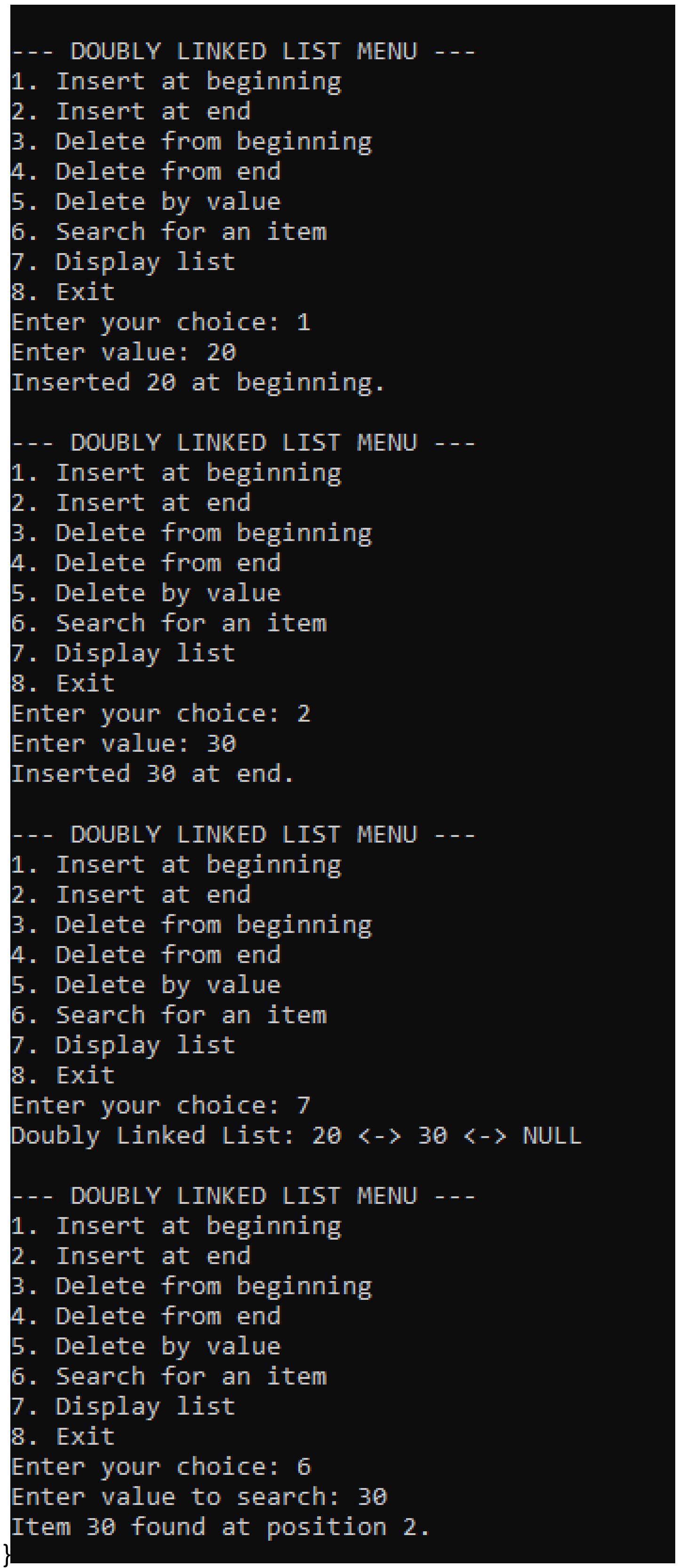
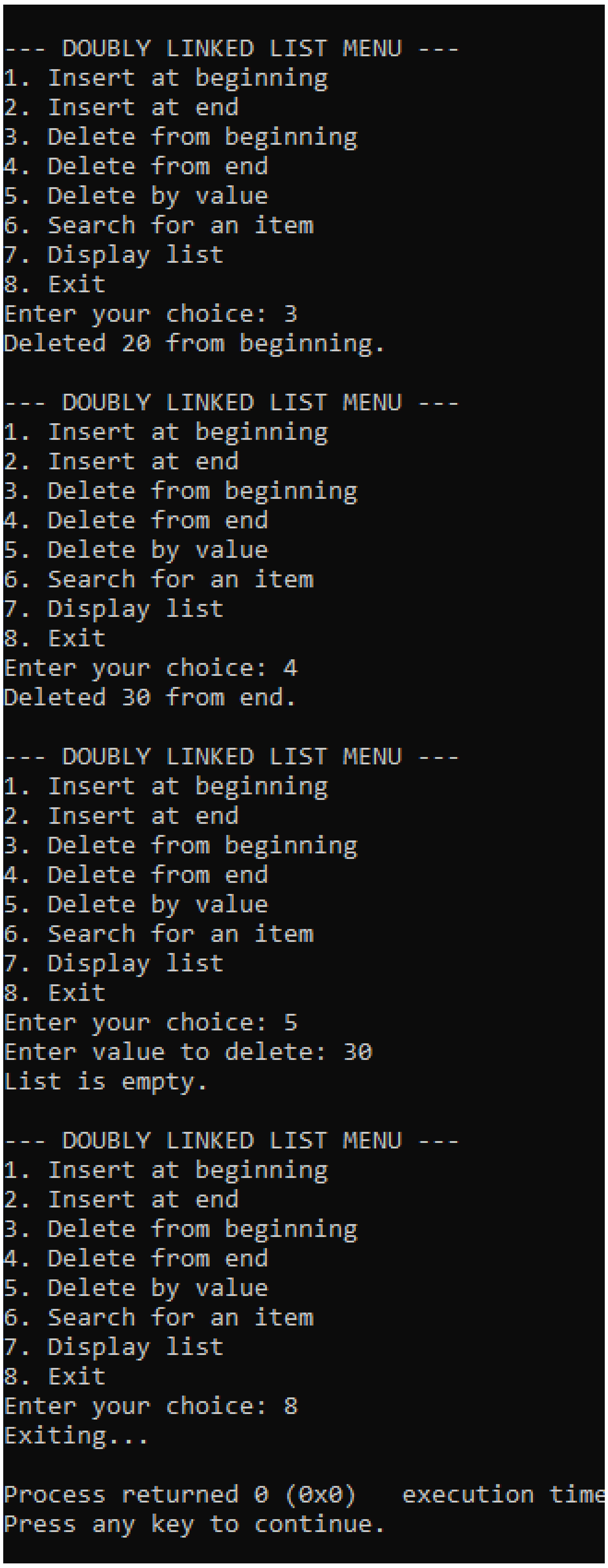
```c
#include <stdio.h>

#include <stdlib.h>

struct Node {

    int data;

    struct Node* prev;

    struct Node* next;

};

struct Node* head = NULL;

struct Node* createNode(int data) {

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

    newNode->data = data;

    newNode->prev = NULL;

    newNode->next = NULL;

    return newNode;

}

void insertAtBeginning(int data) {

    struct Node* newNode = createNode(data);

    newNode->next = head;

    if (head != NULL)

        head->prev = newNode;

    head = newNode;

    printf("Inserted %d at beginning.\n", data);

}

void insertAtEnd(int data) {

    struct Node* newNode = createNode(data);

    if (head == NULL) {

        head = newNode;

        printf("Inserted %d at end.\n", data);

        return;

    }
    struct Node* temp = head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
    printf("Inserted %d at end.\n", data);
}
void deleteBeginning() {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    struct Node* temp = head;
    head = head->next;
    if (head != NULL)
        head->prev = NULL;
    printf("Deleted %d from beginning.\n", temp->data);
    free(temp);
}
void deleteEnd() {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    struct Node* temp = head
    if (temp->next == NULL) {
        printf("Deleted %d from end.\n", temp->data);
        free(temp);
        head = NULL;
        return;
    }
    while (temp->next != NULL)
        temp = temp->next;
    printf("Deleted %d from end.\n", temp->data);
    temp->prev->next = NULL;
    free(temp);
}
void deleteByValue(int key) {
    struct Node* temp = head;
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    if (temp->data == key) {
        head = temp->next;
        if (head != NULL)
            head->prev = NULL;
        printf("Deleted %d.\n", key);
        free(temp);
        return;
    }
    while (temp != NULL && temp->data != key)
        temp = temp->next;
    if (temp == NULL) {
        printf("Value not found.\n");
