```
//stack using arrays
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
typedef struct {
    int top;
    int items[MAX];
} St;
void initStack(St *s) {
    s->top = -1;
}
int isEmpty(St *s) {
    return s->top == -1;
}
int isFull(St*s) {
    return s->top == MAX - 1;
}
void push(St *s, int item) {
    if (isFull(s)) {
        printf("Stack overflow. Cannot push %d.\n", item);
        return;
    }
```

```
    s->items[++(s->top)] = item;
}
int pop(St *s) {
    if (isEmpty(s)) {
        printf("Stack underflow. Cannot pop.\n");
        return -1;
    }
    return s->items[(s->top)--];
}
int peek(St *s) {
    if (isEmpty(s)) {
        printf("Stack is empty.\n");
        return -1;
    }
    return s->items[s->top];
}
void display(St*s) {
    if (isEmpty(s)) {
        printf("Stack is empty.\n");
        return;
    }
    printf("Stack elements are:\n");
```

```
    for (int i = s->top; i >= 0; i--) {
        printf("%d\n", s->items[i]);
    }
}
int main() {
    St s;
    initStack(&s);
    push(&s, 10);
    push(&s, 20);
    push(&s, 30);
    printf("Top element is %d\n", peek(&s));
    display(&s);
    printf("Popped element is %d\n", pop(&s));
    printf("Popped element is %d\n", pop(&s));
    display(&s);
    return 0;
}
```

2.//stack using linked lists

```
#include <stdio.h>
#include <stdlib.h>
typedef struct ListNode {
```

```
    int data;
    struct ListNode* next;
} ListNode;
typedef struct StackNode {
    ListNode* listNode;
    struct StackNode* next;
} StackNode;
typedef struct {
    StackNode* top;
} Stack;
Stack* createStack() {
    Stack* stack = (Stack*)malloc(sizeof(Stack));
    stack->top = NULL;
    return stack;}
void push(Stack* stack, ListNode* listNode) {
    StackNode* newNode = (StackNode*)malloc(sizeof(StackNode));
    newNode->listNode = listNode;
    newNode->next = stack->top;
    stack->top = newNode;}
ListNode* pop(Stack* stack) {
    if (stack->top == NULL) {
        printf("Stack underflow\n");
```

```
        return NULL;}
    StackNode* temp = stack->top;
    ListNode* poppedNode = temp->listNode;
    stack->top = stack->top->next;
    free(temp);
    return poppedNode;}
ListNode* peek(Stack* stack) {
    if (stack->top == NULL) {
        printf("Stack is empty\n");
        return NULL;}
    return stack->top->listNode;}
int isEmpty(Stack* stack) {
    return stack->top == NULL;
}
void pushListToStack(Stack* stack, ListNode* head) {
    ListNode* current = head;
    while (current != NULL) {
        push(stack, current);
        current = current->next;}
}
ListNode* reverseLinkedListUsingStack(Stack* stack) {
    if (isEmpty(stack)) {
```

```
        return NULL;}
    ListNode* head = pop(stack);
    ListNode* current = head;
    while (!isEmpty(stack)) {
        current->next = pop(stack);
        current = current->next;}
    current->next = NULL; // End the list
    return head;}

void printList(ListNode* head) {
    ListNode* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}
int main() {
    // Create a stack
    Stack* stack = createStack();
    ListNode* head = (ListNode*)malloc(sizeof(ListNode));
    head->data = 1;
```

```
    head->next = (ListNode*)malloc(sizeof(ListNode));
    head->next->data = 2;
    head->next->next = (ListNode*)malloc(sizeof(ListNode));
    head->next->next->data = 3;
    head->next->next->next = NULL;
    printf("Original linked list:\n");
    printList(head);
    pushListToStack(stack, head);
    head = reverseLinkedListUsingStack(stack);
    printf("Reversed linked list:\n");
    printList(head);
    free(stack);
    return 0;
}
```