Programmer Name: Param Chordiya
Batch: E5
Problem Statement:

Create a singly linked list to perform following operation on it

a. Insert (at front, at end, in the middle), b. Delete (at front, at end, in the middle), c. Display, d. Display Reverse, e. Revert the SLL

INPUT:

```
#include <stdio.h>
#include <malloc.h>
#define ISEMPTY printf("\nEMPTY LIST:");

struct node {
        int value;
        struct node *next;
};

typedef struct node snode;

snode* create_node(int);
void insert_node_first();
void insert_node_last();
void insert_node_pos();
void sorted_ascend();
void delete_pos();
void search();
void display();
void rev_display(snode *);

snode *newnode, *ptr, *prev, *temp;
snode *first = NULL, *last = NULL;

void main() {
        int ch;
        char ans = 'Y';
```

```
while (ans == 'Y'||ans == 'y') {
   printf("\n---------------------------------\n");
   printf("\nOperations on singly linked list\n");
   printf("\n---------------------------------\n");
   printf("\t  ROLL NO:22119");
   printf("\n---------------------------------\n");
   printf("\n1.Insert node at front");
   printf("\n2.Insert node at end");
   printf("\n3.Insert node at position");
   printf("\n4.Sort Linked List in Ascending Order");
   printf("\n5.Delete Node from any Position");

   printf("\n6.Display List");
   printf("\n7.Display reverse list");
   printf("\n8.Exit\n");
   printf("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
   printf("\nEnter your choice:");
   scanf("%d", &ch);

   switch (ch) {
          case 1:
                printf("\n...Inserting node at first...\n");
                insert_node_first();
                break;
          case 2:
                printf("\n...Inserting node at last...\n");
                insert_node_last();
                break;
          case 3:
                printf("\n...Inserting node at position...\n");
                insert_node_pos();
                break;
          case 4:
                printf("\n...Sorted Linked List in Ascending Order...\n");
                sorted_ascend();
                break;
```

```
                case 5:
                    printf("\n...Deleting Node from any Position...\n");
                    delete_pos();
                    break;

                case 6:
                    printf("\n...Displaying List From Beginning to End...\n");
                    display();
                    break;
                case 7:
                    printf("\n...Displaying List From End using Recursion...\n");
                    rev_display(first);
                    break;
                case 8:
                    printf("\n...Exiting...\n");
                    return 0;
                    break;
                default:
                    printf("\n...Invalid Choice...\n");
                    break;
            }
            printf("\nDO YOU WANT TO CONTINUE ? (Y/N) :");
            scanf(" %c", &ans);
        }


}

snode* create_node(int val) {
        newnode = (snode *)malloc(sizeof(snode));

        if (newnode == NULL) {
            printf("\nMemory was not allocated");
            return 0;
        }
        else {
            newnode->value = val;
```

```
        newnode->next = NULL;
        return newnode;
    }
}

void insert_node_first() {
    int val;
    printf("\nEnter the value for the node:");
    scanf("%d", &val);
    newnode = create_node(val);

    if (first == last && first == NULL) {
        first = last = newnode;
        first->next = NULL;
        last->next = NULL;
    }
    else {
        temp = first;
        first = newnode;
        first->next = temp;
    }

    printf("\n----INSERTED----");
}


void insert_node_last() {
    int val;
    printf("\nEnter the value for the Node:");
    scanf("%d", &val);
    newnode = create_node(val);

    if (first == last && last == NULL) {
        first = last = newnode;
        first->next = NULL;
        last->next = NULL;
    }
```

```
        else {
           last->next = newnode;
           last = newnode;
           last->next = NULL;
        }

        printf("\n----INSERTED----");
}


void insert_node_pos() {
        int pos, val, cnt = 0, i;
        printf("\nEnter the value for the Node:");
        scanf("%d", &val);
        newnode = create_node(val);
        printf("\nEnter the position: ");
        scanf("%d", &pos);
        ptr = first;
        while (ptr != NULL) {
           ptr = ptr->next;
           cnt++;
        }

        if (pos == 1) {
           if (first == last && first == NULL) {
                first = last = newnode;
                first->next = NULL;
                last->next = NULL;
           }
           else {
                temp = first;
                first = newnode;
                first->next = temp;
           }

           printf("\n----Inserted----");
        }
```

```
        else if (pos>1 && pos<=cnt) {
           ptr = first;
           for (i = 1;i < pos;i++) {
                 prev = ptr;
                 ptr = ptr->next;
           }
           prev->next = newnode;
           newnode->next = ptr;
           printf("\n----INSERTED----");
        }
        else {
           printf("Position is out of range");
        }
}

void sorted_ascend() {
        snode *nxt;
        int t;

        if (first == NULL) {
           ISEMPTY;
           printf("No elements to sort\n");
        }
        else {
           for (ptr = first;ptr != NULL;ptr = ptr->next) {
                 for (nxt = ptr->next;nxt != NULL;nxt = nxt->next) {
                   if (ptr->value > nxt->value) {
                      t = ptr->value;
                      ptr->value = nxt->value;
                      nxt->value = t;
                   }
                 }
           }

           printf("\n---Sorted List---");

           for (ptr = first;ptr != NULL;ptr = ptr->next) {
```

```
            printf("%d\t", ptr->value);
        }
    }
}

void delete_pos() {
    int pos, cnt = 0, i;

    if (first == NULL) {
        ISEMPTY;
        printf("No node to delete\n");
    }
    else {
        printf("\nEnter the position of value to be deleted:");
        scanf(" %d", &pos);
        ptr = first;

        if (pos == 1) {
            first = ptr->next;
            printf("\n----Element deleted----");
        }
        else {
            while (ptr != NULL) {
                ptr = ptr->next;
                cnt = cnt + 1;
            }
            if (pos > 0 && pos <= cnt) {
                ptr = first;
                for (i = 1;i < pos;i++) {
                    prev = ptr;
                    ptr = ptr->next;
                }
                prev->next = ptr->next;
            }
            else {
                printf("Position is out of range");
            }
```

```
            free(ptr);
            printf("\n----Element deleted----");
            }
        }
}
void display() {
        if (first == NULL) {
            ISEMPTY;
            printf("No nodes in the list to display\n");
        }
        else {
            for (ptr = first;ptr != NULL;ptr = ptr->next) {
                printf("%d\t", ptr->value);
            }
        }
}


void rev_display(snode *ptr) {
        int val;

        if (ptr == NULL) {
            ISEMPTY;
            printf("No nodes to display\n");
        }
        else {
            if (ptr != NULL) {
                val = ptr->value;
                rev_display(ptr->next);
                printf("%d\t", val);
            }
        }
}
```