        return;
    }
    if (temp->next != NULL)
        temp->next->prev = temp->prev;
    if (temp->prev != NULL)
        temp->prev->next = temp->next;
    printf("Deleted %d.\n", key);
    free(temp);
}
int search(int key) {
    struct Node* temp = head;
    int pos = 1;
    while (temp != NULL) {
        if (temp->data == key)
            return pos;
        temp = temp->next;
        pos++;
    }
    return -1;
}
void display() {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    struct Node* temp = head;
    printf("Doubly Linked List: ");
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
int main() {
    int choice, value, key, result;
    while (1) {
        printf("\n--- DOUBLY LINKED LIST MENU ---\n");
        printf("1. Insert at beginning\n");
        printf("2. Insert at end\n");
        printf("3. Delete from beginning\n");
        printf("4. Delete from end\n");
        printf("5. Delete by value\n");
        printf("6. Search for an item\n");
        printf("7. Display list\n");
        printf("8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
              printf("Enter value: ");
              scanf("%d", &value);
              insertAtBeginning(value);
              break;
            case 2:
              printf("Enter value: ");
              scanf("%d", &value);
              insertAtEnd(value);
              break;
            case 3:
              deleteBeginning();
              break;
            case 4:
              deleteEnd();
              break;
            case 5:
              printf("Enter value to delete: ");
              scanf("%d", &key);
              deleteByValue(key);
```

```
            break;
        case 6:
            printf("Enter value to search: ");
            scanf("%d", &key);
            result = search(key);
            if (result == -1)
                printf("Item %d NOT found.\n", key);
            else
                printf("Item %d found at position %d.\n", key, result);
            break;
        case 7:
            display();
            break;
        case 8:
            printf("Exiting...\n");
