Experiment No -5 Represent a node of a singly linked linear list. Implement the following functions. 1) Create a list 2) insert an element – at the beginning, at the end and at a specified position in the list 3) delete an element from the beginning, end or a specified position at the list 4) reverse the list 5) search for an element in the list. Create a menu-driven program to test all the functions.

CODE:

#include <stdio.h>

#include <stdlib.h>

struct Node {

    int data;

    struct Node\* next;

};

struct Node\* head = NULL;

void createList() {

    head = (struct Node\*)malloc(sizeof(struct Node));

    head->data = 0;

    head->next = NULL;

    printf(" List created successfully!\n");

}

void insertAtBeginning(int value) {

    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

    newNode->data = value;

    newNode->next = head;

    head = newNode;

    printf(" Element inserted at the beginning successfully!\n");

}

void insertAtEnd(int value) {

    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

    newNode->data = value;

    newNode->next = NULL;

    struct Node\* temp = head;

    while (temp->next != NULL) {

        temp = temp->next;

    }

    temp->next = newNode;

    printf(" Element inserted at the end successfully!\n");

}

void insertAtPosition(int value, int position) {

    if (position < 1) {

        printf(" Invalid position!\n");

        return;

    }

    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

    newNode->data = value;

    if (position == 1) {

        newNode->next = head;

        head = newNode;

        printf(" Element inserted at position %d successfully!\n", position);

        return;

    }

    struct Node\* temp = head;

    for (int i = 1; i < position-1; i++) {

        if (temp->next == NULL) {

            printf(" Invalid position!\n");

            return;

        }

        temp = temp->next;

    }

    newNode->next = temp->next;

    temp->next = newNode;

    printf(" Element inserted at position %d successfully!\n", position);

}

void deleteFromBeginning() {

    if (head == NULL) {

        printf(" List is empty!\n");

        return;

    }

    struct Node\* temp = head;

    head = head->next;

    free(temp);

    printf(" Element deleted from the beginning successfully!\n");

}

void deleteFromEnd() {

    if (head == NULL) {

        printf(" List is empty!\n");

        return;

    }

    if (head->next == NULL) {

        free(head);

        head = NULL;

        printf(" Element deleted from the end successfully!\n");

        return;

    }

    struct Node\* temp = head;

    while (temp->next->next != NULL) {

        temp = temp->next;

    }

    free(temp->next);

    temp->next = NULL;

    printf(" Element deleted from the end successfully!\n");

}

void deleteFromPosition(int position) {

    if (head == NULL) {

        printf(" List is empty!\n");

        return;

    }

    if (position < 1)

    {

printf(" Invalid position!\n");

return;

}

if (position == 1) {

struct Node\* temp = head;

head = head->next;

free(temp);

printf(" Element deleted from position %d successfully!\n", position);

return;

}

struct Node\* temp = head;

for (int i = 1; i < position-1; i++) {

if (temp->next == NULL) {

printf(" Invalid position!\n");

return;

}

temp = temp->next;

}

struct Node\* temp2 = temp->next;

temp->next = temp->next->next;

free(temp2);

printf(" Element deleted from position %d successfully!\n", position);

}

void reverseList() {

struct Node\* prev = NULL;

struct Node\* current = head;

struct Node\* next;

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

printf(" List reversed successfully!\n");

}

void searchForElement(int value) {

struct Node\* temp = head;

int position = 1;

while (temp != NULL) {

if (temp->data == value) {

printf(" Element %d found at position %d\n", value, position);

return;

}

temp = temp->next;

position++;

}

printf(" Element %d not found in the list!\n", value);

}

void displayList() {

struct Node\* temp = head;

if (temp == NULL) {

printf("List is empty!\n");

return;

}

printf("List elements: ");

while (temp != NULL) {

printf("%d ", temp->data);

temp = temp->next;

}

printf("\n");

