Implementation of Singly linked list:

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


struct node {   // declaration for main linked list
    int data;
    struct node *next;
};
//Start node
struct node *start = NULL;


struct nodeTwo {   // Declaration for secondary linked list
    int dataTwo;
    struct nodeTwo *nextTwo;
};
// Start node of secondary linked list
struct nodeTwo *startTwo = NULL;

void secondLinkedList() {     // Initialises second linked list with static values
    // declare nodes
    struct nodeTwo *newNodeOne;
    struct nodeTwo *newNodeTwo;
    struct nodeTwo *newNodeThree;
    // allocates memory for nodes
    newNodeOne = (struct nodeTwo *)malloc(sizeof(struct nodeTwo));
    newNodeTwo = (struct nodeTwo *)malloc(sizeof(struct nodeTwo));
    newNodeThree = (struct nodeTwo *)malloc(sizeof(struct nodeTwo));
    // enter data and link the nodes
    startTwo = newNodeOne;
    newNodeOne->dataTwo = 4;
    newNodeOne->nextTwo = newNodeTwo;

    newNodeTwo->dataTwo = 8;
    newNodeTwo->nextTwo = newNodeThree;

    newNodeThree->dataTwo = 12;
    newNodeThree->nextTwo = NULL;

}

void insertAtBegining(int val) {    // Inserts node at the begining
    struct node *newNode;
    newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = val;
    if (start == NULL) {    // when 0 nodes are present
        start = newNode;
        start->data = newNode->data;
        start->next = NULL;
        return;
    }
    newNode->next = start;
    start = newNode;
}

void insertAtEnd(int val) {     // Inserts at the end
    struct node *newNode;
    newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = val;
    if (start == NULL) {    // Entering first node
        start = newNode;
        start->data = newNode->data;
        start->next = NULL;
        return;
    } else {
        struct node *ptr;
        ptr = start;

        while (ptr->next != NULL) {
            ptr = ptr->next;
        }
        ptr->next = newNode;
        newNode->next = NULL;
    }

}
```

```
void insertAfterNum(int toInsert, int val) {    // Inserts after a value
    struct node *newNode;
    struct node *temp;  // to store address of next pointer
    struct node *ptr;   // traversing pointer
    newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = toInsert;
    ptr = start;
    while (ptr->data!=val) {    //traverse upto val
        ptr = ptr->next;
    }

    temp = ptr->next;       // store address of next node
    ptr->next = newNode;    // change address to address of new node
    newNode->next = temp;   // set address of new node to the following node
    return;
    printf("\nValue is not present!");
}

void insertBeforeNum(int toInsert, int val) {   // Insert before a value
    if (start == NULL) {
        printf("\nLinked list is empty!");
        return;
    }

    struct node *newNode;
    struct node *ptr;
    struct node *prePtr;
    ptr = start;

    newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = toInsert;

    if  (start->data == val) {  // Inserting before first node
        start = newNode;
        newNode->next = ptr;
        return;
    }

    while(ptr->data != val) {   // Traversing
        prePtr = ptr;
        ptr = ptr->next;
    }
    // Inserting before any node
    prePtr->next = newNode;
    newNode->next = ptr;
}

void insertAfterPos(int toInsert, int pos) {    // Insert after a given position
    struct node *newNode;
    struct node *temp;  // to store address of next pointer
    struct node *ptr;   // traversing pointer
    newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = toInsert;
    ptr = start;
    int count = 1;
    while (count != pos) {    // traverse upto pos
        ptr = ptr->next;
        count++;
    }
    temp = ptr->next;       // store address of next node
    ptr->next = newNode;    // change address to address of new node
    newNode->next = temp;   // set address of new node to the following node
    return;
}

void deleteAtBegining() {   // Delete element at the begining
    struct node *ptr;
    ptr = start;
    if (start == NULL) {
        printf("\nLinked list is empty!");
        return;
    }
    if (start->next == NULL) {  // Deleting only remaining first node
        printf("\nDeleted element is :  %d", ptr->data);
        start = NULL;
        return;
    }

