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

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

struct node *head = NULL;

void beginsert();
void lastinsert();
void randominsert();
void begin_delete();
void last_delete();
void random_delete();
void display();
void search();

int main() {
   int choice = 0;
   while (choice != 9) {
      printf("\n\n*********Main Menu*********\n");
      printf("Choose one option from the following list ...\n");
      printf("===============================================\n");
      printf("1. Insert in beginning\n2. Insert at last\n3. Insert at any random
location\n");
      printf("4. Delete from Beginning\n5. Delete from last\n6. Delete node after
specified location\n");
```

```
        printf("7. Search for an element\n8. Show\n9. Exit\n");
        printf("Enter your choice:\n");
        scanf("%d", &choice);

        switch (choice) {
            case 1: beginsert(); break;
            case 2: lastinsert(); break;
            case 3: randominsert(); break;
            case 4: begin_delete(); break;
            case 5: last_delete(); break;
            case 6: random_delete(); break;
            case 7: search(); break;
            case 8: display(); break;
            case 9: exit(0);
            default: printf("Please enter a valid choice...\n");
        }
    }
    return 0;
}

void beginsert() {
    struct node *ptr;
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL) {
        printf("\nOVERFLOW\n");
        return;
    }
```

```
    printf("Enter value:\n");
    scanf("%d", &item);
    ptr->data = item;
    ptr->next = head;
    head = ptr;
    printf("Node inserted at the beginning\n");
}

void lastinsert() {
    struct node *ptr, *temp;
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL) {
        printf("\nOVERFLOW\n");
        return;
    }

    printf("Enter value:\n");
    scanf("%d", &item);
    ptr->data = item;
    ptr->next = NULL;

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

```
    }
    printf("Node inserted at the end\n");
}

void randominsert() {
    int i, loc, item;
    struct node *ptr, *temp;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL) {
        printf("\nOVERFLOW\n");
        return;
    }

    printf("Enter element value:\n");
    scanf("%d", &item);
    ptr->data = item;

    printf("Enter the location after which you want to insert:\n");
    scanf("%d", &loc);

    temp = head;
    for (i = 0; i < loc; i++) {
        if (temp == NULL) {
            printf("\nCan't insert, location out of range\n");
            free(ptr);
            return;
        }
        temp = temp->next;
    }
```

```
    ptr->next = temp->next;
    temp->next = ptr;
    printf("Node inserted\n");
}

void begin_delete() {
    struct node *ptr;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }

    ptr = head;
    head = head->next;
    free(ptr);
    printf("Node deleted from the beginning\n");
}

void last_delete() {
    struct node *ptr, *ptr1;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    } else if (head->next == NULL) {
        free(head);
        head = NULL;
        printf("Only node deleted\n");
        return;
```

```
    }

    ptr = head;
    while (ptr->next != NULL) {
        ptr1 = ptr;
        ptr = ptr->next;
    }
    ptr1->next = NULL;
    free(ptr);
    printf("Node deleted from the last\n");
}

void random_delete() {
    struct node *ptr, *ptr1;
    int loc, i;

    printf("Enter the location of the node to delete:\n");
    scanf("%d", &loc);

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

    ptr = head;
    for (i = 0; i < loc; i++) {
        ptr1 = ptr;
        ptr = ptr->next;
        if (ptr == NULL) {
```

```
            printf("Can't delete, location out of range\n");
            return;
        }
    }

    ptr1->next = ptr->next;
    free(ptr);
    printf("Deleted node at location %d\n", loc + 1);
}

void search() {
    struct node *ptr;
    int item, i = 0, found = 0;

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

    printf("Enter item to search:\n");
    scanf("%d", &item);
    ptr = head;

    while (ptr != NULL) {
        if (ptr->data == item) {
            printf("Item found at location %d\n", i + 1);
            found = 1;
            break;
        }
```

```
        ptr = ptr->next;
        i++;
    }

    if (!found)
        printf("Item not found\n");
}

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

    printf("Linked list contents:\n");
    while (ptr != NULL) {
        printf("%d -> ", ptr->data);
        ptr = ptr->next;
    }
    printf("NULL\n");
}
```