INSERTION AND DELETION IN SINGLY LINKED LIST:

```
//SINGLY LINKED LIST
#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);
    if (head == NULL) {
        head = newNode;
    } else {
        newNode->next = head;
        head->prev = newNode;
        head = newNode;
    }
    printf("Node inserted at the beginning.\n");
}

void insertAtEnd(int data) {
    struct Node* newNode = createNode(data);
    if (head == NULL) {
        head = newNode;
    } else {
        struct Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
    printf("Node inserted at the end.\n");
}

void insertAtPosition(int data, int position) {
    struct Node* newNode = createNode(data);
    if (position == 1) {
        insertAtBeginning(data);
        return;
    }
    struct Node* temp = head;
    for (int i = 1; i < position - 1; i++) {
        if (temp == NULL) {
            printf("Position out of range.\n");
            return;
        }
        temp = temp->next;
    }
    if (temp->next == NULL) {
        insertAtEnd(data);
    } else {
        newNode->next = temp->next;
        newNode->prev = temp;
        temp->next->prev = newNode;
        temp->next = newNode;
        printf("Node inserted at position %d.\n", position);
    }
}

void deleteFromBeginning() {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    struct Node* temp = head;
    if (head->next == NULL) {
        head = NULL;
    } else {
        head = head->next;
        head->prev = NULL;
    }
    free(temp);
    printf("Node deleted from the beginning.\n");
}

void deleteFromEnd() {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    struct Node* temp = head;
    if (head->next == NULL) {
        head = NULL;
    } else {
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->prev->next = NULL;
    }
    free(temp);
    printf("Node deleted from the end.\n");
}

void deleteFromPosition(int position) {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    struct Node* temp = head;
    if (position == 1) {
        deleteFromBeginning();
        return;
    }
    for (int i = 1; i < position; i++) {
        if (temp == NULL) {
            printf("Position out of range.\n");
            return;
        }
        temp = temp->next;
    }
    if (temp->next == NULL) {
        deleteFromEnd();
    } else {
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;
        free(temp);
        printf("Node deleted from position %d.\n", position);
    }
}

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("\n");
}

int main() {
    int choice, data, position;
    while (1) {
        printf("\nDoubly Linked List Menu:\n");
        printf("1. Insert at Beginning\n");
        printf("2. Insert at End\n");
        printf("3. Insert at Position\n");
        printf("4. Delete from Beginning\n");
        printf("5. Delete from End\n");
        printf("6. Delete from Position\n");
        printf("7. Display\n");
        printf("8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter data: ");
                scanf("%d", &data);
                insertAtBeginning(data);
                break;
            case 2:
                printf("Enter data: ");
                scanf("%d", &data);
                insertAtEnd(data);
                break;
            case 3:
                printf("Enter data: ");
                scanf("%d", &data);
                printf("Enter position: ");
                scanf("%d", &position);
                insertAtPosition(data, position);
                break;
            case 4:
                deleteFromBeginning();
                break;
            case 5:
                deleteFromEnd();
                break;
            case 6:
                printf("Enter position: ");
                scanf("%d", &position);
                deleteFromPosition(position);
                break;
            case 7:
                display();
                break;
            case 8:
                exit(0);
            default:
                printf("Invalid choice.\n");
        }
    }
    return 0;
}
```

INSERTION AND DELETION IN DOUBLY LINKED LIST:

```
//Doubly linked list

#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);
    if (head == NULL) {
        head = newNode;
    } else {
        newNode->next = head;
        head->prev = newNode;
        head = newNode;
    }
    printf("Node inserted at the beginning.\n");
}

void insertAtEnd(int data) {
    struct Node* newNode = createNode(data);
    if (head == NULL) {
        head = newNode;
    } else {
        struct Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
    printf("Node inserted at the end.\n");
}

void insertAtPosition(int data, int position) {
    struct Node* newNode = createNode(data);
    if (position == 1) {
        insertAtBeginning(data);
        return;
    }
    struct Node* temp = head;
    for (int i = 1; i < position - 1; i++) {
        if (temp == NULL) {
            printf("Position out of range.\n");
            return;
        }
        temp = temp->next;
    }
    if (temp->next == NULL) {
        insertAtEnd(data);
    } else {
        newNode->next = temp->next;
        newNode->prev = temp;
        temp->next->prev = newNode;
        temp->next = newNode;
        printf("Node inserted at position %d.\n", position);
    }
}
```

```
void deleteFromBeginning() {
   if (head == NULL) {
      printf("List is empty.\n");
      return;
   }
   struct Node* temp = head;
   if (head->next == NULL) {
      head = NULL;
   } else {
      head = head->next;
      head->prev = NULL;
   }
   free(temp);
   printf("Node deleted from the beginning.\n");
}

void deleteFromEnd() {
   if (head == NULL) {
      printf("List is empty.\n");
      return;
   }
   struct Node* temp = head;
   if (head->next == NULL) {
      head = NULL;
   } else {
      while (temp->next != NULL) {
         temp = temp->next;
      }
      temp->prev->next = NULL;
   }
   free(temp);
   printf("Node deleted from the end.\n");
}

void deleteFromPosition(int position) {
   if (head == NULL) {
      printf("List is empty.\n");
      return;
   }
   struct Node* temp = head;
   if (position == 1) {
      deleteFromBeginning();
      return;
   }
   for (int i = 1; i < position; i++) {
      if (temp == NULL) {
         printf("Position out of range.\n");
```

```
            return;
        }
        temp = temp->next;
    }
    if (temp->next == NULL) {
        deleteFromEnd();
    } else {
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;
        free(temp);
        printf("Node deleted from position %d.\n", position);
    }
}

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("\n");
}

int main() {
    int choice, data, position;
    while (1) {
        printf("\nDoubly Linked List Menu:\n");
        printf("1. Insert at Beginning\n");
        printf("2. Insert at End\n");
        printf("3. Insert at Position\n");
        printf("4. Delete from Beginning\n");
        printf("5. Delete from End\n");
        printf("6. Delete from Position\n");
        printf("7. Display\n");
        printf("8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter data: ");
                scanf("%d", &data);
                insertAtBeginning(data);
                break;
            case 2:
                printf("Enter data: ");
                scanf("%d", &data);
                insertAtEnd(data);
                break;
            case 3:
                printf("Enter data: ");
                scanf("%d", &data);
                printf("Enter position: ");
                scanf("%d", &position);
                insertAtPosition(data, position);
                break;
            case 4:
                deleteFromBeginning();
                break;
            case 5:
                deleteFromEnd();
                break;
            case 6:
                printf("Enter position: ");
                scanf("%d", &position);
                deleteFromPosition(position);
                break;
            case 7:
                display();
                break;
            case 8:
                exit(0);
            default:
                printf("Invalid choice.\n");
        }
    }
    return 0;
}
```