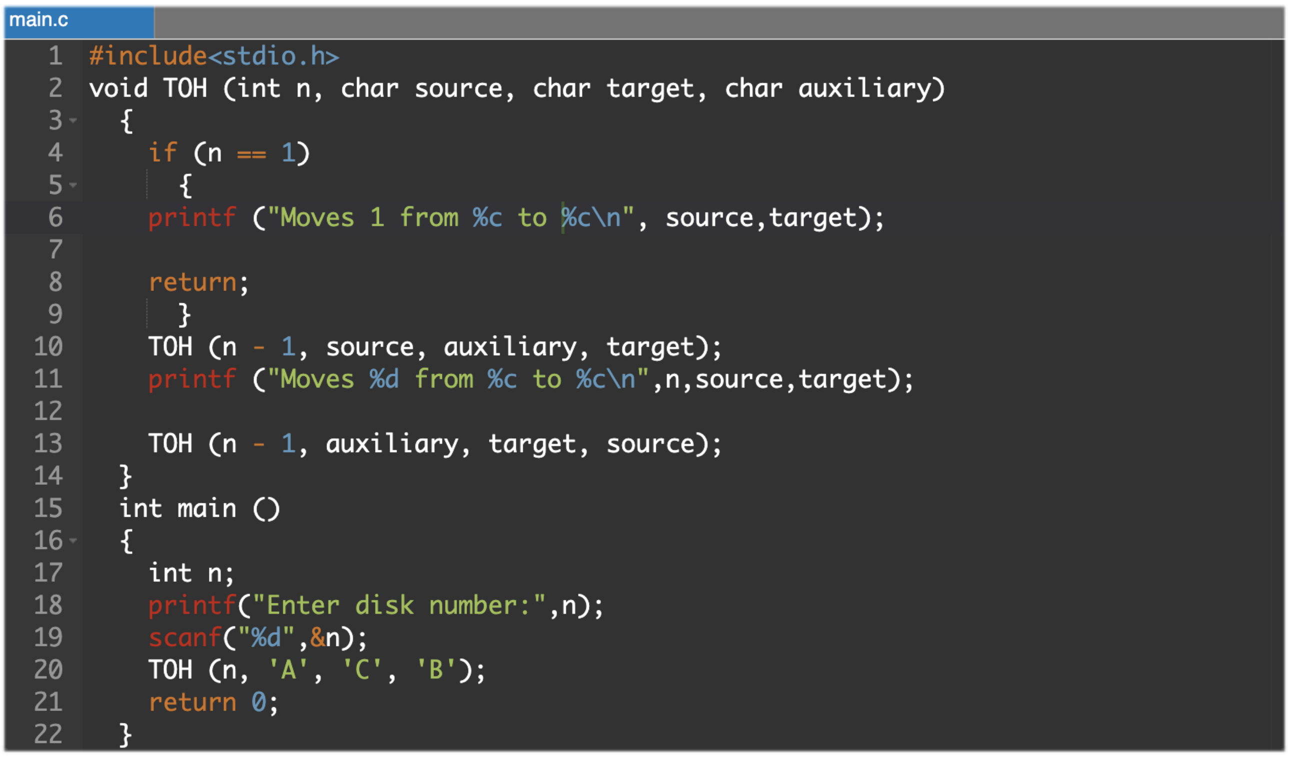
PROBLEM: 1

AIM : To write a program to implement Tower of Hanoi in C programming.

