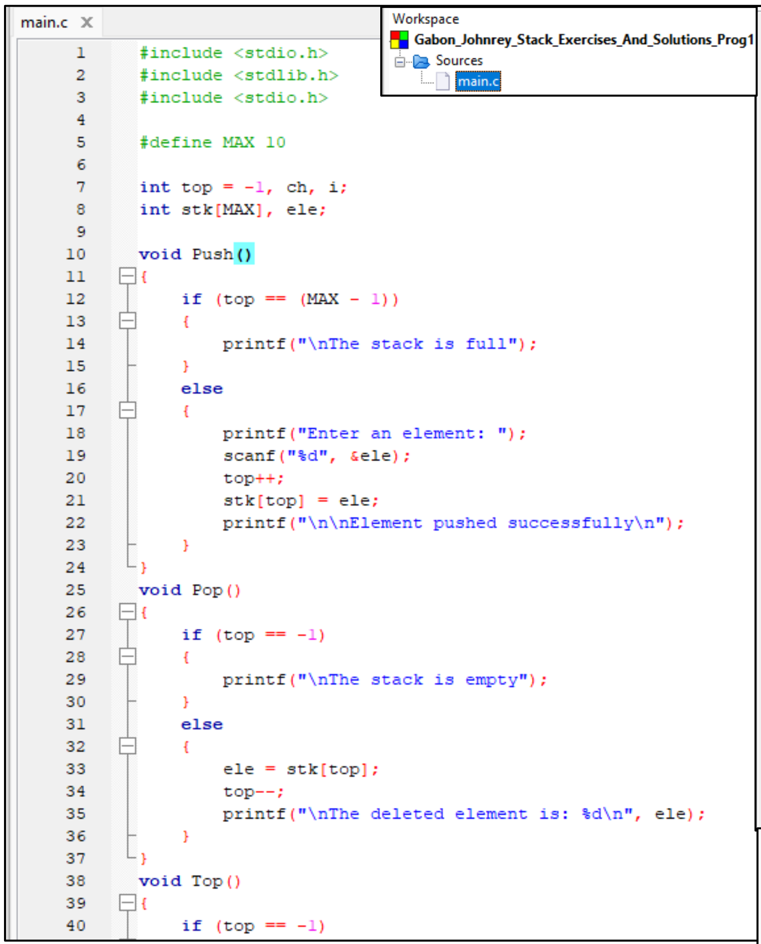
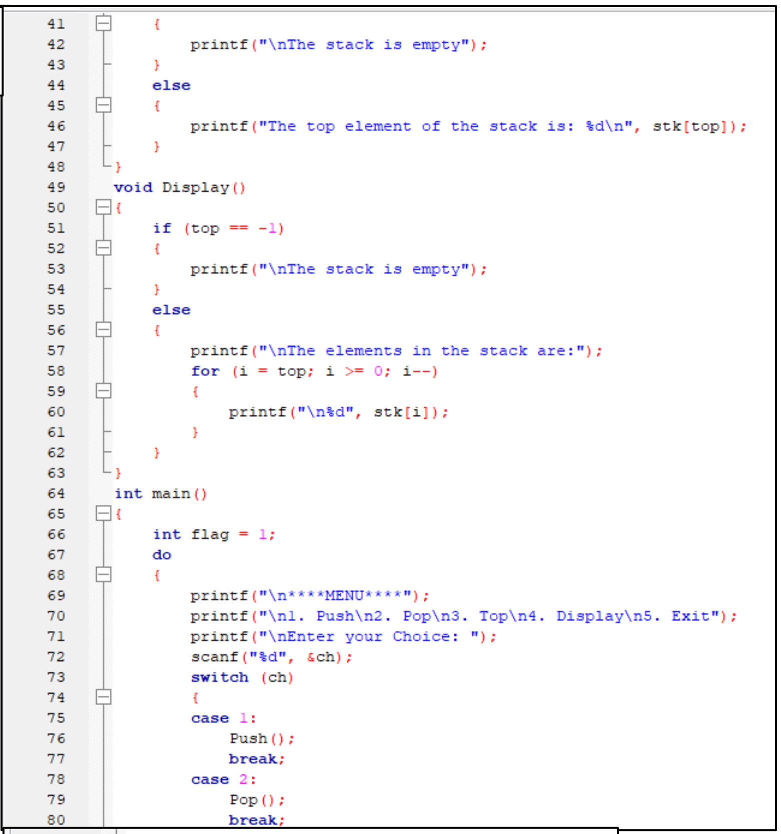
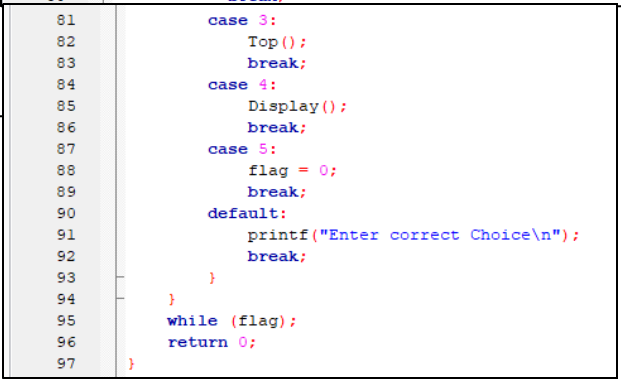
STACK EXERCISES SOLUTIONS 1



