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

#define MAX_TEXT 100

struct Node {
char text[MAX_TEXT];
struct Node* next;
};


struct Node* createNode(const char* line) {
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
if (newNode == NULL) {
printf("Memory allocation failed!\n");
exit(1);
}
strcpy(newNode->text, line);
newNode->next = NULL;
return newNode;
}

void insertFront(struct Node** head, const char* line) {
struct Node* node = createNode(line);
node->next = *head;
*head = node;
}

void insertEnd(struct Node** head, const char* line) {
struct Node* node = createNode(line);
if (*head == NULL) {
*head = node;
return;
}
struct Node* temp = *head;
while (temp->next != NULL)
temp = temp->next;
temp->next = node;
}

void insertMiddle(struct Node** head, const char* line, int pos) {
if (pos <= 1 || *head == NULL) {
insertFront(head, line);
return;
}
struct Node* temp = *head;
for (int i = 1; temp->next != NULL && i < pos - 1; i++)
temp = temp->next;

struct Node* node = createNode(line);
node->next = temp->next;
temp->next = node;
}
```

```
void deleteFront(struct Node** head) {
if (*head == NULL) {
printf("List is empty.\n");
return;
}
struct Node* temp = *head;
*head = (*head)->next;
free(temp);
}

void deleteEnd(struct Node** head) {
if (*head == NULL) {
printf("List is empty.\n");
return;
}
if ((*head)->next == NULL) {
free(*head);
*head = NULL;
return;
}
struct Node* temp = *head;
while (temp->next->next != NULL)
temp = temp->next;
free(temp->next);
temp->next = NULL;
}

void deleteMiddle(struct Node** head, int pos) {
if (*head == NULL) {
printf("List is empty.\n");
return;
}
if (pos <= 1) {
deleteFront(head);
return;
}
struct Node* temp = *head;
for (int i = 1; temp->next != NULL && i < pos - 1; i++)
temp = temp->next;

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

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

void display(struct Node* head) {
if (head == NULL) {
printf("List is empty.\n");
```

```
return;
}
printf("List contents:\n");
struct Node* temp = head;
while (temp != NULL) {
printf("%s\n", temp->text);
temp = temp->next;
}
}

int main() {
struct Node* head = NULL;
int choice, pos;
char line[MAX_TEXT];

while (1) {
printf("\n----- LINKED LIST MENU -----\n");
printf("1. Insert at Front\n");
printf("2. Insert at End\n");
printf("3. Insert at Position\n");
printf("4. Delete from Front\n");
printf("5. Delete from End\n");
printf("6. Delete from Position\n");
printf("7. Display List\n");
printf("8. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
getchar();  // clear newline from buffer

switch (choice) {
case 1:
printf("Enter text: ");
fgets(line, MAX_TEXT, stdin);
line[strcspn(line, "\n")] = '\0';
insertFront(&head, line);
break;

case 2:
printf("Enter text: ");
fgets(line, MAX_TEXT, stdin);
line[strcspn(line, "\n")] = '\0';
insertEnd(&head, line);
break;

case 3:
printf("Enter text: ");
fgets(line, MAX_TEXT, stdin);
line[strcspn(line, "\n")] = '\0';
printf("Enter position: ");
scanf("%d", &pos);
insertMiddle(&head, line, pos);
break;

case 4:
```

```
deleteFront(&head);
break;

case 5:
deleteEnd(&head);
break;

case 6:
printf("Enter position: ");
scanf("%d", &pos);
deleteMiddle(&head, pos);
break;

case 7:
display(head);
break;

case 8:
printf("Exiting program.\n");
exit(0);

default:
printf("Invalid choice! Try again.\n");
}
}

return 0;
}
```