Data Structures

Experiment no. 7

Develop code to implement different operations on linked list – copy, concatenate, split, reverse, count no. of nodes, etc.

Q. WAP in C to implement different operations on Linked List.

Code:

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

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

struct Node *node, *list=NULL, *list2=NULL, *last=NULL, *last2=NULL, *temp;

// Create new node
struct Node* createNode(int info) {
   struct Node *node = (struct Node*)malloc(sizeof(struct Node));
   node->data = info;
   node->next = NULL;
   return node;
}

// Add element at end (for both lists)
void addAtEnd(int info, int listNum) {
   node = createNode(info);
   if(listNum == 1) {
      if(list == NULL) {
         list = node;
         last = node;
      } else {
         last->next = node;
         last = node;
      }
   } else {
      if(list2 == NULL) {
         list2 = node;
         last2 = node;
      } else {
```

```
            last2->next = node;
            last2 = node;
        }
    }
}

// Copy linked list from list1 to list2
void copyLinkedList() {
    if(list == NULL) {
        printf("\nOriginal list is empty, cannot copy.");
        return;
    }

    // Clear list2 before copying
    list2 = NULL;
    last2 = NULL;

    temp = list;
    while(temp != NULL) {
        addAtEnd(temp->data, 2); // Add to list2
        temp = temp->next;
    }
}

// Concatenate lists
void concatenateLists() {
    if(list == NULL) {
        list = list2;
    } else {
        last->next = list2;
    }
    list2 = NULL;
    last2 = NULL;
}

// Split list into two halves
void splitList() {
    if(list == NULL || list->next == NULL) {
        printf("\nCannot split - List too short");
        return;
    }

    struct Node *slow = list;
    struct Node *fast = list->next;

    while(fast != NULL && fast->next != NULL) {
        slow = slow->next;
        fast = fast->next->next;
```

```
   }

   list2 = slow->next;
   slow->next = NULL;
}

// Reverse linked list
void reverse() {
   struct Node *prev = NULL, *current = list, *next = NULL;
   while(current != NULL) {
      next = current->next;
      current->next = prev;
      prev = current;
      current = next;
   }
   list = prev;
}

// Count nodes
int countNodes() {
   int count = 0;
   temp = list;
   while(temp != NULL) {
      count++;
      temp = temp->next;
   }
   return count;
}

// Display list
void display(struct Node* head) {
   if(head == NULL) {
      printf("\nList is empty");
      return;
   }
   temp = head;
   while(temp != NULL) {
      printf("%d\t", temp->data);
      temp = temp->next;
   }
   printf("\n");
}

void main() {
   int info, ch;
   struct Node* copiedList;
   clrscr();
```

```
do {
   printf("\n1. Add element to List 1");
   printf("\n2. Add element to List 2");
   printf("\n3. Copy List");
   printf("\n4. Concatenate Lists");
   printf("\n5. Split List");
   printf("\n6. Reverse List");
   printf("\n7. Count Nodes");
   printf("\n8. Display List 1");
   printf("\n9. Display List 2");
   printf("\n10. Exit");
   printf("\nEnter your choice: ");
   scanf("%d", &ch);

   switch(ch) {
      case 1:
         printf("\nEnter value to be added to List 1: ");
         scanf("%d", &info);
         addAtEnd(info, 1);
         break;

      case 2:
         printf("\nEnter value to be added to List 2: ");
         scanf("%d", &info);
         addAtEnd(info, 2);
         break;

      case 3:
         copyLinkedList();
         printf("\nOriginal List: ");
         display(list);
         printf("\nCopied List: ");
         display(list2);
         break;

      case 4:
         concatenateLists();
         printf("\nLists concatenated. Result: ");
         display(list);
         break;

      case 5:
         splitList();
         printf("\nAfter splitting:");
         printf("\nFirst List: ");
         display(list);
         printf("\nSecond List: ");
         display(list2);
```

```
            break;

        case 6:
            reverse();
            printf("\nReversed List: ");
            display(list);
            break;

        case 7:
            printf("\nNumber of nodes: %d", countNodes());
            break;

        case 8:
            printf("\nList 1: ");
            display(list);
            break;

        case 9:
            printf("\nList 2: ");
            display(list2);
            break;

        case 10:
            exit(0);
            break;

        default:
            printf("\nInvalid choice!");
      }
   } while(ch != 10);

   getch();
}
```

Output:

```
1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 1
```

```
Enter value to be added to List 1: 100

1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 1

Enter value to be added to List 1: 20

1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 1

Enter value to be added to List 1: 60

1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 8

List 1: 100     20      60

1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
```

```
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 3

Original List: 100      20      60

Copied List: 100        20      60

1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 9

List 2: 100     20      60

1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 2

Enter value to be added to List 2: 80

1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 9

List 2: 100     20      60      80
```

```
1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 4

Lists concatenated. Result: 100 20      60      100     20
60      80

1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 5

After splitting:
First List: 100 20      60      100

Second List: 20 60      80

1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 6

Reversed List: 100      60      20      100

1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
```

```
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 8

List 1: 100     60      20      100

1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 7

Number of nodes: 4
1. Add element to List 1
2. Add element to List 2
3. Copy List
4. Concatenate Lists
5. Split List
6. Reverse List
7. Count Nodes
8. Display List 1
9. Display List 2
10. Exit
Enter your choice: 10
```