Program:-

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

struct Node {
    int data;
    struct Node *next;
};
struct Node *head = NULL;
void createList(int n) {
    int data, i;
    if (n <= 0) {
        head = NULL;
        return;
    }
    struct Node *newNode;
    struct Node *temp;

    newNode = (struct Node*)malloc(sizeof(struct
Node));
    printf("Enter the 1st node: ");
    scanf("%d", &data);
    newNode->data = data;
    newNode->next = NULL;
    head = newNode;
    temp = head;

    for (i = 1; i < n; i++) {
        newNode = (struct Node*)malloc(sizeof(struct
Node));
        printf("Enter the %d node: ", i + 1);
        scanf("%d", &data);
        newNode->data = data;
        newNode->next = NULL;
        temp->next = newNode;
        temp = newNode;
    }
}
void insertAtBeginning(int val) {
    struct Node *newNode = (struct
Node*)malloc(sizeof(struct Node));
    newNode->data = val;
    newNode->next = head;
    head = newNode;
}
void insertAtEnd(int val) {
    struct Node *newNode = (struct
Node*)malloc(sizeof(struct Node));
    newNode->data = val;
    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
        return;
    }
    struct Node *temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}
void insertAtPosition(int pos, int val) {
    struct Node *newNode = (struct
Node*)malloc(sizeof(struct Node));
    newNode->data = val;

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

    struct Node *temp = head;
    int i;
    for (i = 1; temp != NULL && i < pos - 1; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Position out of range\n");
        free(newNode);
        return;
    }
    newNode->next = temp->next;
    temp->next = newNode;
}
void deleteAtBeginning() {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node *temp = head;
    head = head->next;
    free(temp);
}
void deleteAtEnd() {
    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 deleteAtPosition(int pos) {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node *temp = head;
```

```
 if (pos == 1) {
        head = head->next;
        free(temp);
        return;
    }

    struct Node *prev = NULL;
    int i;
    for (i = 1; temp != NULL && i < pos; i++) {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Position out of range\n");
        return;
    }
    prev->next = temp->next;
    free(temp);
}
void traverse() {
    struct Node *temp = head;
    if (temp == NULL) {
        printf("List is empty\n");
        return;
    }
    printf("Linked List: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
void search(int key) {
    struct Node *temp = head;
    int pos = 1;
    while (temp != NULL) {
        if (temp->data == key) {
            printf("%d found at position %d\n", key,
pos);
            return;
        }
        temp = temp->next;
        pos++;
    }
    printf("Element not found\n");
}

int main() {
    int ch, val, pos, key, n;
    do {
        printf("1.Create List\n2.Insert at
beginning\n3.Insert at end\n4.Insert at
position\n5.Delete at beginning\n6.Delete at
end\n7.Delete at
position\n8.Traverse\n9.Search\n10.Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);

        switch (ch) {
case 1:
            printf("Enter number of nodes: ");
            scanf("%d", &n);
            createList(n);
            break;
        case 2:
            printf("Enter value: ");
            scanf("%d", &val);
            insertAtBeginning(val);
            break;
        case 3:
            printf("Enter value: ");
            scanf("%d", &val);
            insertAtEnd(val);
            break;
        case 4:
            printf("Enter position: ");
            scanf("%d", &pos);
            printf("Enter value: ");
            scanf("%d", &val);
            insertAtPosition(pos, val);
            break;
        case 5:
            deleteAtBeginning();
            break;
        case 6:
            deleteAtEnd();
            break;
        case 7:
            printf("Enter position: ");
            scanf("%d", &pos);
            deleteAtPosition(pos);
            break;
        case 8:
            traverse();
            break;
        case 9:
            printf("Enter value to search: ");
            scanf("%d", &key);
            search(key);
            break;
        case 10:
            printf("Exiting program...\n");
            exit(0);
        default:
            printf("Invalid choice\n");
      }
    } while (ch != 10);
    return 0;
}
```

Output:-

```
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 1
Enter number of nodes: 3
Enter the 1st node: 23
Enter the 2 node: 45
Enter the 3 node: 67
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 8
Linked List: 23 -> 45 -> 67 -> NULL
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 2
Enter value: 33
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 8
Linked List: 33 -> 23 -> 45 -> 67 -> NULL
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 3
Enter value: 49
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 8
Linked List: 33 -> 23 -> 45 -> 67 -> 49 -> NULL
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 4
Enter position: 2
Enter value: 22
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 8
Linked List: 33 -> 22 -> 23 -> 45 -> 67 -> 49 ->
NULL
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
```

```
Enter your choice: 5
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 8
Linked List: 22 -> 23 -> 45 -> 67 -> 49 -> NULL
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 6
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 8
Linked List: 22 -> 23 -> 45 -> 67 -> NULL
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 7
Enter position: 3
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 8
Linked List: 22 -> 23 -> 67 -> NULL
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 9
Enter value to search: 23
23 found at position 2
1.Create List
2.Insert at beginning
3.Insert at end
4.Insert at position
5.Delete at beginning
6.Delete at end
7.Delete at position
8.Traverse
9.Search
10.Exit
Enter your choice: 10
Exiting program...
```