OUTPUT:

```
--------------------------------

Operations on singly linked list

--------------------------------
          ROLL NO:22119
--------------------------------

1.Insert node at front
2.Insert node at end
3.Insert node at position
4.Sort Linked List in Ascending Order
5.Delete Node from any Position
6.Display List
7.Display reverse list
8.Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your choice:1

...Inserting node at first...

Enter the value for the node:1

----INSERTED----
DO YOU WANT TO CONTINUE ? (Y/N) :y

--------------------------------

Operations on singly linked list

--------------------------------
          ROLL NO:22119
--------------------------------

1.Insert node at front
2.Insert node at end
3.Insert node at position
4.Sort Linked List in Ascending Order
5.Delete Node from any Position
6.Display List
7.Display reverse list
8.Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your choice:2
```

```
Enter the value for the Node:6

----INSERTED----
DO YOU WANT TO CONTINUE ? (Y/N) :y

---------------------------------

Operations on singly linked list

---------------------------------
          ROLL NO:22119
---------------------------------

1.Insert node at front
2.Insert node at end
3.Insert node at position
4.Sort Linked List in Ascending Order
5.Delete Node from any Position
6.Display List
7.Display reverse list
8.Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your choice:3

...Inserting node at position...

Enter the value for the Node:8

Enter the position: 1

----Inserted----
DO YOU WANT TO CONTINUE ? (Y/N) :y

---------------------------------

Operations on singly linked list

---------------------------------
          ROLL NO:22119
---------------------------------

1.Insert node at front
2.Insert node at end
3.Insert node at position
4.Sort Linked List in Ascending Order
5.Delete Node from any Position
6.Display List
```

```
5.Delete Node from any Position
6.Display List
7.Display reverse list
8.Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your choice:6

...Displaying List From Beginning to End...
8       1       6
DO YOU WANT TO CONTINUE ? (Y/N) :y

---------------------------------

Operations on singly linked list

---------------------------------
          ROLL NO:22119
---------------------------------

1.Insert node at front
2.Insert node at end
3.Insert node at position
4.Sort Linked List in Ascending Order
5.Delete Node from any Position
6.Display List
7.Display reverse list
8.Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your choice:7

...Displaying List From End using Recursion...

EMPTY LIST:No nodes to display
6       1       8
DO YOU WANT TO CONTINUE ? (Y/N) :y

---------------------------------

Operations on singly linked list
```

```
          ROLL NO:22119
---------------------------------

1.Insert node at front
2.Insert node at end
3.Insert node at position
4.Sort Linked List in Ascending Order
5.Delete Node from any Position
6.Display List
7.Display reverse list
8.Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your choice:4

...Sorted Linked List in Ascending Order...

---Sorted List---1      6       8
DO YOU WANT TO CONTINUE ? (Y/N) :y

---------------------------------

Operations on singly linked list

---------------------------------
          ROLL NO:22119
---------------------------------

1.Insert node at front
2.Insert node at end
3.Insert node at position
4.Sort Linked List in Ascending Order
5.Delete Node from any Position
6.Display List
7.Display reverse list
8.Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your choice:5

...Deleting Node from any Position...

Enter the position of value to be deleted:1

----Element deleted----
DO YOU WANT TO CONTINUE ? (Y/N) :y

---------------------------------
```

```
---------------------------------

Operations on singly linked list

---------------------------------
          ROLL NO:22119
---------------------------------

1.Insert node at front
2.Insert node at end
3.Insert node at position
4.Sort Linked List in Ascending Order
5.Delete Node from any Position
6.Display List
7.Display reverse list
8.Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your choice:6

...Displaying List From Beginning to End...
6       8
DO YOU WANT TO CONTINUE ? (Y/N) :y

---------------------------------

Operations on singly linked list

---------------------------------
          ROLL NO:22119
---------------------------------

1.Insert node at front
2.Insert node at end
3.Insert node at position
4.Sort Linked List in Ascending Order
5.Delete Node from any Position
6.Display List
7.Display reverse list
8.Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your choice:8

...Exiting...

---------------------------------
```