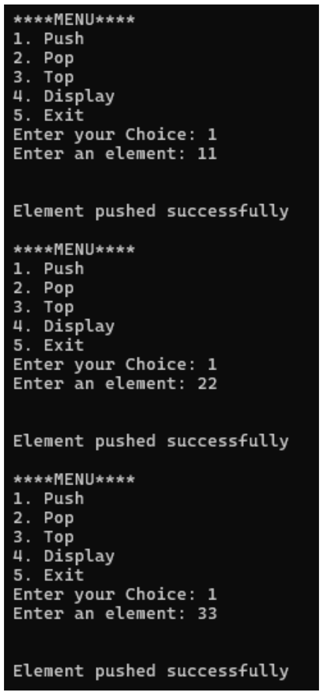
```c
#include <stdio.h>
#include <stdlib.h>
#include <stdio.h>

#define MAX 10

int top = -1, ch, i;
int stk[MAX], ele;

void Push()
{
    if (top == (MAX - 1))
    {
        printf("\nThe stack is full");
    }
    else
    {
        printf("Enter an element: ");
        scanf("%d", &ele);
        top++;
        stk[top] = ele;
        printf("\n\nElement pushed successfully\n");
    }
}
void Pop()
{
    if (top == -1)
    {
        printf("\nThe stack is empty");
    }
    else
    {
        ele = stk[top];
        top--;
        printf("\nThe deleted element is: %d\n", ele);
    }
}
void Top()
{
    if (top == -1)
```



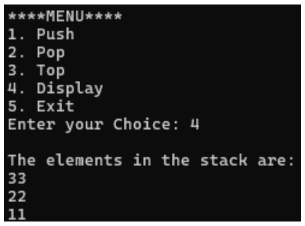
```c
    {
        printf("\nThe stack is empty");
    }
    else
    {
        printf("The top element of the stack is: %d\n", stk[top]);
    }
}
void Display()
{
    if (top == -1)
    {
        printf("\nThe stack is empty");
    }
    else
    {
        printf("\nThe elements in the stack are:");
        for (i = top; i >= 0; i--)
        {
            printf("\n%d", stk[i]);
        }
    }
}
int main()
{
    int flag = 1;
    do
    {
        printf("\n****MENU****");
        printf("\n1. Push\n2. Pop\n3. Top\n4. Display\n5. Exit");
        printf("\nEnter your Choice: ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            Push();
            break;
        case 2:
            Pop();
            break;
```



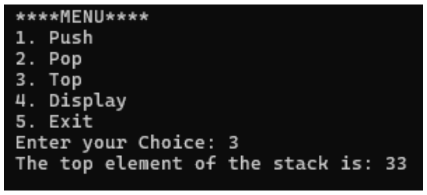
```c
        case 3:
            Top();
            break;
        case 4:
            Display();
            break;
        case 5:
            flag = 0;
            break;
        default:
            printf("Enter correct Choice\n");
            break;
        }
    }
    while (flag);
    return 0;
}
```