    // Deleting any node
    printf("\nDeleted element is :  %d", ptr->data);
    ptr = ptr->next;
    start->data = ptr->data;
    start->next = ptr->next;
}

void deleteAtEnd() {    // Deletes element at the end
    if (start == NULL) {
        printf("\nLinked list is empty!");
    }
    struct node *ptr;
    struct node *prePtr;
    ptr = start;
    if (start->next == NULL) {  // deleting only remaining node
        printf("\nDeleted element is :  %d", ptr->data);
        start = NULL;
        return;
    }
    while (ptr->next != NULL) { // Traversing
        prePtr = ptr;
        ptr = ptr->next;
    }
    printf("\nDeleted element is :  %d", ptr->data);
    prePtr->next = NULL;
}

void deleteAtPos(int pos) {     // Deltes node after entered position
    if (start == NULL) {
        printf("\nLinked list is empty!");
        return;
    }
    struct node *ptr;
    struct node *prePtr;
    int count = 1;
    ptr = start;
    prePtr = ptr;
    if (start->next == NULL) {  // deleting only remaining node
        printf("\nDeleted element is :  %d", ptr->data);
        start = NULL;
        return;
    }
    while (count < pos) {   // Traversing
        prePtr = ptr;
        ptr = ptr->next;
        count++;
    }
    if (count == 1) {   // Deleting first node
        printf("\nDeleted Element is :  %d", ptr->data);
        start = ptr->next;
        ptr->next = NULL;
        free(ptr);
    } else {    // Deleting any other node
        printf("\nDeleted Element is :  %d", ptr->data);
        prePtr->next = ptr->next;
        ptr->next = NULL;
        free(ptr);
    }
}

void deleteAfterVal(int val) {   // Deletes after a given value
    if (start == NULL) {
        printf("\nLinked list is empty!");
        return;
    }
    struct node *ptr = start;
    struct node *postPtr;

    while (ptr->data != val) {  // Traversing
        ptr = ptr->next;
    }
    if (ptr->next == NULL) {
        printf("\nThere is no element after this!");
    } else {
        printf("\nDeleted element is :  %d", ptr->next->data);
        postPtr = ptr->next;
        ptr->next = postPtr->next;
        postPtr->next = NULL;
    }
}

void deleteBeforeVal(int val) {     // Deletes a node before a given value
```

```
    if (start == NULL) {
        printf("\nLinked list is empty!");
        return;
    }
    struct node *ptr = start;
    struct node *prePtr = ptr;

    if (start->data == val) {
        printf("\nNo node before this!");
        return;
    }

    if (start->next->data == val) {     // If first node is to be deleted
        printf("\nDeleted element is :  %d", start->data);
        start = start->next;
        return;
    }

    ptr = start;
    prePtr = ptr;
    while (ptr->next->data != val) {
        prePtr = ptr;
        ptr = ptr->next;
    }

    // Deleting any other node
    printf("\nDeleted element is :  %d", ptr->data);
    prePtr->next = ptr->next;
    ptr->next = NULL;
    free(ptr);

}

void updateAtBeginning (int val) {      // Updates value at the start
    if (start == NULL) {
        printf("\nLinked list is empty!");
        return;
    }
    start->data = val;
}
```

```
void updateAtEnd (int val) {    // Updates value at the end
    if (start == NULL) {
        printf("\nLinked list is empty!");
        return;
    }
    struct node *ptr = start;
    while (ptr->next != NULL) {
        ptr = ptr->next;
    }
    ptr->data = val;
}

void updateAtPos(int toInsert, int pos) {    // Updates value at the given position
    if (start == NULL) {
        printf("\nLinked list is empty!");
        return;
    }
    int count = 1;
    struct node *ptr = start;
    while (count != pos) {
        ptr = ptr->next;
        count++;
    }
    ptr->data = toInsert;
}

void updateAfterVal(int toInsert, int val) {    // Updates after entered value is encountered
    if (start == NULL) {
        printf("\nLinked list is empty!");
        return;
    }
    struct node *ptr = start;
    struct node *postPtr;

    while (ptr->data != val) {    // Traversing
        ptr = ptr->next;
    }
    if (ptr->next == NULL) {     // If the value is of last node
        printf("\nThere is no element after this!");
    } else {    // Update any other node
        postPtr = ptr->next;
        postPtr->data = toInsert;
    }
}

void updateBeforeVal(int toInsert, int val) {    // Updates before entered value is encounterd
    if (start == NULL) {
        printf("\nLinked list is empty!");
        return;
    }
    struct node *ptr = start;
    struct node *prePtr;
    int count = 0;
    while (ptr->data != val) {    // Traverse
        prePtr = ptr;
        ptr = ptr->next;
        count++;
    }
    if (count == 0) {    // If value is of first node
        printf("\nThere is no element before this!");
        return;
    }
    // Update any other node
    prePtr->data = toInsert;
}