APPARAITOR USED : MAC-OS,ONLINE GDB COMPILER

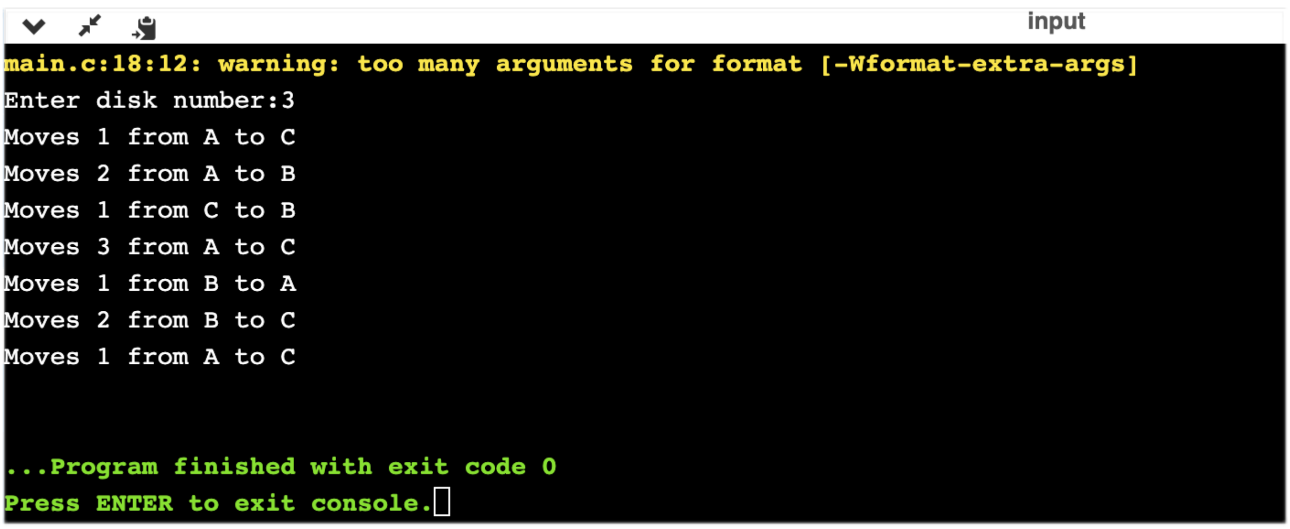
SOURCE CODE:



main.c

```c
#include<stdio.h>
void TOH (int n, char source, char target, char auxiliary)
  {
    if (n == 1)
      {
    printf ("Moves 1 from %c to %c\n", source,target);

    return;
      }
    TOH (n - 1, source, auxiliary, target);
    printf ("Moves %d from %c to %c\n",n,source,target);

    TOH (n - 1, auxiliary, target, source);
  }
  int main ()
  {
    int n;
    printf("Enter disk number:",n);
    scanf("%d",&n);
    TOH (n, 'A', 'C', 'B');
    return 0;
  }
```

OUTPUT:



input

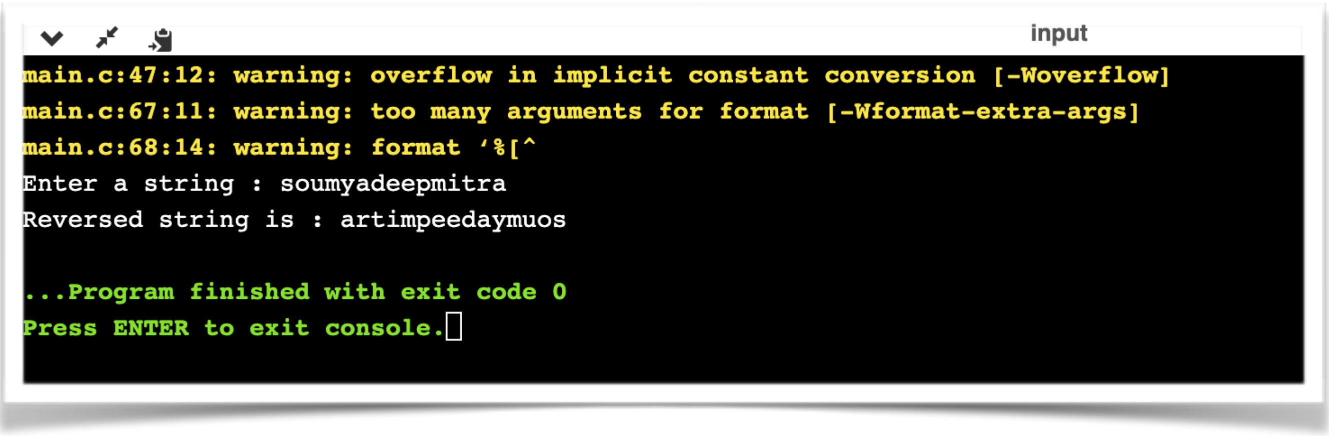
```
main.c:18:12: warning: too many arguments for format [-Wformat-extra-args]
Enter disk number:3
Moves 1 from A to C
Moves 2 from A to B
Moves 1 from C to B
Moves 3 from A to C
Moves 1 from B to A
Moves 2 from B to C
Moves 1 from A to C


...Program finished with exit code 0
Press ENTER to exit console.
```