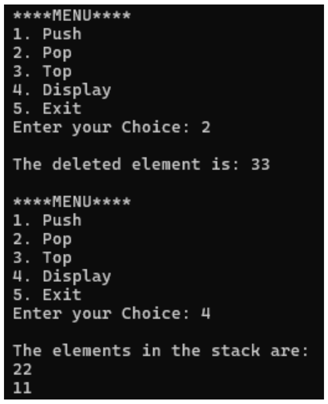
```
****MENU****
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your Choice: 1
Enter an element: 11


Element pushed successfully

****MENU****
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your Choice: 1
Enter an element: 22


Element pushed successfully

****MENU****
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your Choice: 1
Enter an element: 33


Element pushed successfully
```



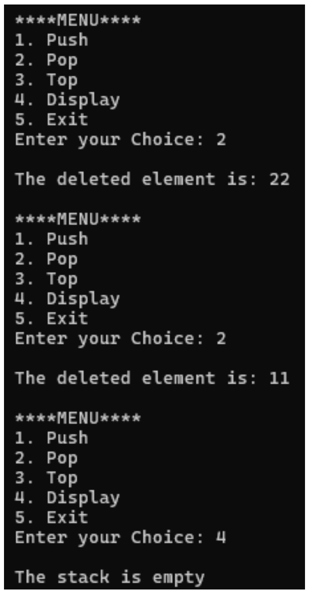
```
****MENU****
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your Choice: 4

The elements in the stack are:
33
22
11
```



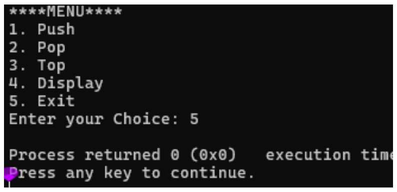
```
****MENU****
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your Choice: 3
The top element of the stack is: 33
```



```
****MENU****
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your Choice: 2

The deleted element is: 33

****MENU****
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your Choice: 4

The elements in the stack are:
22
11
```

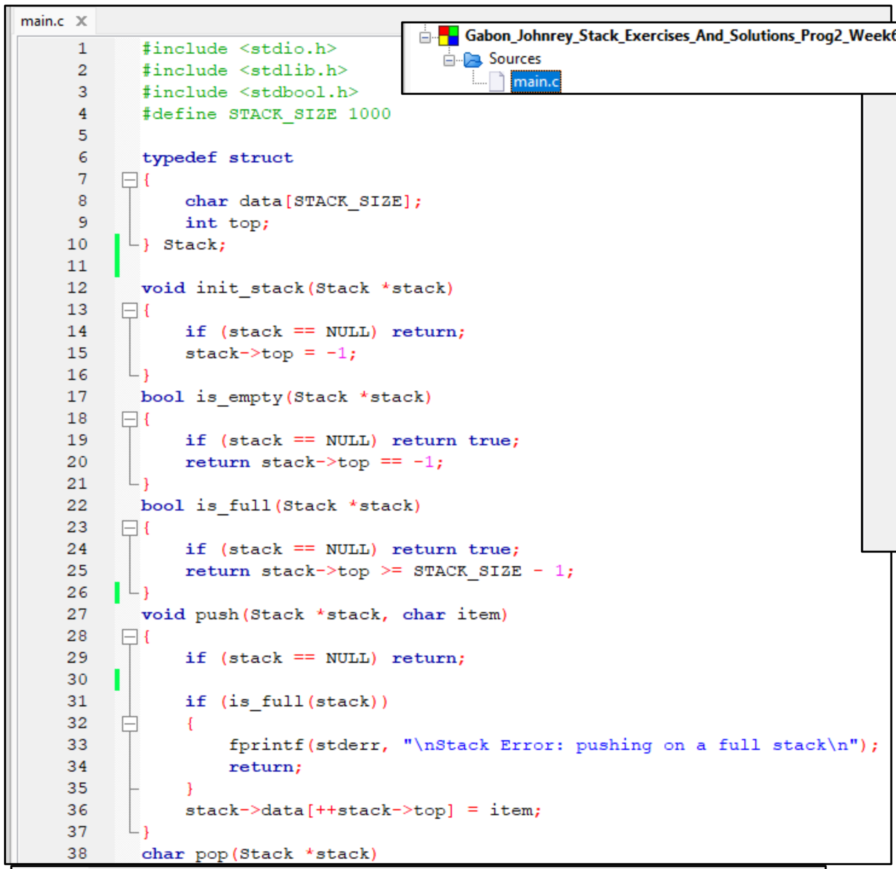


```
****MENU****
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your Choice: 2

The deleted element is: 22

****MENU****
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your Choice: 2

The deleted element is: 11

****MENU****
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your Choice: 4

The stack is empty
```

