LAB PROGRAM - 4

Write a program to implement Singly Linked List with following operations:

a) Create a linked list

b) Insertion of a node at first position, at any position and at end of list.

Display the contents of the linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* head = NULL;
void createList(int n) {
    struct Node *newNode, *temp;
    int data, i;
    if (n <= 0) {
        printf("Invalid size.\n");
        return;
    }

    head = (struct Node*)malloc(sizeof(struct Node));
    if (head == NULL) {
        printf("Memory allocation failed.\n");
```

```
        return;
    }

    printf("Enter data for node 1: ");
    scanf("%d", &data);
    head->data = data;
    head->next = NULL;
    temp = head;
    for (i = 2; i <= n; i++) {
        newNode = (struct Node*)malloc(sizeof(struct Node));
        if (newNode == NULL) {
            printf("Memory allocation failed.\n");
            return;
        }
        printf("Enter data for node %d: ", i);
        scanf("%d", &data);
        newNode->data = data;
        newNode->next = NULL;
        temp->next = newNode;
        temp = newNode;
    }
    printf("Linked list created successfully.\n");
}
```

```
void insertAtBeginning(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = head;
    head = newNode;
    printf("Node inserted at beginning.\n");
}
void insertAtEnd(int data) {
    struct Node *newNode, *temp;
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    if (head == NULL) {
        head = newNode;
    } else {
        temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }
    printf("Node inserted at end.\n");
}
```

```
void insertAtPosition(int data, int position) {
    struct Node *newNode, *temp;
    int i;
    if (position < 1) {
        printf("Invalid position.\n");
        return;
    }
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    if (position == 1) {
        newNode->next = head;
        head = newNode;
        printf("Node inserted at position 1.\n");
        return;
    }
    temp = head;
    for (i = 1; i < position - 1 && temp != NULL; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Position out of range.\n");
        free(newNode);
        return;
    }
```

```
    newNode->next = temp->next;
    temp->next = newNode;
    printf("Node inserted at position %d.\n", position);
}
void displayList() {
    struct Node* temp = head;
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    printf("Linked list contents: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
int main() {
    int choice, n, data, pos;
    while (1) {
        printf("\n--- Singly Linked List Menu ---\n");
        printf("1. Create linked list\n");
        printf("2. Insert at beginning\n");
```

```
printf("3. Insert at position\n");
printf("4. Insert at end\n");
printf("5. Display list\n");
printf("6. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
case 1:
    printf("Enter number of nodes: ");
    scanf("%d", &n);
    createList(n);
    break;
case 2:
    printf("Enter data to insert at beginning: ");
    scanf("%d", &data);
    insertAtBeginning(data);
    break;
case 3:
    printf("Enter position: ");
    scanf("%d", &pos);
    printf("Enter data to insert: ");
    scanf("%d", &data);
    insertAtPosition(data, pos);
```

```
            break;
        case 4:
            printf("Enter data to insert at end: ");
            scanf("%d", &data);
            insertAtEnd(data);
            break;
        case 5:
            displayList();
            break;
        case 6:
            printf("Exiting program.\n");
            exit(0);
        default:
            printf("Invalid choice.\n");
        }
    }

    return 0;
}
```

OUTPUT:

```
--- Singly Linked List Menu ---
1. Create linked list
2. Insert at beginning
3. Insert at position
4. Insert at end
5. Display list
6. Exit
Enter your choice: 1
Enter number of nodes: 3
Enter data for node 1: 22
Enter data for node 2: 33
Enter data for node 3: 44
Linked list created successfully.

--- Singly Linked List Menu ---
1. Create linked list
2. Insert at beginning
3. Insert at position
4. Insert at end
5. Display list
6. Exit
Enter your choice: 2
Enter data to insert at beginning: 11
Node inserted at beginning.

--- Singly Linked List Menu ---
1. Create linked list
2. Insert at beginning
3. Insert at position
4. Insert at end
5. Display list
6. Exit
Enter your choice: 3
Enter position: 2
Enter data to insert: 90
Node inserted at position 2.

--- Singly Linked List Menu ---
1. Create linked list
2. Insert at beginning
3. Insert at position
4. Insert at end
5. Display list
6. Exit
Enter your choice: 4
Enter data to insert at end: 89
Node inserted at end.

--- Singly Linked List Menu ---
1. Create linked list
2. Insert at beginning
3. Insert at position
4. Insert at end
5. Display list
6. Exit
Enter your choice: 5
Linked list contents: 11 -> 90 -> 22 -> 33 -> 44 -> 89 -> NULL
```

```
--- Singly Linked List Menu ---
1. Create linked list
2. Insert at beginning
3. Insert at position
4. Insert at end
5. Display list
6. Exit
Enter your choice: 6
Exiting program.

Process returned 0 (0x0)   execution time : 32.573 s
Press any key to continue.
```