PROBLEM : 2

AIM : To write a program to implement the reverse string output using stack in C programming.

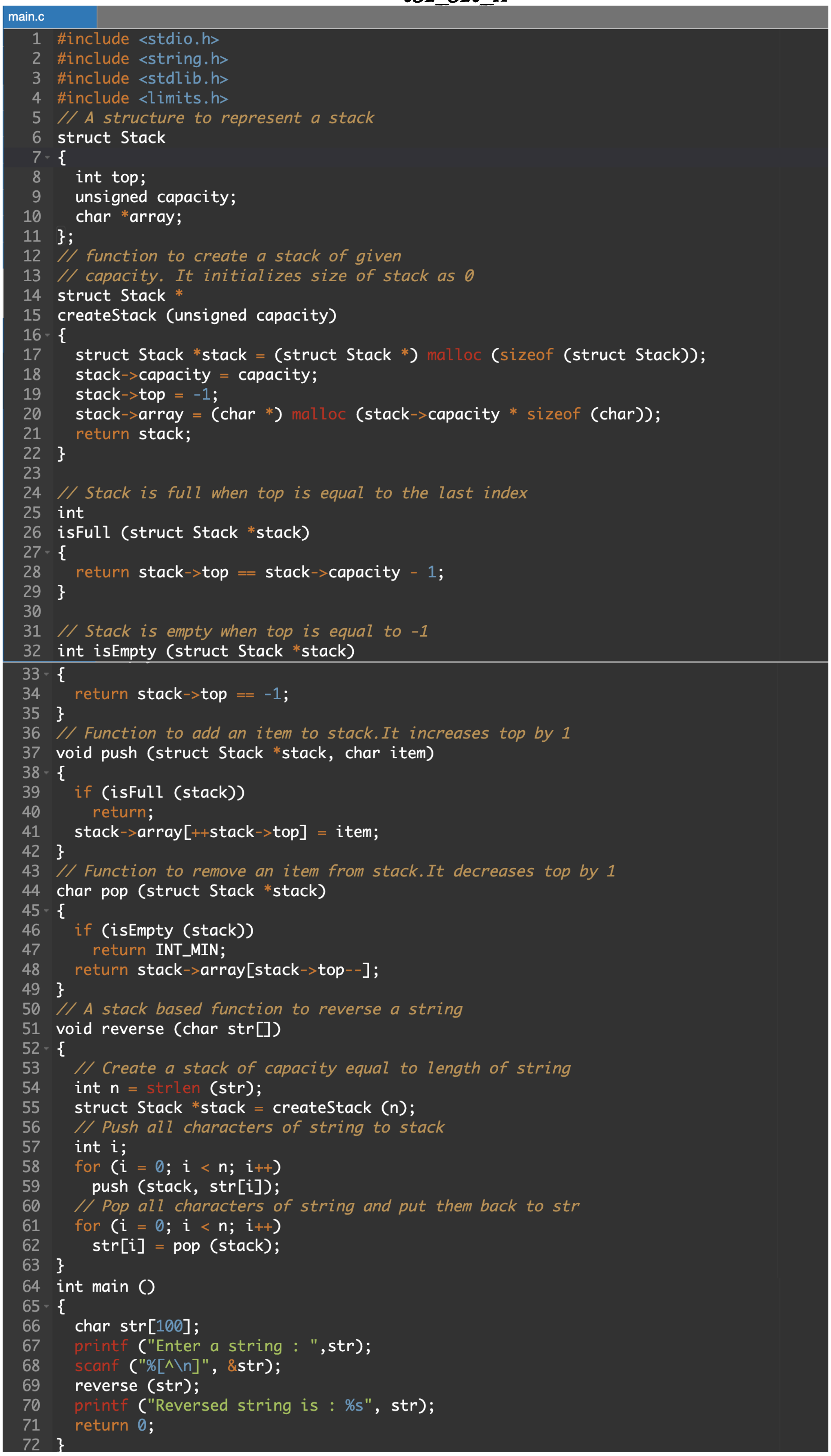
APPARAITOR USED : MAC-OS,ONLINE GDB COMPILER