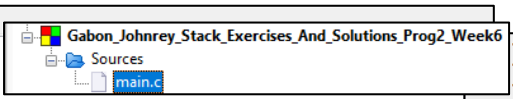


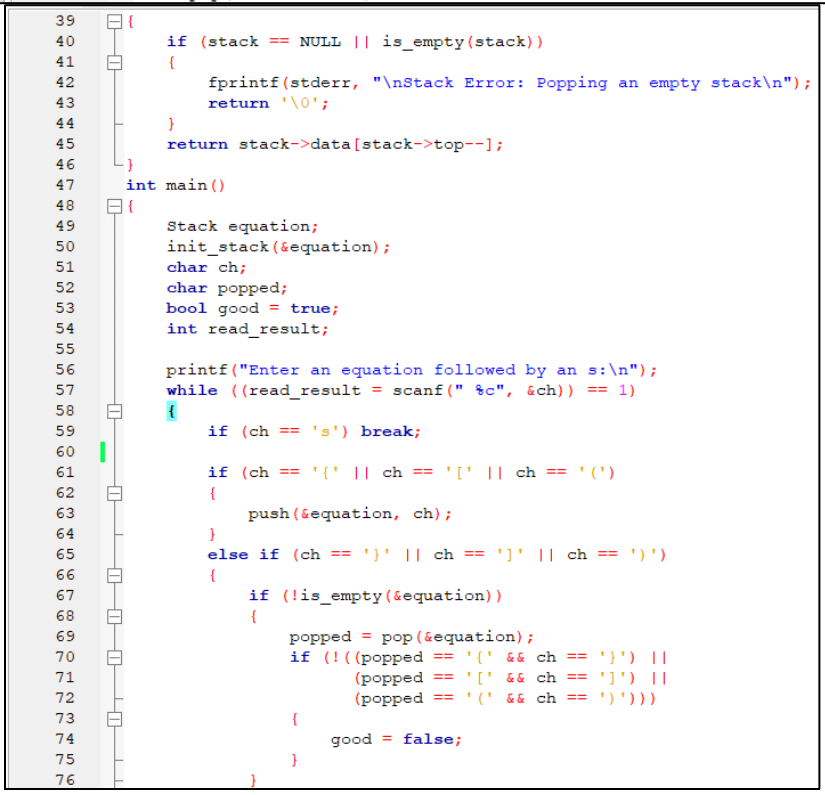
```
****MENU****
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your Choice: 5

Process returned 0 (0x0)   execution tim
Press any key to continue.
```

STACK EXERCISES SOLUTIONS WEEK 6 (1)

