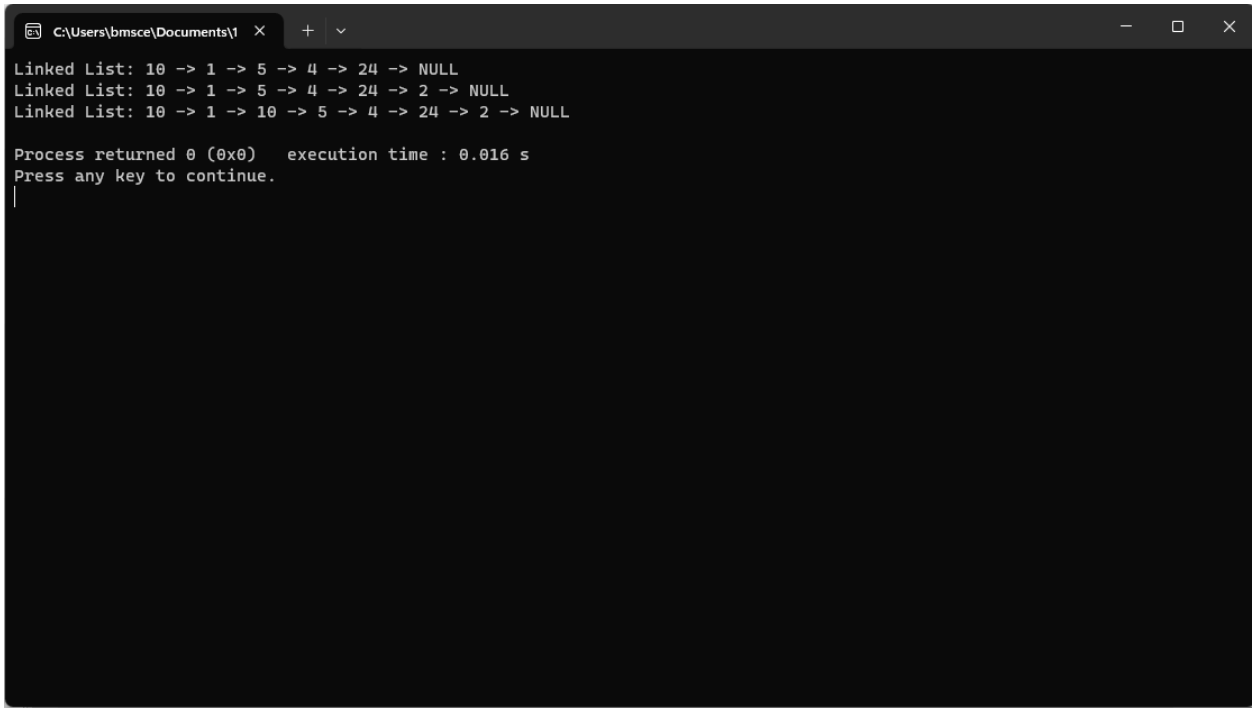
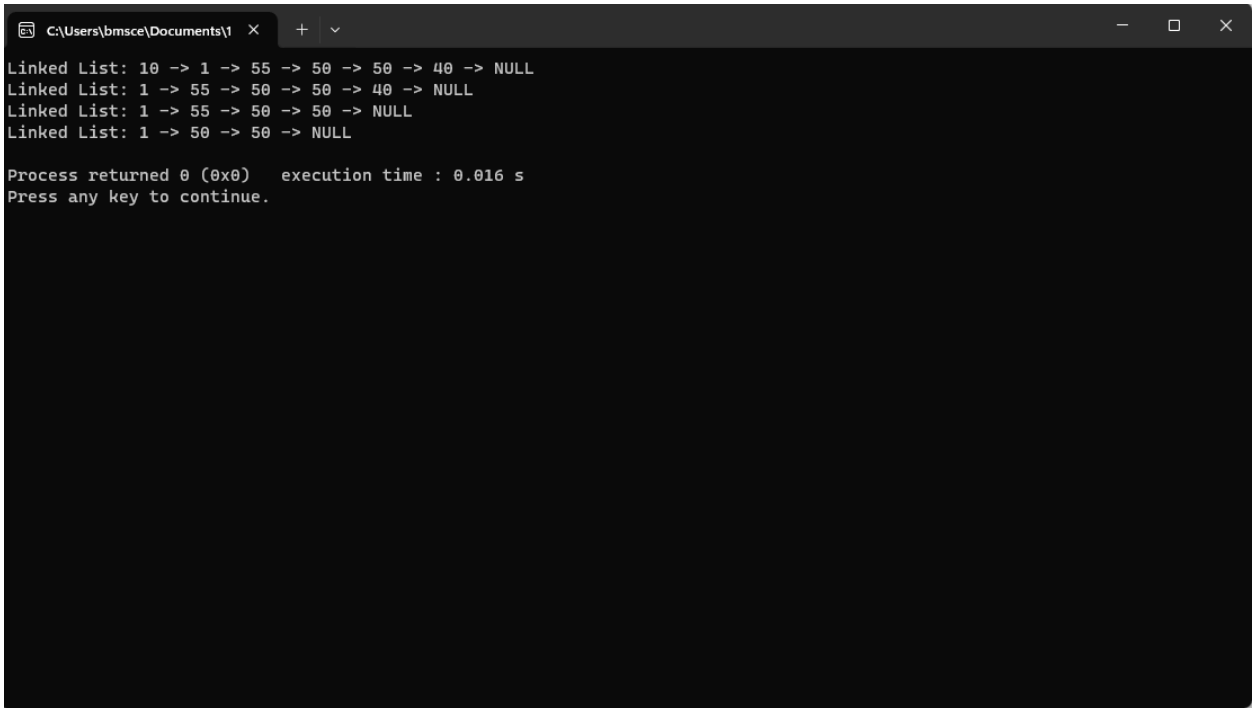



```
#include <stdio.h>
#include <stdlib.h>

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

struct Node* create_list() {
    return NULL;
}

    struct Node* create_node(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("allocation failed\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

struct Node* insertAtFirst(struct Node* head, int data) {
    struct Node* newNode = create_node(data);
    newNode->next = head;
    return newNode;
```

```
}

struct Node* insertAtPosition(struct Node* head, int data, int position) {
    if (position < 1) {
        printf("Invalid position\n");
        return head;
    }

    struct Node* newNode = create_node(data);

    if (position == 1) {
        newNode->next = head;
        return newNode;
    }

    struct Node* temp = head;
    for (int i = 1; i < position - 1 && temp != NULL; ++i) {
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Invalid position\n");
        return head;
    }

    newNode->next = temp->next;
    temp->next = newNode;

    return head;
}


struct Node* insertAtEnd(struct Node* head, int data) {
    struct Node* newNode = create_node(data);

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

    struct Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
```

```
    temp->next = newNode;
    return head;
}


void display_list(struct Node* head) {
    printf("Linked List: ");
    while (head != NULL) {
        printf("%d -> ", head->data);
        head = head->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = create_list();

    head = insertAtFirst(head, 5);
    head = insertAtEnd(head, 4);
    head = insertAtFirst(head, 1);
    head = insertAtEnd(head,24);
    head = insertAtFirst(head,10);

    display_list(head);

    head = insertAtEnd(head, 2);


    display_list(head);

    head = insertAtPosition(head, 10, 3);

    display_list(head);

    return 0;
}
```




