**Q. Write a C program to implement stack with an array.**  
  
**Code:**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 100

int stack[MAX\_SIZE];

int top = -1;

void push(int element) {

if (top == MAX\_SIZE - 1) {

printf("Stack is full. Cannot push element.\n");

} else {

top++;

stack[top] = element;

printf("%d pushed onto the stack.\n", element);

}

}

void pop() {

if (top == -1) {

printf("Stack is empty. Cannot pop element.\n");

} else {

printf("%d popped from the stack.\n", stack[top]);

top--;

}

}

void display() {

if (top == -1) {

printf("Stack is empty.\n");

} else {

printf("Elements in the stack: ");

for (int i = 0; i <= top; i++) {

printf("%d ", stack[i]);

}

printf("\n");

}

}

int main() {

int choice, element;

while (1) {

printf("\nStack Operations:\n");

printf("1. Push\n");

printf("2. Pop\n");

printf("3. Display\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the element to push: ");

scanf("%d", &element);

push(element);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

printf("Exiting the program.\n");

exit(0);

default:

printf("Invalid choice. Please try again.\n");

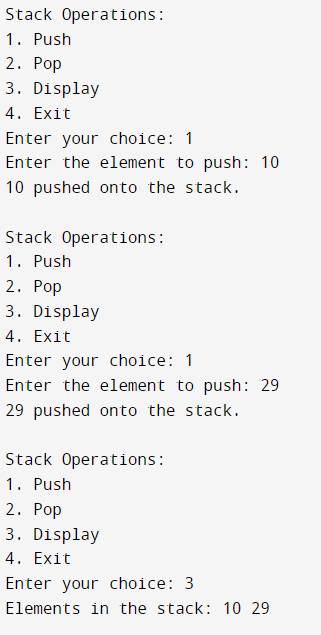
}

}

return 0;

}

**Output:**



**Q. Write a program in C to check whether a given expression containing brackets is balanced or not.**

**Code:**  
#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#define MAX\_SIZE 100

struct Stack {

char data[MAX\_SIZE];

int top;

};

void initialize(struct Stack \*stack) {

stack->top = -1;

}

void push(struct Stack \*stack, char element) {

if (stack->top == MAX\_SIZE - 1) {

printf("Stack is full. Cannot push element.\n");

exit(1);

} else {

stack->data[++stack->top] = element;

}

}

char pop(struct Stack \*stack) {

if (stack->top == -1) {

printf("Stack is empty. Cannot pop element.\n");

exit(1);

} else {

return stack->data[stack->top--];

}

}

bool areBracketsBalanced(const char \*expression) {

struct Stack stack;

initialize(&stack);

for (int i = 0; expression[i] != '\0'; i++) {

char ch = expression[i];

if (ch == '(' || ch == '[' || ch == '{') {

push(&stack, ch);

} else if (ch == ')' || ch == ']' || ch == '}') {

if (stack.top == -1) {

return false;

}

char topElement = pop(&stack);

if ((ch == ')' && topElement != '(') ||

(ch == ']' && topElement != '[') ||

(ch == '}' && topElement != '{')) {

return false;

}

}

}

return stack.top == -1;

}

int main() {

char expression[MAX\_SIZE];

printf("Enter an expression containing brackets: ");

fgets(expression, MAX\_SIZE, stdin);

if (areBracketsBalanced(expression)) {

printf("The expression is balanced.\n");

} else {

printf("The expression is not balanced.\n");

}

return 0;

}

**Output:**



**Q. Write a program in C to reverse a string using stack.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_SIZE 100

struct Stack {

char data[MAX\_SIZE];

int top;

};

void initialize(struct Stack \*stack) {

stack->top = -1;

}

void push(struct Stack \*stack, char element) {

if (stack->top == MAX\_SIZE - 1) {

printf("Stack is full. Cannot push element.\n");

exit(1);

} else {

stack->data[++stack->top] = element;

}

}

char pop(struct Stack \*stack) {

if (stack->top == -1) {

printf("Stack is empty. Cannot pop element.\n");

exit(1);

} else {

return stack->data[stack->top--];

}

}

int main() {

char inputString[MAX\_SIZE];

printf("Enter a string: ");

fgets(inputString, MAX\_SIZE, stdin);

inputString[strcspn(inputString, "\n")] = '\0';

struct Stack stack;

initialize(&stack);

for (int i = 0; inputString[i] != '\0'; i++) {

push(&stack, inputString[i]);

}

printf("Reversed string: ");

while (stack.top != -1) {

printf("%c", pop(&stack));

}

printf("\n");

return 0;

}

**Output:**