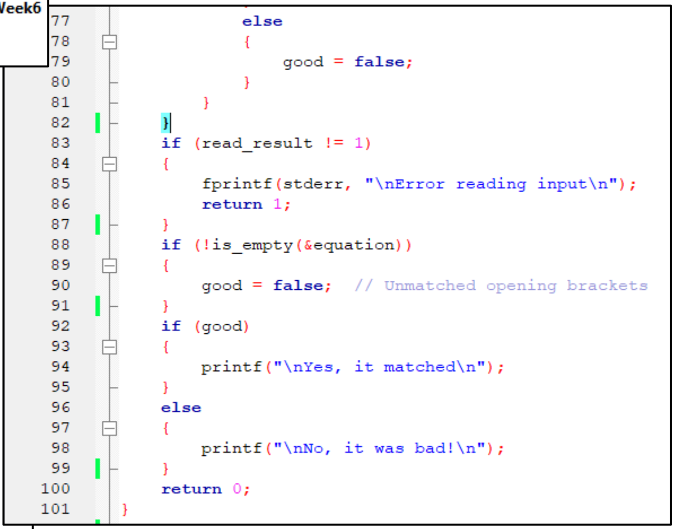


```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define STACK_SIZE 1000

typedef struct
{
    char data[STACK_SIZE];
    int top;
} Stack;

void init_stack(Stack *stack)
{
    if (stack == NULL) return;
    stack->top = -1;
}
bool is_empty(Stack *stack)
{
    if (stack == NULL) return true;
    return stack->top == -1;
}
bool is_full(Stack *stack)
{
    if (stack == NULL) return true;
    return stack->top >= STACK_SIZE - 1;
}
void push(Stack *stack, char item)
{
    if (stack == NULL) return;

    if (is_full(stack))
    {
        fprintf(stderr, "\nStack Error: pushing on a full stack\n");
        return;
    }
    stack->data[++stack->top] = item;
}
char pop(Stack *stack)
```

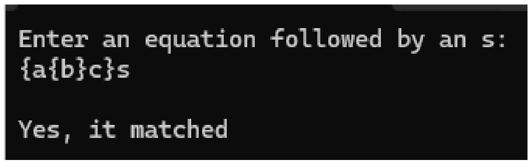




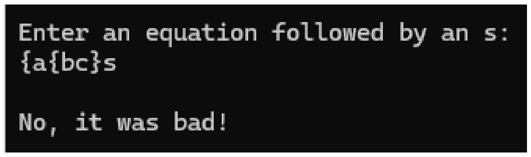
```c
{
    if (stack == NULL || is_empty(stack))
    {
        fprintf(stderr, "\nStack Error: Popping an empty stack\n");
        return '\0';
    }
    return stack->data[stack->top--];
}
int main()
{
    Stack equation;
    init_stack(&equation);
    char ch;
    char popped;
    bool good = true;
    int read_result;

    printf("Enter an equation followed by an s:\n");
    while ((read_result = scanf(" %c", &ch)) == 1)
    {
        if (ch == 's') break;

        if (ch == '{' || ch == '[' || ch == '(')
        {
            push(&equation, ch);
        }
        else if (ch == '}' || ch == ']' || ch == ')')
        {
            if (!is_empty(&equation))
            {
                popped = pop(&equation);
                if (!((popped == '{' && ch == '}') ||
                      (popped == '[' && ch == ']') ||
                      (popped == '(' && ch == ')')))
                {
                    good = false;
                }
            }
```



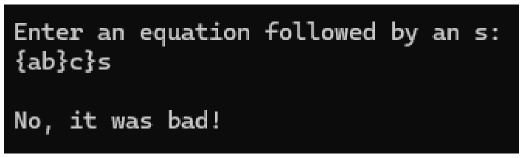
```c
            else
            {
                good = false;
            }
        }
    }
    if (read_result != 1)
    {
        fprintf(stderr, "\nError reading input\n");
        return 1;
    }
    if (!is_empty(&equation))
    {
        good = false;  // Unmatched opening brackets
    }
    if (good)
    {
        printf("\nYes, it matched\n");
    }
    else
    {
        printf("\nNo, it was bad!\n");
    }
    return 0;
}
```



```
Enter an equation followed by an s:
{a{b}c}s

Yes, it matched
```

