Experiment 5

CODE:

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

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

struct Node* create() {
   struct Node* head = NULL;
   struct Node* temp;
   int data;
   char choice;

   do {
      printf("Enter data: ");
      scanf("%d", &data);

      struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
      newNode->data = data;
      newNode->next = NULL;

      if (head == NULL) {
         head = newNode;
      } else {
         temp->next = newNode;
      }
      temp = newNode;

      printf("Want to add more nodes? (y/n): ");
      scanf(" %c", &choice);
   } while (choice == 'y' || choice == 'Y');

   return head;
}
```

```
void display(struct Node* head) {
   struct Node* temp = head;

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

   printf("Linked List: ");
   while (temp != NULL) {
      printf("%d -> ", temp->data);
      temp = temp->next;
   }
   printf("NULL\n");
}

void insertAtBeginning(struct Node** head, int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = *head;
   *head = newNode;
}

void insertInMiddle(struct Node* head, int data, int pos) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   struct Node* temp = head;
   int i;

   newNode->data = data;
   newNode->next = NULL;

   for (i = 1; i < pos - 1 && temp != NULL; i++) {
      temp = temp->next;
   }

   if (temp == NULL) {
      printf("Position out of range\n");
      free(newNode);
   } else {
        newNode->next = temp->next;
        temp->next = newNode;
    }
}

void insertAtEnd(struct Node* head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    struct Node* temp = head;

    newNode->data = data;
    newNode->next = NULL;

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

void deleteFirst(struct Node** head) {
    if (*head == NULL) {
        printf("List is empty.\n");
        return;
    }

    struct Node* temp = *head;
    *head = (*head)->next;
    free(temp);
}

void deleteMiddle(struct Node* head, int pos) {
    struct Node* temp = head;
    struct Node* prev;
    int i;

    if (pos == 1) {
        printf("Use deleteFirst() to delete the first node.\n");
        return;
    }

    for (i = 1; i < pos && temp != NULL; i++) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Position out of range.\n");
    } else {
        prev->next = temp->next;
        free(temp);
    }
}

void deleteLast(struct Node* head) {
    struct Node* temp = head;
    struct Node* prev;

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

    if (head->next == NULL) {
        free(head);
        head = NULL;
        return;
    }

    while (temp->next != NULL) {
        prev = temp;
        temp = temp->next;
    }

    prev->next = NULL;
    free(temp);
```

```
}

void updateSLL(struct Node* head, int oldData, int newData) {
   struct Node* temp = head;

   while (temp != NULL) {
      if (temp->data == oldData) {
         temp->data = newData;
         printf("Data updated successfully.\n");
         return;
      }
      temp = temp->next;
   }
   printf("Data not found in the list.\n");
}

int countNodes(struct Node* head) {
   int count = 0;
   struct Node* temp = head;

   while (temp != NULL) {
      count++;
      temp = temp->next;
   }

   return count;
}

int main() {
   struct Node* head = NULL;
   int choice, data, pos, oldData, newData;

   do {
      printf("\nMenu:\n");
      printf("1. Create Linked List\n");
      printf("2. Display Linked List\n");
      printf("3. Insert at Beginning\n");
      printf("4. Insert in Middle\n");
      printf("5. Insert at End\n");
      printf("6. Delete First Node\n");
      printf("7. Delete Middle Node\n");
      printf("8. Delete Last Node\n");
      printf("9. Update Node Data\n");
      printf("10. Count Nodes\n");
      printf("11. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            head = create();
            break;
         case 2:
            display(head);
            break;
         case 3:
            printf("Enter data to insert at beginning: ");
            scanf("%d", &data);
            insertAtBeginning(&head, data);
            display(head);
            break;
         case 4:
            printf("Enter position to insert in middle: ");
            scanf("%d", &pos);
            printf("Enter data to insert: ");
            scanf("%d", &data);
            insertInMiddle(head, data, pos);
            display(head);
            break;
         case 5:
            printf("Enter data to insert at end: ");
            scanf("%d", &data);
            insertAtEnd(head, data);
            display(head);
            break;
         case 6:
            deleteFirst(&head);
            display(head);
```

```
                break;
            case 7:
                printf("Enter position to delete: ");
                scanf("%d", &pos);
                deleteMiddle(head, pos);
                display(head);
                break;
            case 8:
                deleteLast(head);
                display(head);
                break;
            case 9:
                printf("Enter old data to update: ");
                scanf("%d", &oldData);
                printf("Enter new data: ");
                scanf("%d", &newData);
                updateSLL(head, oldData, newData);
                display(head);
                break;
            case 10:
                printf("Number of nodes: %d\n", countNodes(head));
                display(head);
                break;
            case 11:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    } while (choice != 11);

    return 0;
}
```

Output:

