Aim:
Linked implementation of stack in C.

Theory:
Stack is a data structure which is based on LIFO (Last In First Out) system. To access the items of the stack we have top of the stack. We can add or delete items from the top of the stack only. The operations that can be performed on the stack are push(adding an item to the stack) and pop(deleting an item from the stack).

ASSIGNMENT:1

PROBLEM:
Write a C program to implement stack using:
a) Array

SOURCE CODE:

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

struct stack
{
   int size;
   int top;
   int *arr;
};

int isEmpty(struct stack *ptr)
{
   if (ptr->top == -1)
   {
      return 1;
   }
   else
   {
      return 0;
   }
}

int isFull(struct stack *ptr)
{
   if (ptr->top == ptr->size - 1)
   {
      return 1;
   }
   else
   {
      return 0;
```

```
    }
}

int main()
{
    struct stack *s;
    s->size = 80;
    s->top = -1;
    s->arr = (int *)malloc(s->size * sizeof(int));

s->arr[0]=4;
s->arr[1]=5;
s->top++;

 if(isEmpty(s)){
        printf("The stack is empty");
    }
    else{
        printf("The stack is not empty");
    }
    return 0;
}
```

OUTPUT:

```
The stack is not empty
```

PROBLEM:

Write a C program to implement stack using:

b) Linked List

SOURCE CODE:

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

struct Node {
    int data;
    struct Node *next;

struct Node *head = NULL;

void push(int value) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = head;
    head = newNode;
    printf("Pushed %d onto the stack.\n", value);
}

int pop() {
    if (head == NULL) {
        printf("Stack Underflow. Cannot pop.\n");
        return -1;  // Returning -1 to indicate an error
    }

    struct Node *temp = head;
    int poppedValue = temp->data;
    head = head->next;
    free(temp);
    return poppedValue;
}

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

    printf("Stack elements: ");
    struct Node *current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

int main() {
    int choice, value;
    printf("Implementation of Stack using Linked List\n");

    while (1) {
        printf("\n1. Push\n2. Pop\n3. Display\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the value to insert: ");
                scanf("%d", &value);
                push(value);
                break;
            case 2:
                printf("Popped element is: %d\n", pop());
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
            default:
                printf("Wrong Choice. Please enter a valid option.\n");
        }
    }
    return 0;
}
```

Output:

```
Implementation of Stack using Linked List

1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 5
Pushed 5 onto the stack.

1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 10
Pushed 10 onto the stack.

1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack elements: 10 5

1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Popped element is: 10

1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack elements: 5

1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 4
```

ASSIGNMENT:2

Problem:

Given a stack, sort it using recursion. Use of any loop constructs like while, for etc. is not allowed.
Use following functions on Stack S.

a) isEmpty(S) /*Tests whether stack is empty or not*/
b) push(S) /* Adds new element to the stack*/
c) pop(S) /* Removes top element from the stack*/
d) top(S) /*Returns value of the top of the element*/

SOURCE CODE:

```
#include <stdio.h>

int stack[100];
int i, j, choice = 0, n, top = -1;

void push();
void pop();
void show();
void oop();

int main()
{
   printf("enter the no of element in the stack:");
   scanf("%d", &n);

   while (choice != 4)
   {
      printf("1: push \n");
      printf("2: pop \n");
      printf("3: show \n");
      printf("4: top element \n");
      printf(":5 exit \n");
      printf("enter the choice for stack:\n");
      scanf("\n%d", &choice);

      switch (choice)
      {
      case 1:
      {
         push();
         break;
      }

      case 2:
      {
         pop();
         break;
      }
      case 3:
      {
         show();
         break;
```

```
        }
        case 4:
        {
            top_element();
            // printf("your are exiting....");
            break;
        }
        case 5:
        {
            printf("your are exiting....");
            break;
        }

        default:
        {
            printf("enter the number from 1 to 4");
        }
        };
    }
    return 0;
}


void push(){
    int val;

  if(top==(n-1)){
    printf("overflow");
  }
  else{
    printf("enter the value:");
    scanf("%d",&val);

    top=top+1;
    stack[top]=val;
  };
}

void pop(){
    if(top==-1){
        printf("the empty stack");
    }
    else{
        top = top - 1;
    };
}
void show(){
    for(i=top;i>=0;i--){
        printf("%d\n", stack[i]);
    }
    if (top == -1)
    {
        printf("Stack is empty");
    };

}
```

```
void top_element() {
    if (top == -1) {
        printf("Stack is empty\n");
    } else {
        printf("Top element: %d\n", stack[top]);
    }
}
```

Output:

```
enter the no of element in the stack:3
1: push
2: pop
3: show
4: top element
:5 exit
enter the choice for stack:
1
enter the value:23
1: push
2: pop
3: show
4: top element
:5 exit
enter the choice for stack:
1
enter the value:22
1: push
2: pop
3: show
4: top element
:5 exit
enter the choice for stack:
1
enter the value:43
1: push
2: pop
3: show
4: top element
:5 exit
enter the choice for stack:
4
Top element: 43
```

ASSIGNMENT:3

PROBLEM:

Write a C program to implement a stack using linked list and accept some numeric values. Remove the number whose value is the minimum on the stack from the given variable.

SOURCE CODE:

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

// Structure to define each node in the linked list
struct Node {
    int info;
    struct Node *ptr;
};

// Global pointer to the top of the stack
struct Node *head = NULL;

// Function to push an element onto the stack
void push(int data) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->info = data;
    newNode->ptr = head;
    head = newNode;
}

// Function to pop an element from the stack
int pop() {
    if (head == NULL) {
        printf("Stack Underflow. Cannot pop.\n");
        return -1;
    }

    struct Node *temp = head;
    int poppedValue = temp->info;
    head = head->ptr;
    free(temp);
    return poppedValue;
}

// Function to remove the first occurrence of the minimum value in the stack
void removeMin() {
    if (head == NULL) {
        printf("Stack is empty. Cannot remove minimum.\n");
        return;
    }

    int min = head->info;
    struct Node *current = head;
    struct Node *prev = NULL;
    // Find the minimum value in the stack
    while (current != NULL) {
        if (current->info < min) {
            min = current->info;
        }
        prev = current;
        current = current->ptr;
    }

    // Remove the first occurrence of the minimum value
    current = head;
    while (current != NULL) {
        if (current->info == min) {
            if (prev == NULL) {
                head = current->ptr;
            } else {
                prev->ptr = current->ptr;
            }
            free(current);
            printf("Removed %d from the stack.\n", min);
            return;
        }
        prev = current;
        current = current->ptr;
    }
    printf("Minimum value %d not found in the stack.\n", min);
}

int main() {
    int n, data;
    printf("Enter the number of values: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter value %d: ", i + 1);
        scanf("%d", &data);
        push(data);
    }

    printf("Removing the minimum value from the stack...\n");
    removeMin();
    return 0;
}
```

OUTPUT:

```
Enter the number of values: 4
Enter value 1: 5
Enter value 2: 10
Enter value 3: 3
Enter value 4: 8
Removing the minimum value from the stack...
Removed 3 from the stack.
```