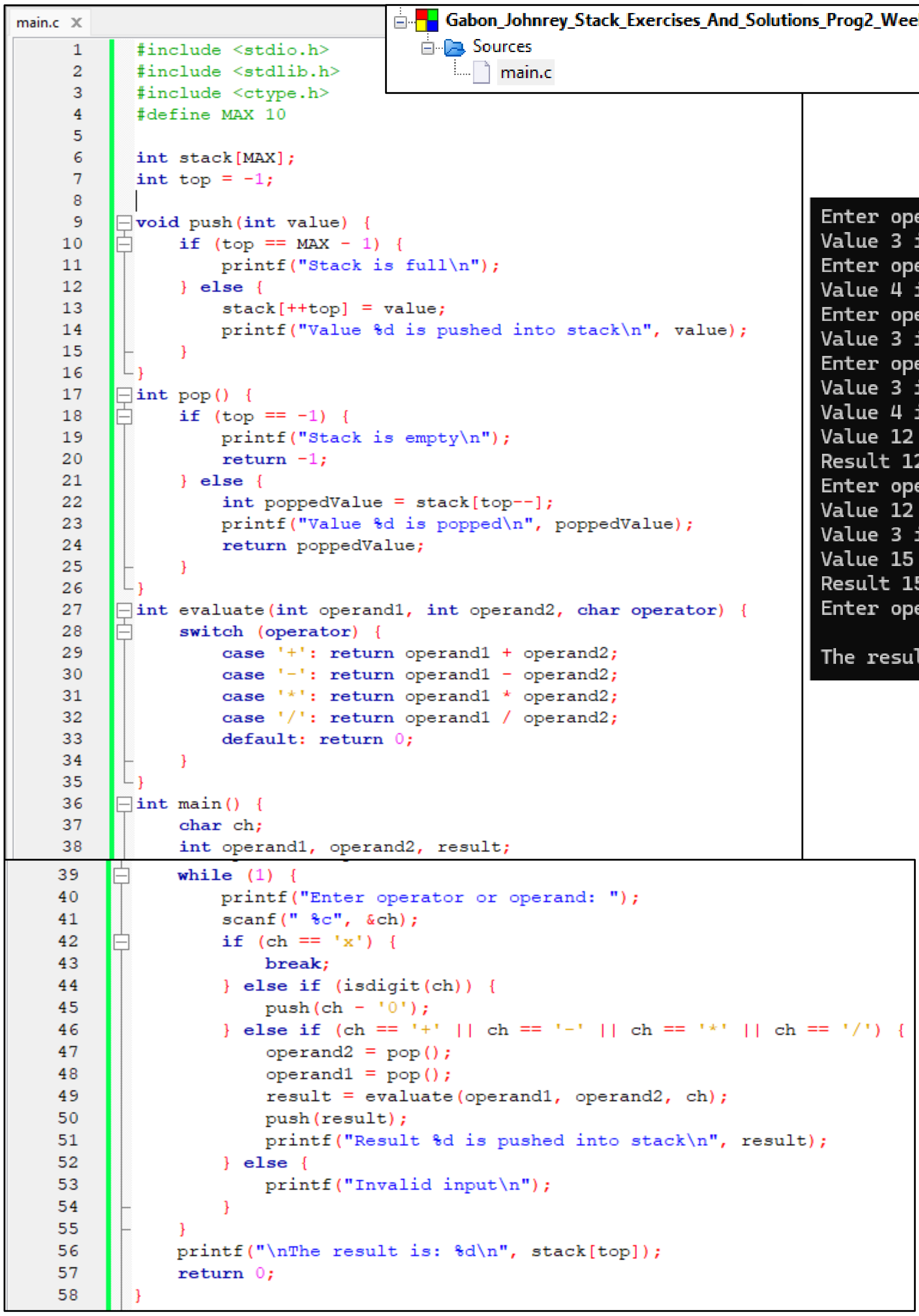


```
Enter an equation followed by an s:
{a{bc}s

No, it was bad!
```



```
Enter an equation followed by an s:
{ab}c}s

No, it was bad!
```

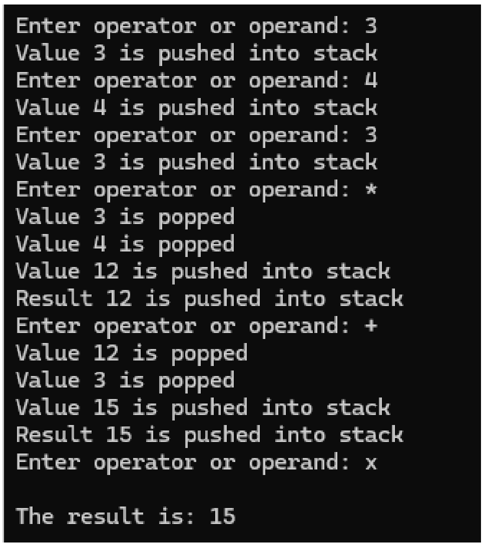
STACK EXERCISES SOLUTIONS WEEK 7 (1)



