1.stack using Array Implementation:

```
#include <stdio.h>

#include <stdlib.h>


#define MAX 100


Typedef struct Stack {

   Int items[MAX];

   Int top;

} Stack;

Void initialize(Stack* stack) {

   Stack->top = -1;

}

Int isEmpty(Stack* stack) {

   Return stack->top == -1;

}

Int isFull(Stack* stack) {

   Return stack->top == MAX – 1;

}

Void push(Stack* stack, int item) {

   If (isFull(stack)) {

      Printf(“Stack is full. Cannot push %d\n”, item);

      Return;

   }

   Stack->items[++stack->top] = item;
```

```
    Printf(“Pushed %d onto the stack.\n”, item);
}
Int pop(Stack* stack) {
    If (isEmpty(stack)) {
        Printf(“Stack is empty. Cannot pop.\n”);
        Exit(1);
    }
    Return stack->items[stack->top--];
}
Int peek(Stack* stack) {
    If (isEmpty(stack)) {
        Printf(“Stack is empty. Cannot peek.\n”);
        Exit(1);
    }
    Return stack->items[stack->top];
}
Int size(Stack* stack) {
    Return stack->top + 1;
}

Int main() {
    Stack stack;
    Initialize(&stack);
    Printf(“Pushing elements onto the stack:\n”);
    Push(&stack, 10);
    Push(&stack, 20);
```

```
   Push(&stack, 30);

   Printf("\nTop element is: %d\n", peek(&stack));

   Printf("\nPopping elements from the stack:\n");

   Printf("Popped element: %d\n", pop(&stack));

   Printf("Popped element: %d\n", pop(&stack));

   Printf("\nIs the stack empty? %s\n", isEmpty(&stack) ? "Yes" : "No");

   Printf("\nPopping the last element from the stack:\n");

   Printf("Popped element: %d\n", pop(&stack));

   Printf("\nIs the stack empty now? %s\n", isEmpty(&stack) ? "Yes" : "No");

   Printf("\nTrying to pop from an empty stack:\n");

   Pop(&stack); // This will exit the program with an error message


   Return 0;

}
```

2.Stack using Linked list

```
#include <stdio.h>

#include <stdlib.h>


Typedef struct Node {

   Int data;

   Struct Node* next;

} Node;


Typedef struct Stack {

   Node* top;
```

```
} Stack;
Void initialize(Stack* stack) {
    Stack->top = NULL;
}

Int isEmpty(Stack* stack) {
    Return stack->top == NULL;
}

Void push(Stack* stack, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    If (!newNode) {
        Printf(“Memory allocation failed. Cannot push %d\n”, data);
        Return;
    }
    newNode->data = data;
    newNode->next = stack->top;
    stack->top = newNode;
    printf(“Pushed %d onto the stack.\n”, data);
}

Int pop(Stack* stack) {
    If (isEmpty(stack)) {
        Printf(“Stack is empty. Cannot pop.\n”);
        Exit(1);
    }
    Node* temp = stack->top;
    Int poppedData = temp->data;
    Stack->top = stack->top->next;
    Free(temp);
    Return poppedData;
}
Int peek(Stack* stack) {
    If (isEmpty(stack)) {
        Printf("Stack is empty. Cannot peek.\n");
        Exit(1);
    }
    Return stack->top->data;
}
Int size(Stack* stack) {
    Int count = 0;
    Node* current = stack->top;
    While (current != NULL) {
        Count++;
        Current = current->next;
    }
    Return count;
}

Int main() {
    Stack stack;
    Initialize(&stack);
    Printf(“Pushing elements onto the stack:\n”);
    Push(&stack, 10);
    Push(&stack, 20);
    Push(&stack, 30);
    Printf(“\nTop element is: %d\n”, peek(&stack));
    Printf(“\nPopping elements from the stack:\n”);
    Printf(“Popped element: %d\n”, pop(&stack));
    Printf(“Popped element: %d\n”, pop(&stack));
    Printf(“\nIs the stack empty? %s\n”, isEmpty(&stack) ? “Yes” : “No”);
    Printf(“\nPopping the last element from the stack:\n”);
    Printf(“Popped element: %d\n”, pop(&stack))
    Printf(“\nIs the stack empty now? %s\n”, isEmpty(&stack) ? “Yes” : “No”);
    Printf(“\nTrying to pop from an empty stack:\n”);
    Pop(&stack); // This will exit the program with an error message

    Return 0;
}
```