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

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

struct Node *header = NULL;

struct Node *createNode(int data) {
   struct Node *newnode;
   newnode = malloc(sizeof(struct Node));
   newnode->data = data;
   newnode->link = NULL;
   return newnode;
}
```

```
void insertAtFront(int data) {
    struct Node *newnode = createNode(data);
    if (header == NULL) {
        header = newnode;
    } else {
        newnode->link = header;
        header = newnode;
    }
}

void insertAtEnd(int data) {
    struct Node *newnode = createNode(data);
    if (header == NULL) {
        header = newnode;
    } else {
        struct Node *current = header;
        while (current->link != NULL) {
            current = current->link;
        }
        current->link = newnode;
    }
}

void insertAtAny(int data, int position) {
    struct Node *newnode = createNode(data);
    if (position == 1) {
        newnode->link = header;
        header = newnode;
    } else {
        struct Node *current = header;
        int i;
        for (i = 1; i < position - 1 && current != NULL; i++) {
            current = current->link;
        }
        if (current == NULL) {
            printf("Position out of bounds. Inserting at the end.\n");
            insertAtEnd(data);
        } else {
            newnode->link = current->link;
            current->link = newnode;
        }
    }
}

void deleteAtFront() {
    if (header == NULL) {
        printf("List is empty, nothing to delete.\n");
        return;
    }
    struct Node *temp = header;
    printf("Node with value %d deleted from the front.\n", temp->data);
    header = header->link;
    free(temp);
}
```

```c
void deleteAtEnd() {
    if (header == NULL) {
        printf("List is empty, nothing to delete.\n");
        return;
    }
    if (header->link == NULL) {
        printf("Node with value %d deleted from the end.\n", header->data);
        free(header);
        header = NULL;
    } else {
        struct Node *current = header;
        while (current->link->link != NULL) {
            current = current->link;
        }
        printf("Node with value %d deleted from the end.\n", current->link->data);
        free(current->link);
        current->link = NULL;
    }
}

void deleteAtAny(int position) {
    if (header == NULL) {
        printf("List is empty, nothing to delete.\n");
        return;
    }
    struct Node *current = header;
    struct Node *prev = NULL;
    int i;
    for (i = 1; i < position && current != NULL; i++) {
        prev = current;
        current = current->link;
    }
    if (current == NULL) {
        printf("Position out of bounds. Nothing to delete.\n");
        return;
    }
    prev->link = current->link;
    printf("Node with value %d deleted at position %d.\n", current->data, position);
    free(current);
}

void search(int key) {
    struct Node *current = header;
    int position = 1;
    while (current != NULL) {
        if (current->data == key) {
            printf("Value %d found at position %d.\n", key, position);
            return;
        }
        current = current->link;
        position++;
    }
    printf("Value %d not found in the list.\n", key);
}
```

```
void traversal() {
    struct Node *ptr = header;
    while (ptr != NULL) {
        printf("%d ", ptr->data);
        ptr = ptr->link;
    }
    printf("\n");
}

int main() {
    int choice, data, position;

    while (1) {
        printf("\nMenu:\n");
        printf("1. Insert at Front\n");
        printf("2. Insert at End\n");
        printf("3. Insert at Any Position\n");
        printf("4. Delete at Front\n");
        printf("5. Delete at End\n");
        printf("6. Delete at Any Position\n");
        printf("7. Search in List\n");
        printf("8. Display List\n");
        printf("9. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
switch (choice) {
    case 1:
        printf("Enter data to insert at front: ");
        scanf("%d", &data);
        insertAtFront(data);
        break;
    case 2:
        printf("Enter data to insert at end: ");
        scanf("%d", &data);
        insertAtEnd(data);
        break;
    case 3:
        printf("Enter the position to insert: ");
        scanf("%d", &position);
        printf("Enter the data to insert: ");
        scanf("%d", &data);
        insertAtAny(data, position);
        break;
    case 4:
        deleteAtFront();
        break;
    case 5:
        deleteAtEnd();
        break;
    case 6:
        printf("Enter the position to delete: ");
        scanf("%d", &position);
        deleteAtAny(position);
        break;
        case 7:
            printf("Enter the value to search: ");
            scanf("%d", &data);
            search(data);
            break;
         case 8:
            printf("Current List: ");
            traversal();
            break;
         case 9:
            exit(0);
         default:
            printf("Invalid choice. Please try again.\n");
      }
   }
   return 0;
}
```