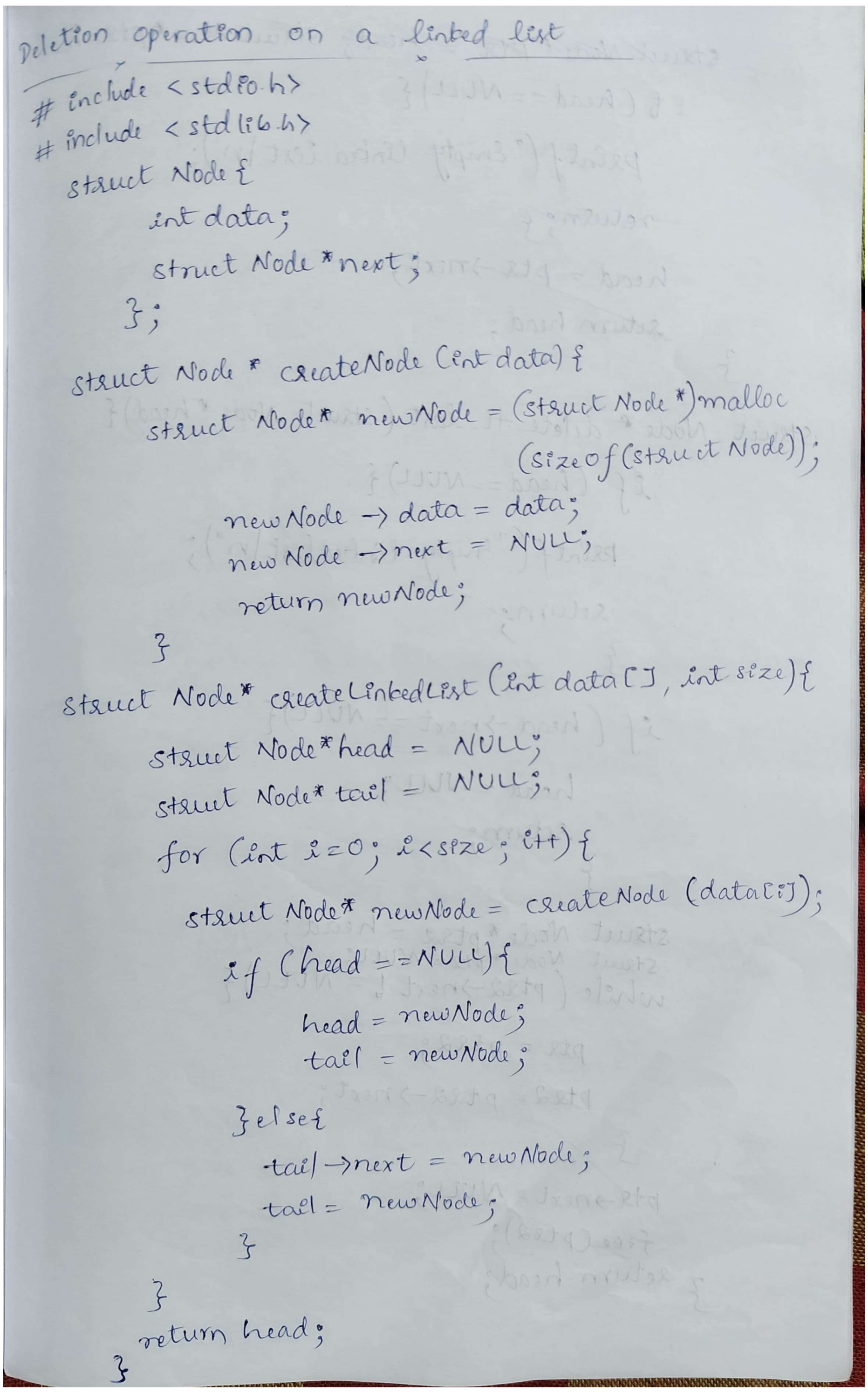
Program 6 :
Write a Program to Implement Singly Linked List with following operations
a) Create a linked list.
b) Deletion of first element, specified element and last element in the list.
c) Display the contents of the linked list. Leetcode Problem -- 739 (Daily Temperature)

Observation:



