int s = nums[i] + nums[i+1];
            if (s < minSum) {
                minSum = s;
                idx = i;
            }
        }

        // Replace pair with their sum
        nums[idx] = minSum;
        for (int j = idx + 1; j < numsSize - 1; j++) {
            nums[j] = nums[j+1];
        }
        numsSize--;  // shrink array
        operations++;
    }

    return operations;
}
```

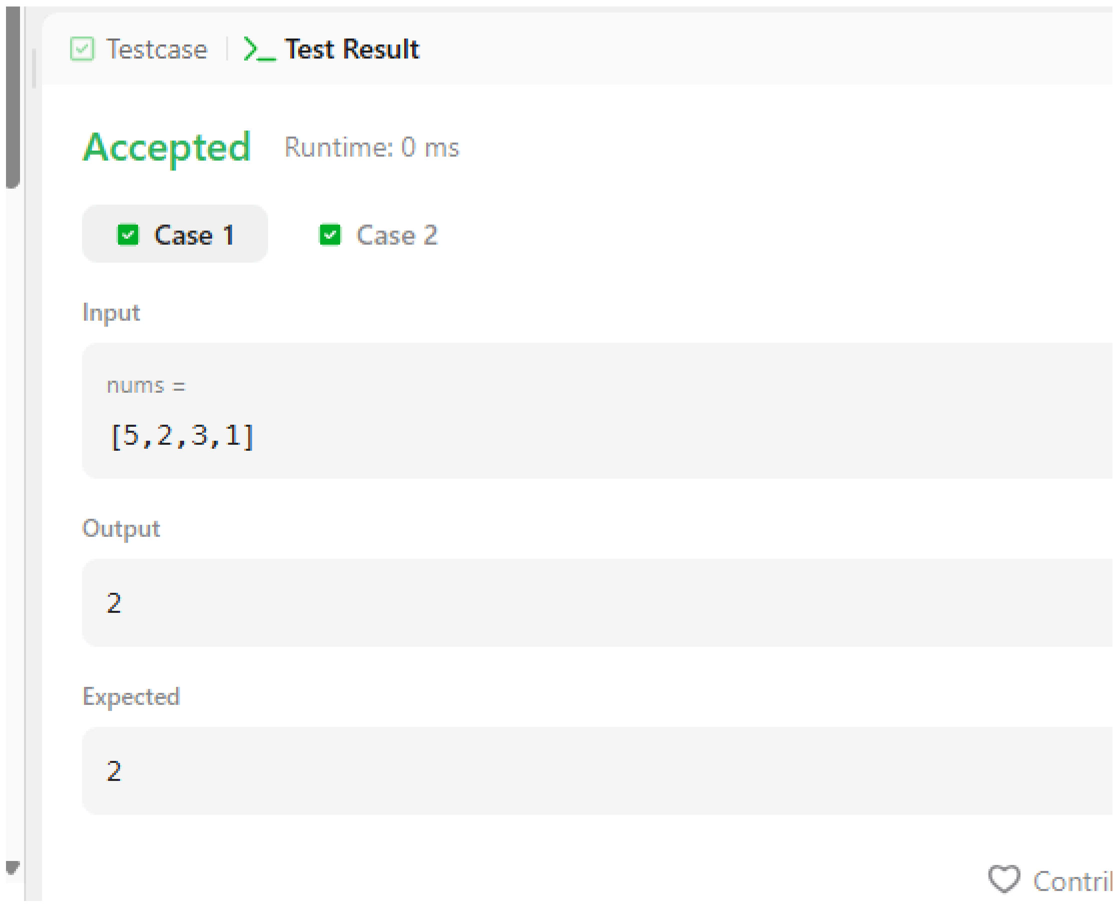
Saved · Ln 37, Col 2

Testcase | Test Result

Accepted Runtime: 0 ms



```c
#include <stdbool.h>
#include <limits.h>

bool isSorted(int* arr, int n) {
    for (int i = 1; i < n; i++) {
        if (arr[i-1] > arr[i]) return false;
    }
    return true;
}
int minimumPairRemoval(int* nums, int numsSize) {
    int operations = 0;

    while (!isSorted(nums, numsSize)) {
        int minSum = INT_MAX;
        int idx = 0;

        // Find leftmost adjacent pair with minimum sum
        for (int i = 0; i < numsSize - 1; i++) {
            int s = nums[i] + nums[i+1];
            if (s < minSum) {
                minSum = s;
                idx = i;
            }
        }

        // Replace pair with their sum
        nums[idx] = minSum;
        for (int j = idx + 1; j < numsSize - 1; j++) {
            nums[j] = nums[j+1];
        }
        numsSize--;  // shrink array
        operations++;
    }
     return operations;
}
```


Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

nums =

```
[5,2,3,1]
```

Output

```
2
```

Expected

```
2
```