void search(int val) {  // Search for element in the array
    struct node *ptr;
    int count = 0;
    ptr = start;
    if (ptr == NULL) {
        printf("\nList is empty");
        return;
    }
    while (ptr->next != NULL) {
        if (val == ptr->data) {
            printf("\n%d  is present on node index  :  %d", val, count);
            return;
        }
        ptr = ptr->next;
        count++;
    }
    printf("\nElement not found!");
}

void reverse() {    // Reverses the list
    struct node *previousNode, *currentNode, *nextNode;
    previousNode = NULL;
    currentNode = nextNode = start;
    while (nextNode != NULL) {
        nextNode = nextNode->next;
        currentNode->next = previousNode;
        previousNode = currentNode;
        currentNode = nextNode;
    }
    start = previousNode;
}

void countNodes() {    // Count nodes in the list
    struct node *ptr = start;
    int count = 1;
    while (ptr->next != NULL) {
        ptr = ptr->next;
        count++;
    }
    printf("There are  %d  nodes", count);
}

void display() {  // traverse through the list
    struct node* ptr;
    ptr = start;
    if (ptr == NULL) {
        printf("\nList is empty!");
        return;
    }
    printf("\n");
    while (ptr->next != NULL) {
        printf("%d  ", ptr->data);
        ptr = ptr->next;
    }
    printf("%d ", ptr->data);
}

void sort() {    // Sorts the list
    struct node *i = start;
    struct node *j = NULL;
    int temp;
    for (i = start ; i != NULL ;  i=i->next) {
        for (j = i->next ; j != NULL ; j = j->next) {
            if (i->data > j->data) {
                temp = i->data;
                i->data = j->data;
                j->data = temp;
            }
        }
    }
}

void concat() {
    struct node *ptr;
    struct nodeTwo *ptrTwo;
    ptr = start;
    while (ptr->next != NULL) {
        ptr = ptr->next;
    }
    ptr->next = (struct node *)startTwo;
}

void displayListTwo() {

    struct nodeTwo* ptr;
    ptr = startTwo;
    if (ptr == NULL) {
        printf("\nList is empty!");
        return;
    }
    printf("\n");
    while (ptr->nextTwo != NULL) {
        printf("%d  ", ptr->dataTwo);
        ptr = ptr->nextTwo;
    }
    printf("%d ", ptr->dataTwo);
}




int main() {
    int choice, item, pos, val;
    // displayListTwo();

    while (1) {
        printf("\n*1  Insert at the beginning");
        printf("\n*2  Insert at the end");
        printf("\n*3  Insert after position");
        printf("\n*4  Insert after a given value");
        printf("\n*5  Insert before given value");
        printf("\n*6  Delete at a particular position");
        printf("\n*7  Delete at beginning");
        printf("\n*8  Delete value at end");
        printf("\n*9  Delete after a particular value");
        printf("\n*10 Delete before a particular value");
        printf("\n*11 Update the value of given position");
        printf("\n*12 Update value at the beginning");
        printf("\n*13 Update value at the end");
        printf("\n*14 Update after a particular value");
        printf("\n*15 Update before a particular value");
        printf("\n*16 Search");
        printf("\n*17 Reverse");
        printf("\n*18 Count Nodes");
        printf("\n*19 Display");
        printf("\n*20 Sort");
        printf("\n*21 Concat");
        printf("\n*22 Merge");
        printf("\n*23 EXIT");
        printf("\n");
        printf("\nEnter your choice :  ");
        scanf("%d", &choice);
```

```
switch(choice) {

    case 1:
        printf("\nEnter an element to add :  ");
        scanf("%d", &item);
        insertAtBegining(item);
        break;

    case 2:
        printf("\nEnter an element to add :  ");
        scanf("%d", &item);
        insertAtEnd(item);
        break;

    case 3:
        printf("\nEnter an element to add :  ");
        scanf("%d", &item);
        printf("\nEnter position after which to add :  ");
        scanf("%d", &pos);
        insertAfterPos(item, pos);
        break;

    case 4:
        printf("\nEnter an element to add :  ");
        scanf("%d", &item);
        printf("\nEnter value after which to add :  ");
        scanf("%d", &val);
        insertAfterNum(item, val);
        break;

    case 5:
        printf("\nEnter an element to add :  ");
        scanf("%d", &item);
        printf("\nEnter value before which to add :  ");
        scanf("%d", &val);
        insertBeforeNum(item, val);
        break;