            exit(0);
        default:
            printf("Invalid choice! Try again.\n");
    }
}
return 0;
```



```
--- DOUBLY LINKED LIST MENU ---
1. Insert at beginning
2. Insert at end
3. Delete from beginning
4. Delete from end
5. Delete by value
6. Search for an item
7. Display list
8. Exit
Enter your choice: 1
Enter value: 20
Inserted 20 at beginning.

--- DOUBLY LINKED LIST MENU ---
1. Insert at beginning
2. Insert at end
3. Delete from beginning
4. Delete from end
5. Delete by value
6. Search for an item
7. Display list
8. Exit
Enter your choice: 2
Enter value: 30
Inserted 30 at end.

--- DOUBLY LINKED LIST MENU ---
1. Insert at beginning
2. Insert at end
3. Delete from beginning
4. Delete from end
5. Delete by value
6. Search for an item
7. Display list
8. Exit
Enter your choice: 7
Doubly Linked List: 20 <-> 30 <-> NULL

--- DOUBLY LINKED LIST MENU ---
1. Insert at beginning
2. Insert at end
3. Delete from beginning
4. Delete from end
5. Delete by value
6. Search for an item
7. Display list
8. Exit
Enter your choice: 6
Enter value to search: 30
Item 30 found at position 2.
```

}



```
--- DOUBLY LINKED LIST MENU ---
1. Insert at beginning
2. Insert at end
3. Delete from beginning
4. Delete from end
5. Delete by value
6. Search for an item
7. Display list
8. Exit
Enter your choice: 3
Deleted 20 from beginning.

--- DOUBLY LINKED LIST MENU ---
1. Insert at beginning
2. Insert at end
3. Delete from beginning
4. Delete from end
5. Delete by value
6. Search for an item
7. Display list
8. Exit
Enter your choice: 4
Deleted 30 from end.

--- DOUBLY LINKED LIST MENU ---
1. Insert at beginning
2. Insert at end
3. Delete from beginning
4. Delete from end
5. Delete by value
6. Search for an item
7. Display list
8. Exit
Enter your choice: 5
Enter value to delete: 30
List is empty.

--- DOUBLY LINKED LIST MENU ---
1. Insert at beginning
2. Insert at end
3. Delete from beginning
4. Delete from end
5. Delete by value
6. Search for an item
7. Display list
8. Exit
Enter your choice: 8
Exiting...

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