1. **Write a C program to take user input for Stack data type and giver user the option for push, pop, print and exit.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

struct Stack {

int top;

int maxSize;

int \*array;

};

struct Stack \*createStack(int maxSize) {

struct Stack \*stack = (struct Stack \*)malloc(sizeof(struct Stack));

stack->maxSize = maxSize;

stack->top = -1;

stack->array = (int \*)malloc(stack->maxSize \* sizeof(int));

return stack;

}

void push(struct Stack \*stack, int data) {

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

printf("Stack is full\n");

return;

}

stack->top++;

stack->array[stack->top] = data;

}

int pop(struct Stack \*stack) {

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

printf("Stack is empty\n");

return -1;

}

int data = stack->array[stack->top];

stack->top--;

return data;

}

int isEmpty(struct Stack \*stack) {

return stack->top == -1;

}

int isFull(struct Stack \*stack) {

return stack->top == stack->maxSize - 1;

}

void printStack(struct Stack \*stack) {

for (int i = stack->top; i >= 0; i--) {

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

}

printf("\n");

}

int main() {

int maxSize = 10;

struct Stack \*stack = createStack(maxSize);

int choice;

int data;

do {

printf("1. Push\n2. Pop\n3. Print\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter data to push: ");

scanf("%d", &data);

push(stack, data);

break;

case 2:

data = pop(stack);

if (data != -1) {

printf("Popped element: %d\n", data);

}

break;

case 3:

printStack(stack);

break;

case 4:

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

break;

default:

printf("Invalid choice\n");

break;

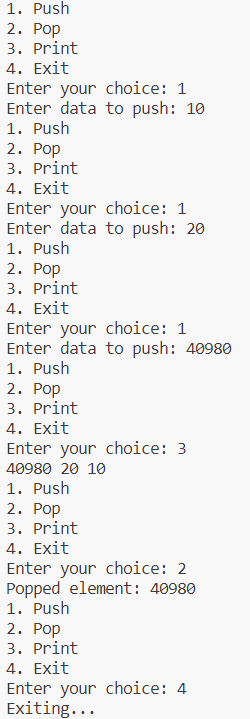
}

} while (choice != 4);

return 0;

}

**Output:**



1. **Write a C program to take user input for parenthesis string and check If it is balanced or not.**

**Code:**

#include <stdio.h>

#include <stdbool.h>

struct Stack {

char data[100];

int top;

};

void initialize(struct Stack \*stack) {

stack->top = -1;

}

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

if (stack->top < 99) {

stack->data[++(stack->top)] = value;

}

}

char pop(struct Stack \*stack) {

if (stack->top >= 0) {

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

}

return '\0';

}

bool isBalanced(char \*str) {

struct Stack stack;

initialize(&stack);

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

if (str[i] == '(' || str[i] == '[' || str[i] == '{') {

push(&stack, str[i]);

} else if (str[i] == ')' || str[i] == ']' || str[i] == '}') {

char popped = pop(&stack);

if ((str[i] == ')' && popped != '(') ||

(str[i] == ']' && popped != '[') ||

(str[i] == '}' && popped != '{')) {

return false;

}

}

}

return stack.top == -1;

}

int main() {

char str[100];

printf("Enter a parenthesis string: ");

scanf("%s", str);

if (isBalanced(str)) {

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

} else {

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

}

return 0;

}

Output:



**Write a C program to print the string given by user in reverse order using stack algorithm.**

**Code:**

#include <stdio.h>

#include <string.h>

#define MAX\_SIZE 100

typedef struct {

char data[MAX\_SIZE];

int top;

} Stack;

void push(Stack \*stack, char data) {

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

printf("Stack overflow!\n");

return;

}

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

}

char pop(Stack \*stack) {

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

printf("Empty stack!\n");

return '\0';

}

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

}

void reverse\_string(char \*str) {

Stack stack;

int len = strlen(str);

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

push(&stack, str[i]);

}

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

str[i] = pop(&stack);

}

}

int main() {

char str[MAX\_SIZE];

printf("Enter a string: ");

fgets(str, MAX\_SIZE, stdin);

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

reverse\_string(str);

printf("Reversed string is: %s\n", str);

return 0;

}

**Output:**