Deletion operation on a linked list

```
# include <stdio.h>
# include <stdlib.h>
struct Node {
        int data;
        struct Node *next;
    };

struct Node * createNode (int data) {
    struct Node* newNode = (struct Node *)malloc
                              (sizeof(struct Node));
        newNode -> data = data;
        newNode -> next = NULL;
        return newNode;
    }
struct Node* createLinkedList (int data[], int size){
    struct Node*head = NULL;
    struct Node* tail = NULL;
    for (int i=0; i<size; i++){
      struct Node* newNode = createNode (data[i]);
        if (head == NULL){
            head = newNode;
            tail = newNode;
        }else{
            tail->next = newNode;
            tail = newNode;
        }
    }
    return head;
}
```

```
struct Node * delete_At first ( struc
    ~~struct Node* ptr~~ = head; struct Node * ptr = head;
      if (head == NULL){
        printf(" Empty linked list \n");
        return; }
      head = ptr->next;
      return head;
  }
struct Node * delete_At_last (struct Node * head){
        if (head == NULL){
          printf(" Empty linkedlist \n");
          return;
        }
      if (head->next == NULL){
          head = NULL;
          return;
        }
      struct Node *ptr2 = head;
      struct Node * ptr = NULL;
      while (ptr2->next != NULL){
          ptr = ptr2;
          ptr2 = ptr2->next;
      }
      ptr->next = NULL;
      free(ptr2);
     return head;
  }
```

```
struct Node* deletespecified (struct Node* head, int ~~index~~ value);
    if (head = =NULL){
        printf("List is empty\n");
        return;
    }

    if (head->data = = value){
     struct Node* temp = head;
        head = head->next;
        free (temp);
        return;
    }

  struct Node* temp = head;
  struct Node* prev = NULL;
   while (temp ! =NULL && temp->data ! = value){
        prev = temp;
        temp = temp->next;
   }
   if (temp = =NULL){
        printf("Element %d not found\n", value);
        return;
   }
   prev->next = temp->next;
     free (temp);
     return head;
}
```

