```c
//Insertion and Deletion in Single Linked List
#include <stdio.h>
#include <stdlib.h>

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

struct Node* head = NULL;

void insertAtBeginning(int num)
{
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    new_node->data = num;
    new_node->next = head;
    head = new_node;
    printf("%d inserted at beginning of list\n", num);
}

void insertAtEnd(int num)
{
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    new_node->data = num;
    new_node->next = NULL;

    if (head == NULL)
    {
        head = new_node;
    }
    else
    {
        struct Node* current = head;
        while (current->next != NULL)
        {
            current = current->next;
        }
        current->next = new_node;
    }
    printf("%d inserted at end of list\n", num);
}

void insertAtPosition(int num, int position)
{
    if (position < 1)
    {
        printf("Invalid position\n");
        return;
    }
    if (position == 1)
    {
        insertAtBeginning(num);
        return;
    }

    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    new_node->data = num;

    struct Node* current = head;
    struct Node* previous = NULL;
    int count = 1;

    while (current != NULL && count < position) {
        previous = current;
        current = current->next;
        count++;
    }

    if (count < position)
    {
        printf("Position exceeds length of list\n");
        free(new_node);
        return;
    }

    previous->next = new_node;
    new_node->next = current;
    printf("%d inserted at position %d\n", num, position);
}

void deleteAtBeginning()
{
    if (head == NULL)
    {
        printf("List is empty\n");
        return;
    }
    struct Node* temp = head;
    head = head->next;
    printf("%d deleted from beginning of list\n", temp->data);
    free(temp);
}

void deleteAtEnd()
{
    if (head == NULL)
    {
        printf("List is empty\n");
        return;
    }
    if (head->next == NULL)
    {
        printf("%d deleted from end of list\n", head->data);
        free(head);
        head = NULL;
        return;
    }

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

    printf("%d deleted from end of list\n", current->data);
    free(current);
    previous->next = NULL;
}

void deleteAtPosition(int position)
{
    if (position < 1)
    {
        printf("Invalid position\n");
        return;
    }
    if (position == 1)
    {
        deleteAtBeginning();
        return;
    }

    struct Node* current = head;
    struct Node* previous = NULL;
    int count = 1;

    while (current != NULL && count < position) {
        previous = current;
        current = current->next;
        count++;
    }

    if (count < position || current == NULL)
    {
        printf("Position %d exceeds length of list\n", position);
        return;
    }

    printf("%d deleted from position %d\n", current->data, position);
    previous->next = current->next;
    free(current);
}

void display()
{
    if (head == NULL)
    {
        printf("List is empty\n");
        return;
    }

    struct Node* current = head;
    printf("List: ");
    while (current != NULL)
    {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}
int main()
{
    insertAtBeginning(3);
    insertAtEnd(5);
    insertAtBeginning(7);
    insertAtPosition(9,2);

    display();

    deleteAtPosition(2);

    display();

    deleteAtPosition(4);

    display();

    return 0;
}
```