OUTPUT:



```
main.c:47:12: warning: overflow in implicit constant conversion [-Woverflow]
main.c:67:11: warning: too many arguments for format [-Wformat-extra-args]
main.c:68:14: warning: format '%[^
Enter a string : soumyadeepmitra
Reversed string is : artimpeedaymuos

...Program finished with exit code 0
Press ENTER to exit console.
```

SOURCE CODE:


main.c

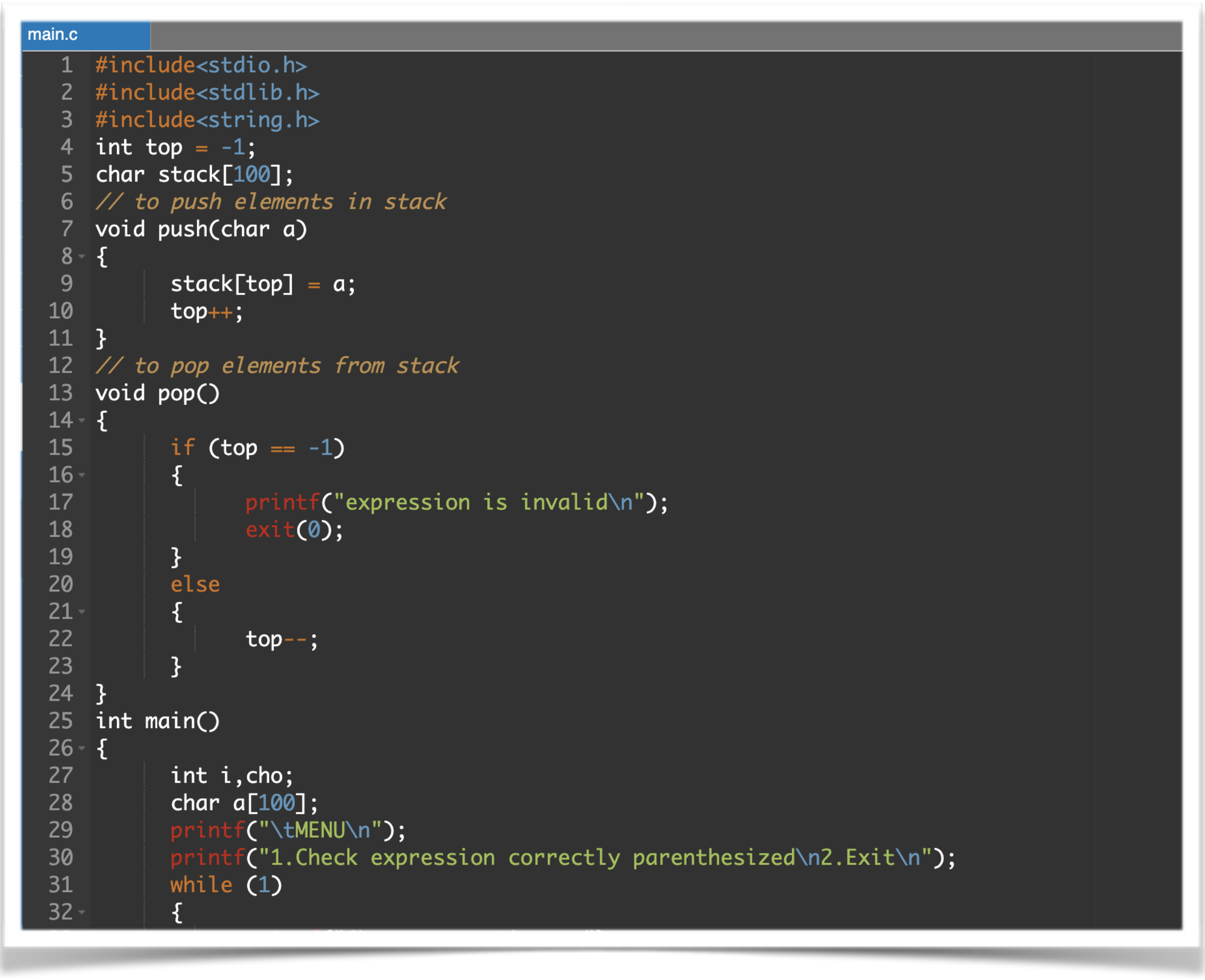
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <limits.h>
// A structure to represent a stack
struct Stack
{
  int top;
  unsigned capacity;
  char *array;
};
// function to create a stack of given
// capacity. It initializes size of stack as 0
struct Stack *
createStack (unsigned capacity)
{
  struct Stack *stack = (struct Stack *) malloc (sizeof (struct Stack));
  stack->capacity = capacity;
  stack->top = -1;
  stack->array = (char *) malloc (stack->capacity * sizeof (char));
  return stack;
}