```
Menu:
1. Create Linked List
2. Display Linked List
3. Insert at Beginning
4. Insert in Middle
5. Insert at End
6. Delete First Node
7. Delete Middle Node
8. Delete Last Node
9. Update Node Data
10. Count Nodes
11. Exit
Enter your choice: 1
Enter data: 12
Want to add more nodes? (y/n): y
Enter data: 13
Want to add more nodes? (y/n): y
Enter data: 14
Want to add more nodes? (y/n): y
Enter data: 15
Want to add more nodes? (y/n): y
Enter data: 16
Want to add more nodes? (y/n): n

Menu:
1. Create Linked List
2. Display Linked List
3. Insert at Beginning
4. Insert in Middle
5. Insert at End
6. Delete First Node
7. Delete Middle Node
8. Delete Last Node
9. Update Node Data
10. Count Nodes
11. Exit
Enter your choice: 2
Linked List: 12 -> 13 -> 14 -> 15 -> 16 -> NULL
```

```
Menu:
1. Create Linked List
2. Display Linked List
3. Insert at Beginning
4. Insert in Middle
5. Insert at End
6. Delete First Node
7. Delete Middle Node
8. Delete Last Node
9. Update Node Data
10. Count Nodes
11. Exit
Enter your choice: 3
Enter data to insert at beginning: 11
Linked List: 11 -> 12 -> 13 -> 14 -> 15 -> 16 -> NULL

Menu:
1. Create Linked List
2. Display Linked List
3. Insert at Beginning
4. Insert in Middle
5. Insert at End
6. Delete First Node
7. Delete Middle Node
8. Delete Last Node
9. Update Node Data
10. Count Nodes
11. Exit
Enter your choice: 4
Enter position to insert in middle: 4
Enter data to insert: 20
Linked List: 11 -> 12 -> 13 -> 20 -> 14 -> 15 -> 16 -> NULL

Menu:
1. Create Linked List
2. Display Linked List
3. Insert at Beginning
4. Insert in Middle
```

```
5. Insert at End
6. Delete First Node
7. Delete Middle Node
8. Delete Last Node
9. Update Node Data
10. Count Nodes
11. Exit
Enter your choice: 5
Enter data to insert at end: 17
Linked List: 11 -> 12 -> 13 -> 20 -> 14 -> 15 -> 16 -> 17 -> NULL

Menu:
1. Create Linked List
2. Display Linked List
3. Insert at Beginning
4. Insert in Middle
5. Insert at End
6. Delete First Node
7. Delete Middle Node
8. Delete Last Node
9. Update Node Data
10. Count Nodes
11. Exit
Enter your choice: 6
Linked List: 12 -> 13 -> 20 -> 14 -> 15 -> 16 -> 17 -> NULL

Menu:
1. Create Linked List
2. Display Linked List
3. Insert at Beginning
4. Insert in Middle
5. Insert at End
6. Delete First Node
7. Delete Middle Node
8. Delete Last Node
9. Update Node Data
10. Count Nodes
11. Exit
Enter your choice: 7
Enter position to delete: 2
Linked List: 12 -> 20 -> 14 -> 15 -> 16 -> 17 -> NULL

Menu:
1. Create Linked List
2. Display Linked List
3. Insert at Beginning
4. Insert in Middle
5. Insert at End
6. Delete First Node
7. Delete Middle Node
8. Delete Last Node
9. Update Node Data
10. Count Nodes
11. Exit
Enter your choice: 8
Linked List: 12 -> 20 -> 14 -> 15 -> 16 -> NULL

Menu:
1. Create Linked List
2. Display Linked List
3. Insert at Beginning
4. Insert in Middle
5. Insert at End
6. Delete First Node
7. Delete Middle Node
8. Delete Last Node
9. Update Node Data
10. Count Nodes
11. Exit
Enter your choice: 9
Enter old data to update: 20
Enter new data: 13
Data updated successfully.
Linked List: 12 -> 13 -> 14 -> 15 -> 16 -> NULL

Menu:
1. Create Linked List
2. Display Linked List
```

```
3. Insert at Beginning
4. Insert in Middle
5. Insert at End
6. Delete First Node
7. Delete Middle Node
8. Delete Last Node
9. Update Node Data
10. Count Nodes
11. Exit
Enter your choice: 10
Number of nodes: 5
Linked List: 12 -> 13 -> 14 -> 15 -> 16 -> NULL

Menu:
1. Create Linked List
2. Display Linked List
3. Insert at Beginning
4. Insert in Middle
5. Insert at End
6. Delete First Node
7. Delete Middle Node
8. Delete Last Node
9. Update Node Data
10. Count Nodes
11. Exit
Enter your choice: 11
Exiting...


=== Code Execution Successful ===
```