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

// Function prototypes
void create();
void display();
void insert_begin();
void insert_end();
void insert_pos();
void delete_begin();
void delete_end();
void delete_pos();
void search();
void reverse();

// Structure definition
struct node {
   int data;
   struct node* next;
};

// Global head pointer
struct node* head = NULL;

int main() {
   int choice;
   while(1) {
      printf("\n*****\n");
```

```
printf("1. Create\n");
printf("2. Display\n");
printf("3. Insert Node at beginning\n");
printf("4. Insert Node in specific position\n");
printf("5. Insert Node at end of LinkedList\n");
printf("6. Delete Node at beginning\n");
printf("7. Delete Node at end\n");
printf("8. Delete Node at position\n");
printf("9. Search for a node\n");
printf("10. Reverse the linked list\n");
printf("0. ** To exit **\n");

printf("\nEnter your choice: ");
scanf("%d", &choice);
switch(choice) {
   case 1: create(); break;
   case 2: display(); break;
   case 3: insert_begin(); break;
   case 4: insert_pos(); break;
   case 5: insert_end(); break;
   case 6: delete_begin(); break;
   case 7: delete_end(); break;
   case 8: delete_pos(); break;
   case 9: search(); break;
   case 10: reverse(); break;
   case 0: exit(0);
   default: printf("\nWrong Choice"); break;
}
```

```
    }
}

// Function to create a new node
void create() {
    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));
    printf("Enter node data: ");
    scanf("%d", &temp->data);
    temp->next = NULL;
    if(head == NULL) {
        head = temp;
    } else {
        struct node* ptr = head;
        while(ptr->next != NULL) {
            ptr = ptr->next;
        }
        ptr->next = temp;
    }
}

// Function to display the linked list
void display() {
    if(head == NULL) {
        printf("Linked List is Empty\n");
        return;
    }
    printf("LinkedList: ");
```

```
    struct node* ptr = head;
    while(ptr != NULL) {
        printf("%d -> ", ptr->data);
        ptr = ptr->next;
    }
    printf("NULL\n");
}

// Function to insert node at the beginning
void insert_begin() {
    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));
    printf("Enter node data: ");
    scanf("%d", &temp->data);
    temp->next = NULL;
    if(head == NULL) {
        head = temp;
        return;
    } else {
        temp->next = head;
        head = temp;
    }
}

// Function to insert node at a specific position
void insert_pos() {
    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));
```

```
    printf("Enter node data: ");
    scanf("%d", &temp->data);
    temp->next = NULL;
    if(head == NULL) {
        head = temp;
        return;
    } else {
        struct node* prev_ptr;
        struct node* ptr = head;
        int pos;
        printf("Enter position: ");
        scanf("%d", &pos);
        for(int i = 1; i < pos; i++) {
            prev_ptr = ptr;
            ptr = ptr->next;
        }
        temp->next = ptr;
        prev_ptr->next = temp;
    }
}

// Function to insert node at the end
void insert_end() {
    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));
    printf("Enter node data: ");
    scanf("%d", &temp->data);
    temp->next = NULL;
```

```
    if(head == NULL) {
        head = temp;
        return;
    } else {
        struct node* ptr = head;
        while(ptr->next != NULL) {
            ptr = ptr->next;
        }
        ptr->next = temp;
    }
}

// Function to delete the first node
void delete_begin() {
    if(head == NULL) {
        printf("Linked List is empty | Nothing to
delete \n");
        return;
    } else {
        struct node* ptr = head;
        head = head->next;
        free(ptr);
        printf("Node Deleted \n");
    }
}

// Function to delete the last node
void delete_end() {
```

```
    if(head == NULL) {
        printf("Linked List is empty | Nothing to
delete \n");
        return;
    } else if(head->next == NULL) {
        struct node* ptr = head;
        head = ptr->next;
        free(ptr);
    } else {
        struct node* ptr = head;
        struct node* prev_ptr = NULL;
        while(ptr->next != NULL) {
            prev_ptr = ptr;
            ptr = ptr->next;
        }
        prev_ptr->next = NULL;
        free(ptr);
    }
}

// Function to delete a node at a given position
void delete_pos() {
    int pos;
    printf("Enter node position to delete: ");
    scanf("%d", &pos);
    struct node* ptr = head;
    if(head == NULL) {
        printf("Linked List is empty \n");
```

```
        return;
    } else if(pos == 0) {
        ptr = head;
        head = ptr->next;
        free(ptr);
    } else {
        struct node* prev_ptr;
        for(int i = 1; i < pos; i++) {
            prev_ptr = ptr;
            ptr = ptr->next;
        }
        prev_ptr->next = ptr->next;
        free(ptr);
    }
}

// Function to search for a node
void search() {
    int value, pos = 0;
    printf("Enter value to search: ");
    scanf("%d", &value);
    struct node* ptr = head;
    while(ptr != NULL) {
        if(ptr->data == value) {
            printf("Node with value %d found at
position %d\n", value, pos);
            return;
        }
```

```
        pos++;
        ptr = ptr->next;
    }
    printf("Node with value %d not found\n", value);
}

// Function to reverse the linked list
void reverse() {
    struct node *prev = NULL, *current = head,
*next = NULL;
    while(current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    head = prev;
    printf("Linked List Reversed\n");
}
```