```c
1    LINKED LIST
2    #include <stdio.h>
3    #include <stdlib.h>
4    struct Node {
5        int data;
6        struct Node* next;
7    };
8    void insertAtBeginning(struct Node** head, int newData) {
9        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
10       newNode->data = newData;
11       newNode->next = *head;
12       *head = newNode;
13   }
14   void printList(struct Node* head) {
15       struct Node* temp = head;
16       while (temp != NULL) {
17           printf("%d ", temp->data);
18           temp = temp->next;
19       }
20       printf("\n");
21   }
22   int main() {
23       struct Node* head = NULL;
24       insertAtBeginning(&head, 5);
25       insertAtBeginning(&head, 10);
26       insertAtBeginning(&head, 15);
27       printf("Linked List: ");
28       printList(head);
29       return 0;
30   }
31
```

```
1    SINGLE LINKED LIST
2    #include <stdio.h>
3    #include <stdlib.h>
4    struct Node {
5        int data;
6        struct Node* next;
7    };
8
```

```c
DOUBLE LINKED LIST
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};
```

```
1   CIRCULAR LINKED LIST
2   #include <stdio.h>
3   #include <stdlib.h>
4   struct Node
5   {
6       int data;
7       struct Node* next;
8   };
9   
```

```c
#include <stdio.h>
#define MAX_SIZE 100
struct Stack {
    int items[MAX_SIZE];
    int top;
};
void initialize(struct Stack *s) {
    s->top = -1;
}
int isFull(struct Stack *s) {
    return s->top == MAX_SIZE - 1;
}
int isEmpty(struct Stack *s) {
    return s->top == -1;
}
void push(struct Stack *s, int value) {
    if (isFull(s)) {
        printf("Stack Overflow\n");
    } else {
        s->top++;
        s->items[s->top] = value;
        printf("%d pushed to stack\n", value);
    }
}

int main() {
    struct Stack stack;
    initialize(&stack);

    push(&stack, 10);
    push(&stack, 20);
    push(&stack, 30);

    return 0;
}
```

```c
STACK POP OPERTIONS
int pop(struct Stack* stack)
{
    if (stack->top == -1)
    {
        printf("Stack Underflow\n");
        return -1;
    int poppedValue = stack->items[stack->top];
    stack->top--;
    printf("%d popped from stack\n", poppedValue);
    return poppedValue;
}
```

```
1   STACK PEEK OPERATIONS
2   int peek(struct Stack* stack)
3   {
4       if (stack->top == -1)
5       {
6           printf("Stack is empty\n");
7           return -1;
8       }
9       return stack->items[stack->top];
10  }
11
```

```c
#include <stdio.h>
#define MAX_SIZE 5
struct Stack {
    int items[MAX_SIZE];
    int top;
};
void initialize(struct Stack *s) {
    s->top = -1;
}
int isFull(struct Stack *s) {
    return s->top == MAX_SIZE - 1;
}
int main() {
    struct Stack stack;
    initialize(&stack);
    for (int i = 0; i < MAX_SIZE; i++) {
        if (!isFull(&stack)) {
            stack.items[++stack.top] = i + 1;
            printf("%d pushed to stack\n", i + 1);
        } else {
            printf("Stack is full\n");
        }
    }
    return 0;
}
```

```c
STACK ISEMPTY
#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;
    }
}
void push(struct stack* ptr, int val){
    if(isFull(ptr)){
        printf("Stack Overflow! Cannot push %d to the stack\n", val);
    }
    else{
        ptr->top++;
        ptr->arr[ptr->top] = val;
    }
}
int pop(struct stack* ptr){
    if(isEmpty(ptr)){
        printf("Stack Underflow! Cannot pop from the stack\n");
        return -1;
```

```c
37            return -1;
38        }
39        else{
40            int val = ptr->arr[ptr->top];
41            ptr->top--;
42            return val;
43        }
44    }
45    int main(){
46        struct stack *sp = (struct stack *) malloc(sizeof(struct stack));
47        sp->size = 10;
48        sp->top = -1;
49        sp->arr = (int *) malloc(sp->size * sizeof(int));
50        printf("Stack has been created successfully\n");
51        return 0;
52    }
53
```

```c
#include <stdio.h>
#define MAX_SIZE 100
struct Stack {
    int items[MAX_SIZE];
    int top;
};
void initialize(struct Stack *s) {
    s->top = -1;
}
int isFull(struct Stack *s) {
    return s->top == MAX_SIZE - 1;
}
int isEmpty(struct Stack *s) {
    return s->top == -1;
}
void push(struct Stack *s, int value) {
    if (isFull(s)) {
        printf("Stack Overflow\n");
    } else {
        s->top++;
        s->items[s->top] = value;
        printf("%d pushed to stack\n", value);
    }
}
int pop(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack Underflow\n");
        return -1;
    } else {
        int popped = s->items[s->top];
        s->top--;
        printf("%d popped from stack\n", popped);
        return popped;
    }
}
```

```c
}
void display(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack is empty\n");
    } else {
        printf("Stack elements: ");
        for (int i = 0; i <= s->top; i++) {
            printf("%d ", s->items[i]);
        }
        printf("\n");
    }
}
int main() {
    struct Stack stack;
    initialize(&stack);
    push(&stack, 10);
    push(&stack, 20);
    push(&stack, 30);
    display(&stack);
    pop(&stack);
    display(&stack);
    return 0;
}
```

```c
#include <stdio.h>
struct Node {
    int data;
    struct Node *next;
};
struct Stack {
    struct Node *top;
};
struct Node* createNode(int data) {
    struct Node *newNode = (struct Node *) malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
int isEmpty(struct Stack *s) {
    return (s->top == NULL);
}
void push(struct Stack *s, int data) {
    struct Node *newNode = createNode(data);
    newNode->next = s->top;
    s->top = newNode;
    printf("%d pushed to stack\n", data);
}
int pop(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack Underflow\n");
        return -1;
    }
```

```c
void display(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack is empty\n");
        return;
    }
    struct Node *current = s->top;
    printf("Stack elements: ");
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}
int main() {
    struct Stack stack;
    stack.top = NULL;
    push(&stack, 10);
    push(&stack, 20);
    push(&stack, 30);
    display(&stack);
    pop(&stack);
    display(&stack);
    return 0;
}
```