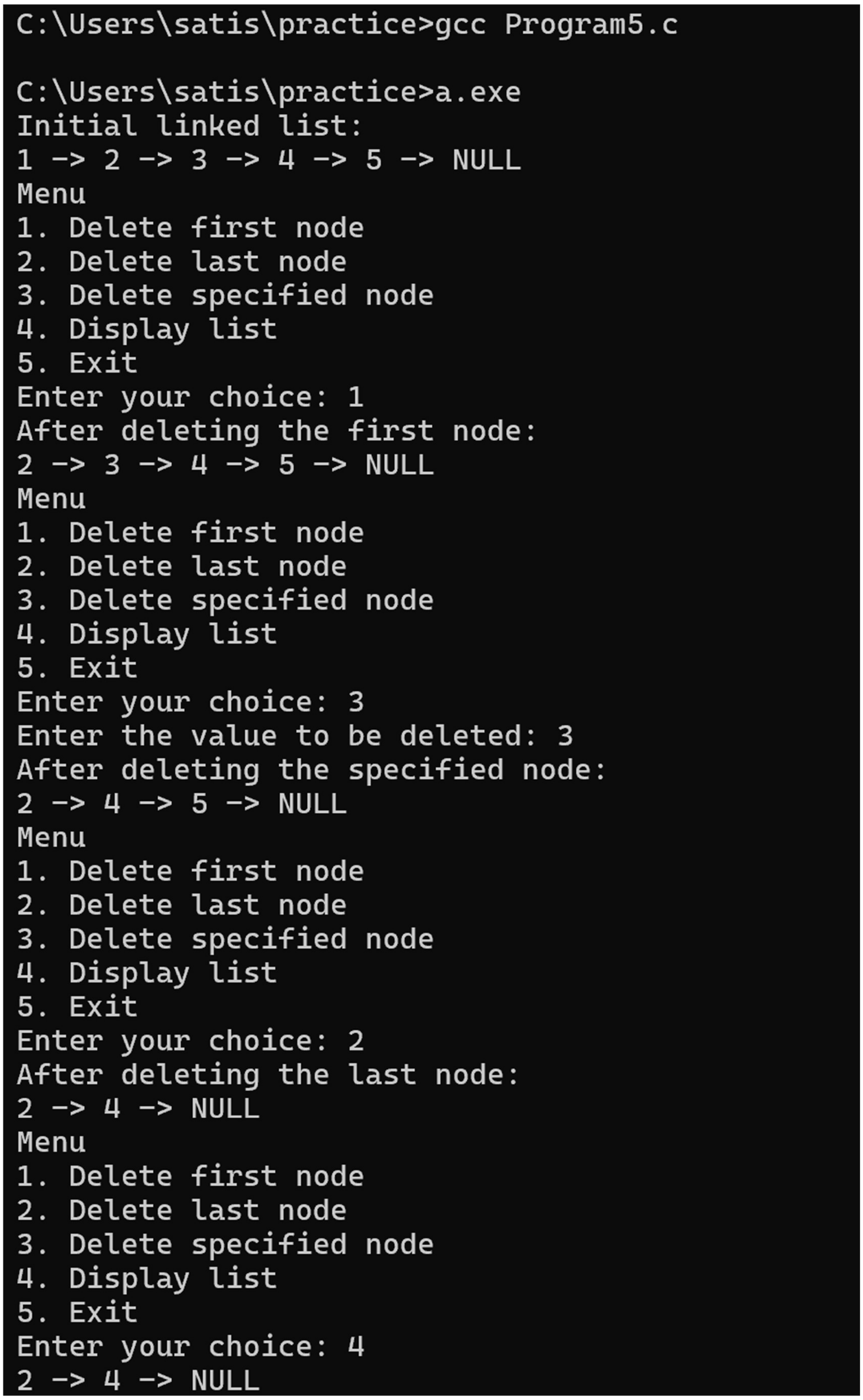
Seen
o/p seen

Code:

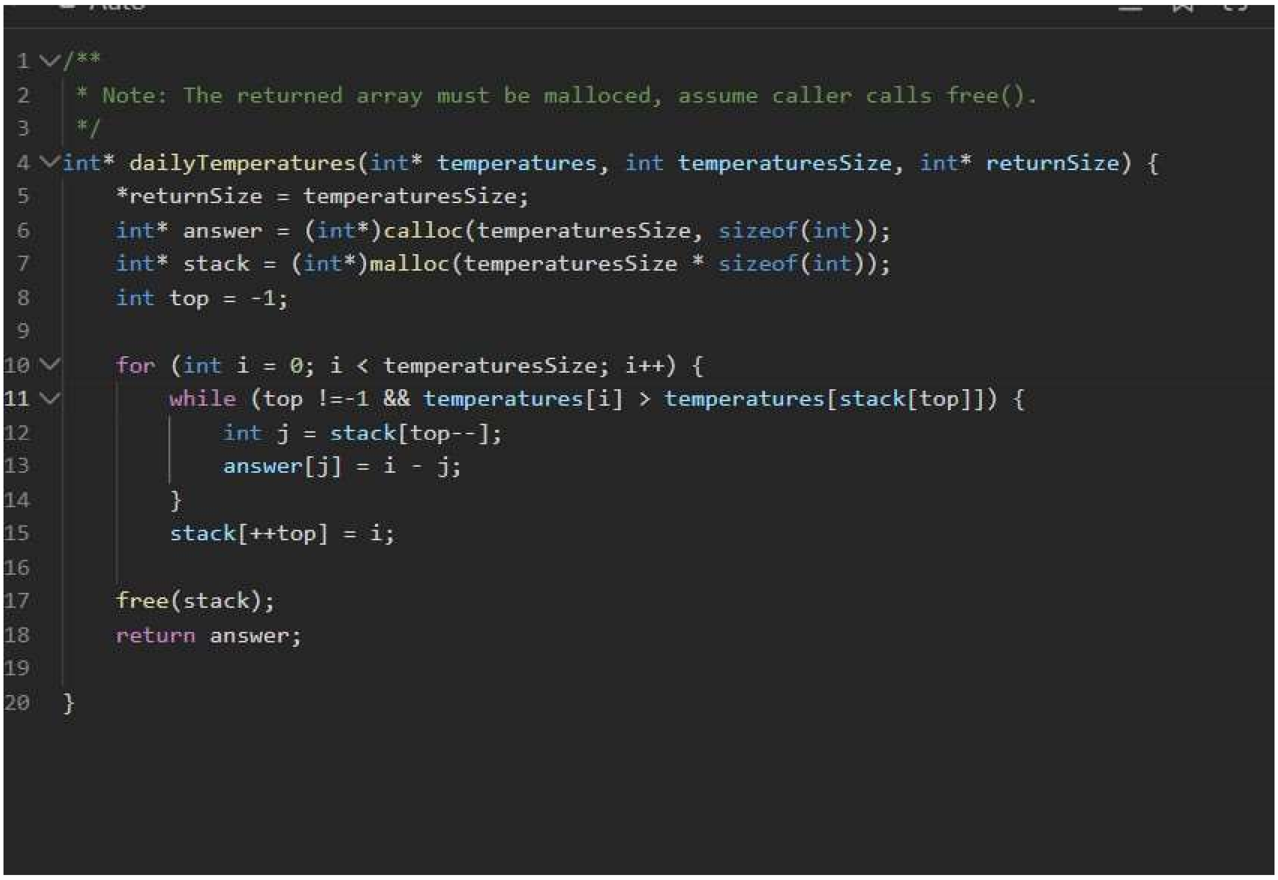
```
 #include <stdio.h>
#include <stdlib.h>

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

struct Node *createNode(int data)
{
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

struct Node *createLinkedList(int data[], int size)
{
    struct Node *head = NULL;
    struct Node *tail = NULL;

    for (int i = 0; i < size; i++)
    {
        struct Node *newNode = createNode(data[i]);
        if (head == NULL)
        {
            head = newNode;
            tail = newNode;
        }
        else
        {
            tail->next = newNode;
            tail = newNode;
        }
    }

    return head;
}

void display(struct Node *head)
{
    struct Node *current = head;
    while (current != NULL)
    {
        printf("%d -> ", current->data);
        current = current->next;
```

```c
    }
    printf("NULL\n");
}

struct Node *delete_at_first(struct Node *head)
{
    if (head == NULL)
    {
        printf("Empty linked list\n");
        return head;
    }

    struct Node *ptr = head;
    head = ptr->next;
    free(ptr);
    return head;
}

struct Node *delete_at_last(struct Node *head)
{
    if (head == NULL)
    {
        printf("Empty linked list\n");
        return head;
    }

    if (head->next == NULL)
    {
        free(head);
        return NULL;
    }

    struct Node *ptr2 = head;
    struct Node *ptr = NULL;
    while (ptr2->next != NULL)
    {
        ptr = ptr2;
        ptr2 = ptr2->next;
    }
    ptr->next = NULL;
    free(ptr2);
    return head;
}

struct Node *deletespecified(struct Node *head, int value)
{
    if (head == NULL)
    {
        printf("List is empty!\n");
        return head;
    }

    if (head->data == value)
    {
        struct Node *temp = head;
        head = temp->next;
        free(temp);
        return head;
    }

    struct Node *current = head;
    while (current->next != NULL && current->next->data != value)
    {
        current = current->next;
    }

    if (current->next == NULL)
    {
        printf("Value %d not found in the list!\n", value);
    }
    else
    {
        struct Node *temp = current->next;
        current->next = current->next->next;
        free(temp);
    }
    return head;
}

int main()
{
    int data[] = {1, 2, 3, 4, 5};
    struct Node *linkedList = createLinkedList(data, 5);

    printf("Initial linked list:\n");
    display(linkedList);

    int choice;
    printf("Menu\n1. Delete first node\n2. Delete last node\n3. Delete specified
node\n4. Display list\n5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    while (choice != 5)
    {
        if (choice == 1)
        {
            linkedList = delete_at_first(linkedList);
            printf("After deleting the first node:\n");
            display(linkedList);
        }
        if (choice == 2)
        {
            linkedList = delete_at_last(linkedList);
            printf("After deleting the last node:\n");
            display(linkedList);
        }
        if (choice == 3)
        {
            int value;
            printf("Enter the value to be deleted: ");
            scanf("%d", &value);
            linkedList = deletespecified(linkedList, value);
            printf("After deleting the specified node:\n");
            display(linkedList);
        }
        if (choice == 4)
        {
            display(linkedList);
        }
        if (choice == 5)
        {
            exit(0);
        }

        printf("Menu\n1. Delete first node\n2. Delete last node\n3. Delete specified node\n4. Display list\n5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
    }

    return 0;
}
```