    case 6:
        printf("\nEnter position from where to delete :  ");
    scanf("%d", &item);
    deleteAtPos(item);
    break;

case 7:
    deleteAtBegining();
    break;

case 8:
    deleteAtEnd();
    break;

case 9:
    printf("\nEnter value after which to delete :  ");
    scanf("%d", &item);
    deleteAfterVal(item);
    break;

case 10:
    printf("\nEnter value before which to delete :  ");
    scanf("%d", &item);
    deleteBeforeVal(item);
    break;

case 11:
    printf("\nEnter an element to update :  ");
    scanf("%d", &item);
    printf("\nEnter value at which to update :  ");
    scanf("%d", &pos);
    updateBeforeVal(item, pos);
    break;

case 12:
    printf("\nEnter an element to update :  ");
    scanf("%d", &item);
    updateAtBeginning(item);
    break;

case 13:
    printf("\nEnter an element to update :  ");
    scanf("%d", &item);
    updateAtEnd(item);
    break;

case 14:
    printf("\nEnter an element to update :  ");
    scanf("%d", &item);
    printf("\nEnter value after which to update :  ");
    scanf("%d", &val);
    updateAfterVal(item, val);
    break;

case 15:
    printf("\nEnter an element to update :  ");
    scanf("%d", &item);
    printf("\nEnter value before which to update :  ");
    scanf("%d", &val);
    updateBeforeVal(item, val);
    break;

case 16:
    printf("\nEnter elment to search ");
    scanf("%d", &item);
    search(item);
    break;

case 17:
    reverse();
    break;

case 18:
    countNodes();
    break;

case 19:
    printf("\nEnlements in the list are :");
    display();
    break;

case 20:
```

```
                sort();
                break;

            case 21:
                printf("List 1 :  ");
                display();
                printf("\nList 2 :  ");
                secondLinkedList();
                displayListTwo();
                concat(item);
                printf("\nList after concatenation :  ");
                display();
                break;

            case 22:
                printf("List 1 :  ");
                display();
                printf("\nList 2 :  ");
                secondLinkedList();
                displayListTwo();
                concat();
                sort();
                printf("\nList after merging :  ");
                display();
                break;

            case 23:
                printf("\n***EXITING***\n");
                exit(1);
                break;
            default:
                printf("INVALID INPUT");
        }
    }

    return 0;
}
```

```
//output
```

```
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  1

Enter an element to add :  5

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
```

```
Enter your choice :  2

Enter an element to add :  10

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  2

Enter an element to add :  15

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
```

```
Enter your choice :  2

Enter an element to add :  15

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  19

Enlements in the list are :

5  10  15
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
```

```
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  3

Enter an element to add :  20

Enter position after which to add :  3

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  4

Enter an element to add :  25

Enter value after which to add :  20

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
```

```
Enter your choice :  4

Enter an element to add :  25

Enter value after which to add :  20

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  19

Enlements in the list are :

5  10  15  20  25
```

```
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  6

Enter position from where to delete :  2
```

```
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  7

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT
```

```
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  19

Enlements in the list are :

15  20
```

```
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  9

Enter value after which to delete :  15

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT
```

```
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  12

Enter an element to update :  20

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT
```

```
Enlements in the list are :

20
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  2

Enter an element to add :  15
```

```
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  2

Enter an element to add :  10

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
```

```
Enter your choice :  19

Enlements in the list are :

20  15  10
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  12

Enter an element to update :  40
```

```
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  19

Enlements in the list are :

40  15  10
```

```
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  16

Enter elment to search 15
```

```
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  18
There are  3  nodes
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT
```

```
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  19

Enlements in the list are :

10  15  40
```

```
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  17

*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT
```

```
40  15  10
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  21

Enter an element to Concat :  55
```

```
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  19

Enlements in the list are :

55  40  15  10
```

```
Enlements in the list are :

55  40  15  10
*1  Insert at the beginning
*2  Insert at the end
*3  Insert after position
*4  Insert after a given value
*5  Insert before given value
*6  Delete at a particular position
*7  Delete at beginning
*8  Delete value at end
*9  Delete after a particular value
*10 Delete before a particular value
*11 Update the value of given position
*12 Update value at the beginning
*13 Update value at the end
*14 Update after a particular value
*15 Update before a particular value
*16 Search
*17 Reverse
*18 Count Nodes
*19 Display
*20 Sort
*21 Concat
*22 EXIT

Enter your choice :  22

***EXITING***

Process returned 1 (0x1)   execution time : 707.504 s
Press any key to continue.
```