}

int main() {

int choice, value, position;

printf("\nMenu:\n");

printf("1. Create List\n");

printf("2. Insert at beginning\n");

printf("3. Insert at end\n");

printf("4. Insert at position\n");

printf("5. Delete from beginning\n");

printf("6. Delete from end\n");

printf("7. Delete from position\n");

printf("8. Reverse List\n");

printf("9. Search for element\n");

printf("10. Display List\n");

printf("11. Exit\n");

while (1) {

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

createList();

break;

case 2:

printf(" Enter value to insert: ");

scanf("%d", &value);

insertAtBeginning(value);

break;

case 3:

printf(" Enter value to insert: ");

scanf("%d", &value);

insertAtEnd(value);

break;

case 4:

printf(" Enter value to insert: ");

scanf("%d", &value);

printf(" Enter position to insert: ");

scanf("%d", &position);

insertAtPosition(value, position);

break;

case 5:

deleteFromBeginning();

break;

case 6:

deleteFromEnd();

break;

case 7:

printf(" Enter position to delete: ");

scanf("%d", &position);

deleteFromPosition(position);

break;

case 8:

reverseList();

break;

case 9:

printf(" Enter value to search for: ");

scanf("%d", &value);

searchForElement(value);

break;

case 10:

displayList();

break;

case 11:

exit(0);

break;

default:

printf(" Invalid choice!\n");

break;

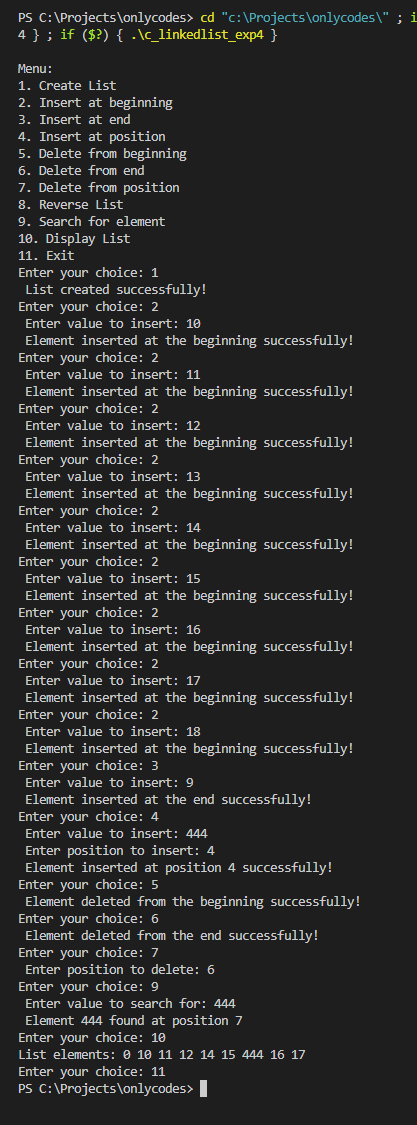
}

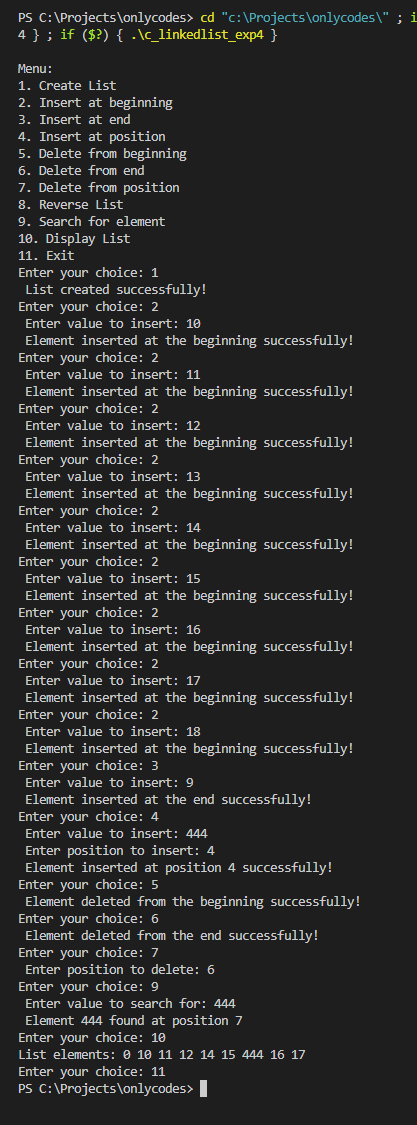
}

return 0;

}

**OUTPUT:**

****

****