Output:



```
C:\Users\satis\practice>gcc Program5.c

C:\Users\satis\practice>a.exe
Initial linked list:
1 -> 2 -> 3 -> 4 -> 5 -> NULL
Menu
1. Delete first node
2. Delete last node
3. Delete specified node
4. Display list
5. Exit
Enter your choice: 1
After deleting the first node:
2 -> 3 -> 4 -> 5 -> NULL
Menu
1. Delete first node
2. Delete last node
3. Delete specified node
4. Display list
5. Exit
Enter your choice: 3
Enter the value to be deleted: 3
After deleting the specified node:
2 -> 4 -> 5 -> NULL
Menu
1. Delete first node
2. Delete last node
3. Delete specified node
4. Display list
5. Exit
Enter your choice: 2
After deleting the last node:
2 -> 4 -> NULL
Menu
1. Delete first node
2. Delete last node
3. Delete specified node
4. Display list
5. Exit
Enter your choice: 4
2 -> 4 -> NULL
```



```c
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* dailyTemperatures(int* temperatures, int temperaturesSize, int* returnSize) {
    *returnSize = temperaturesSize;
    int* answer = (int*)calloc(temperaturesSize, sizeof(int));
    int* stack = (int*)malloc(temperaturesSize * sizeof(int));
    int top = -1;

    for (int i = 0; i < temperaturesSize; i++) {
        while (top !=-1 && temperatures[i] > temperatures[stack[top]]) {
            int j = stack[top--];
            answer[j] = i - j;
        }
        stack[++top] = i;

    free(stack);
    return answer;

}
```

Leet Code Daily Temperatures