// Stack is full when top is equal to the last index
int
isFull (struct Stack *stack)
{
  return stack->top == stack->capacity - 1;
}

// Stack is empty when top is equal to -1
int isEmpty (struct Stack *stack)
{
  return stack->top == -1;
}
// Function to add an item to stack.It increases top by 1
void push (struct Stack *stack, char item)
{
  if (isFull (stack))
    return;
  stack->array[++stack->top] = item;
}
// Function to remove an item from stack.It decreases top by 1
char pop (struct Stack *stack)
{
  if (isEmpty (stack))
    return INT_MIN;
  return stack->array[stack->top--];
}
// A stack based function to reverse a string
void reverse (char str[])
{
  // Create a stack of capacity equal to length of string
  int n = strlen (str);
  struct Stack *stack = createStack (n);
  // Push all characters of string to stack
  int i;
  for (i = 0; i < n; i++)
    push (stack, str[i]);
  // Pop all characters of string and put them back to str
  for (i = 0; i < n; i++)
    str[i] = pop (stack);
}
int main ()
{
  char str[100];
  printf ("Enter a string : ",str);
  scanf ("%[^\n]", &str);
  reverse (str);
  printf ("Reversed string is : %s", str);
  return 0;
}
```

PROBLEM 3:

AIM : To write a program to Check if Expression is correctly Parenthesized in C programming.

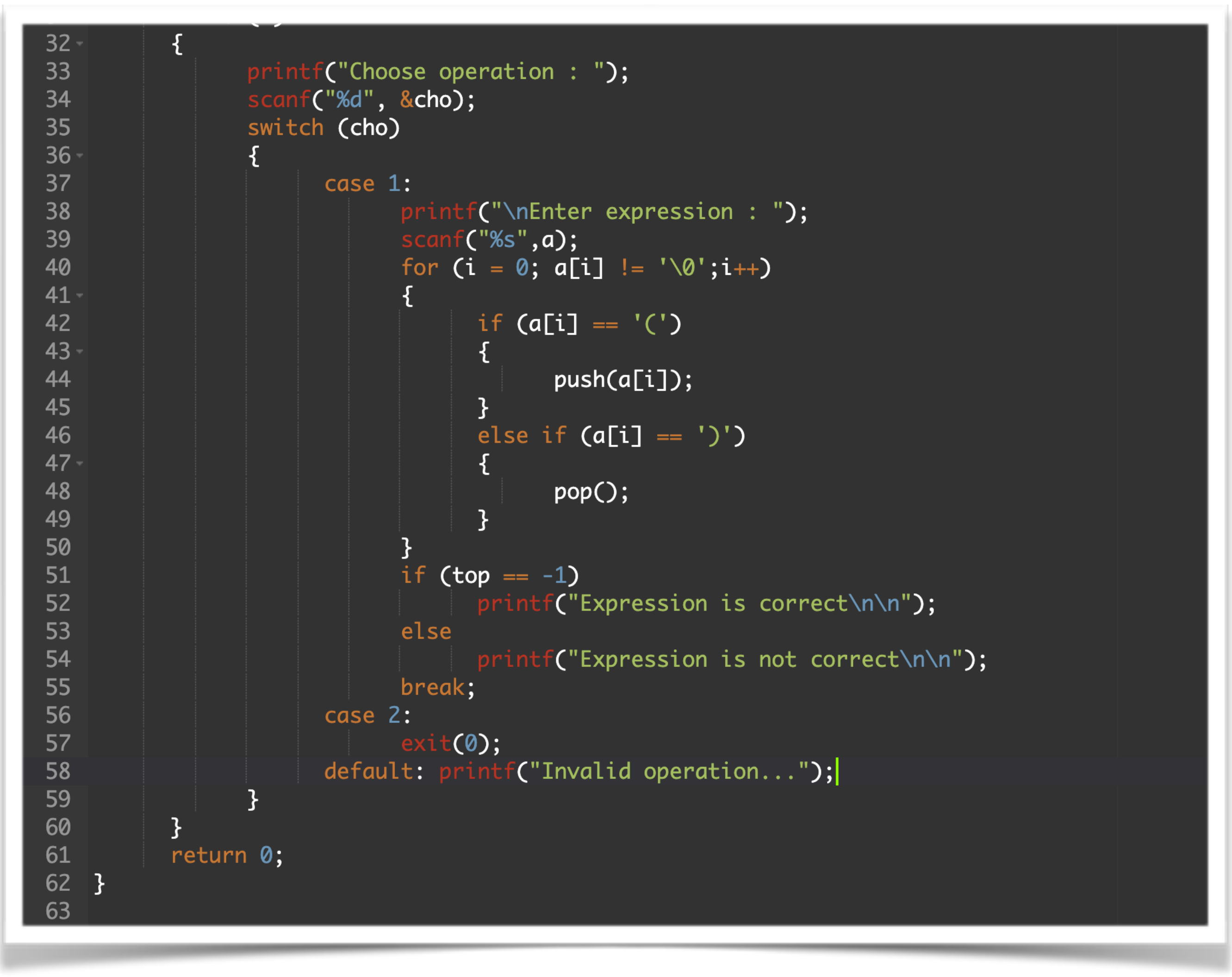
APPARAITOR USED : MAC-OS,ONLINE GDB COMPILER

SOURCE CODE:



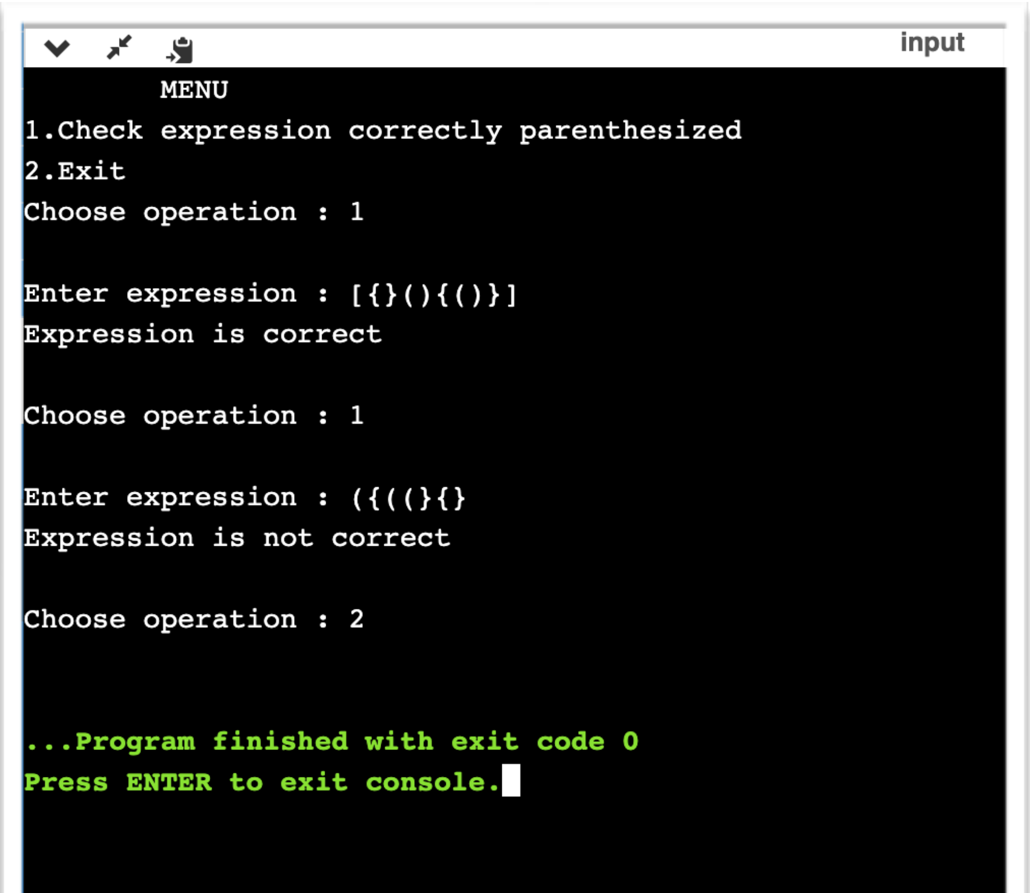
main.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int top = -1;
char stack[100];
// to push elements in stack
void push(char a)
{
	stack[top] = a;
	top++;
}
// to pop elements from stack
void pop()
{
	if (top == -1)
	{
		printf("expression is invalid\n");
		exit(0);
	}
	else
	{
		top--;
	}
}
int main()
{
	int i,cho;
	char a[100];
	printf("\tMENU\n");
	printf("1.Check expression correctly parenthesized\n2.Exit\n");
	while (1)
	{
```



```c
    {
        printf("Choose operation : ");
        scanf("%d", &cho);
        switch (cho)
        {
            case 1:
                printf("\nEnter expression : ");
                scanf("%s",a);
                for (i = 0; a[i] != '\0';i++)
                {
                    if (a[i] == '(')
                    {
                        push(a[i]);
                    }
                    else if (a[i] == ')')
                    {
                        pop();
                    }
                }
                if (top == -1)
                    printf("Expression is correct\n\n");
                else
                    printf("Expression is not correct\n\n");
                break;
            case 2:
                exit(0);
            default: printf("Invalid operation...");
        }
    }
    return 0;
}
```

OUTPUT:



```
        MENU
1.Check expression correctly parenthesized
2.Exit
Choose operation : 1

Enter expression : [{}(){()}]
Expression is correct

Choose operation : 1

Enter expression : ({(()}{}
Expression is not correct

Choose operation : 2


...Program finished with exit code 0
Press ENTER to exit console.
```