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

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

struct Node* create_list() {
    return NULL;
}

struct Node* create_node(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Allocation failed\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
struct Node* insertAtPosition(struct Node* head, int data, int position) {
    if (position < 1) {
        printf("Invalid position\n");
        return head;
```

```
    }

    struct Node* newNode = create_node(data);

    if (position == 1) {
        newNode->next = head;
        return newNode;
    }

    struct Node* temp = head;
    for (int i = 1; i < position - 1 && temp != NULL; ++i) {
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Invalid position\n");
        return head;
    }

    newNode->next = temp->next;
    temp->next = newNode;

    return head;
}
struct Node* deleteAtBeginning(struct Node* head) {
    if (head == NULL) {
        printf("List is empty\n");
        return NULL;
    }

    struct Node* temp = head;
    head = head->next;
    free(temp);

    return head;
}

struct Node* deleteAtEnd(struct Node* head) {
    if (head == NULL) {
        printf("List is empty\n");
        return NULL;
    }

    if (head->next == NULL) {
```

```
        free(head);
        return NULL;
    }

    struct Node* temp = head;
    while (temp->next->next != NULL) {
        temp = temp->next;
    }

    free(temp->next);
    temp->next = NULL;

    return head;
}

struct Node* deleteAtPosition(struct Node* head, int position) {
    if (head == NULL || position < 1) {
        printf("Invalid position\n");
        return head;
    }

    if (position == 1) {
        struct Node* temp = head;
        head = head->next;
        free(temp);
        return head;
    }

    struct Node* temp = head;
    for (int i = 1; i < position - 1 && temp != NULL; ++i) {
        temp = temp->next;
    }

    if (temp == NULL || temp->next == NULL) {
        printf("Invalid position\n");
        return head;
    }

    struct Node* nodeToDelete = temp->next;
    temp->next = nodeToDelete->next;
    free(nodeToDelete);

    return head;
}
```

```
void display_list(struct Node* head) {
    printf("Linked List: ");
    while (head != NULL) {
        printf("%d -> ", head->data);
        head = head->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = create_list();
    head = insertAtPosition(head, 40, 1);
    head = insertAtPosition(head, 50, 1);
    head = insertAtPosition(head, 50, 1);
    head = insertAtPosition(head, 10, 1);
    head = insertAtPosition(head, 1, 2);
    head = insertAtPosition(head, 55, 3);
    display_list(head);

    head = deleteAtBeginning(head);
    display_list(head);

    head = deleteAtEnd(head);
    display_list(head);

    deleteAtPosition(head, 2);
    display_list(head);
    return 0;
}
```