main.c

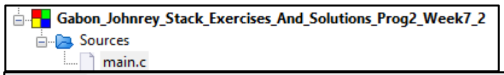
```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define MAX 10

int stack[MAX];
int top = -1;

void push(int value) {
    if (top == MAX - 1) {
        printf("Stack is full\n");
    } else {
        stack[++top] = value;
        printf("Value %d is pushed into stack\n", value);
    }
}
int pop() {
    if (top == -1) {
        printf("Stack is empty\n");
        return -1;
    } else {
        int poppedValue = stack[top--];
        printf("Value %d is popped\n", poppedValue);
        return poppedValue;
    }
}
int evaluate(int operand1, int operand2, char operator) {
    switch (operator) {
        case '+': return operand1 + operand2;
        case '-': return operand1 - operand2;
        case '*': return operand1 * operand2;
        case '/': return operand1 / operand2;
        default: return 0;
    }
}
int main() {
    char ch;
    int operand1, operand2, result;
    while (1) {
        printf("Enter operator or operand: ");
        scanf(" %c", &ch);
        if (ch == 'x') {
            break;
        } else if (isdigit(ch)) {
            push(ch - '0');
        } else if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
            operand2 = pop();
            operand1 = pop();
            result = evaluate(operand1, operand2, ch);
            push(result);
            printf("Result %d is pushed into stack\n", result);
        } else {
            printf("Invalid input\n");
        }
    }
    printf("\nThe result is: %d\n", stack[top]);
    return 0;
}
```

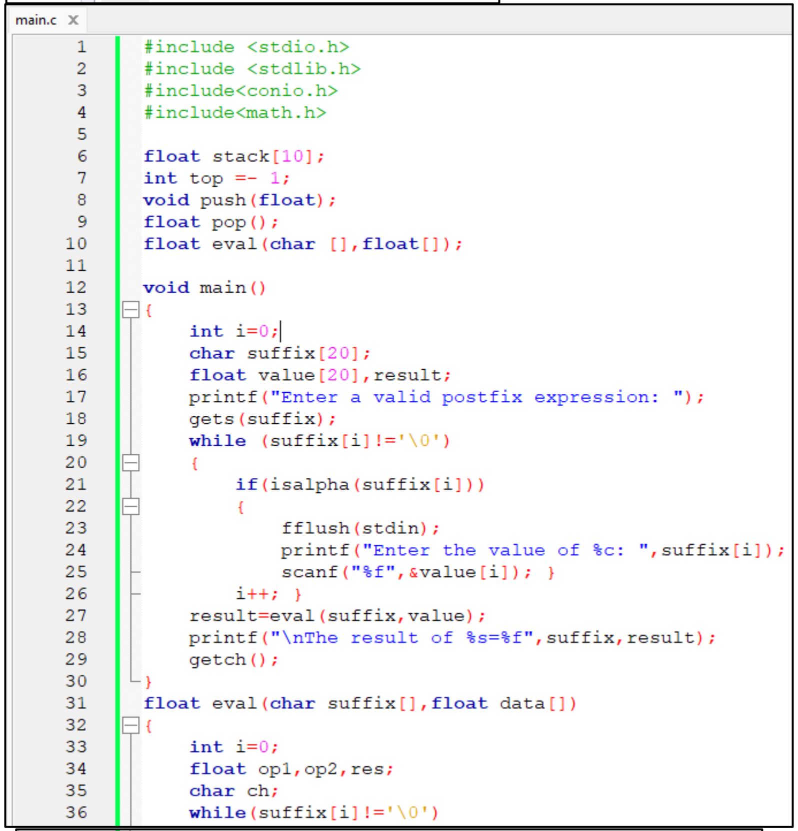
Gabon_Johnrey_Stack_Exercises_And_Solutions_Prog2_Week7_1
Sources
main.c



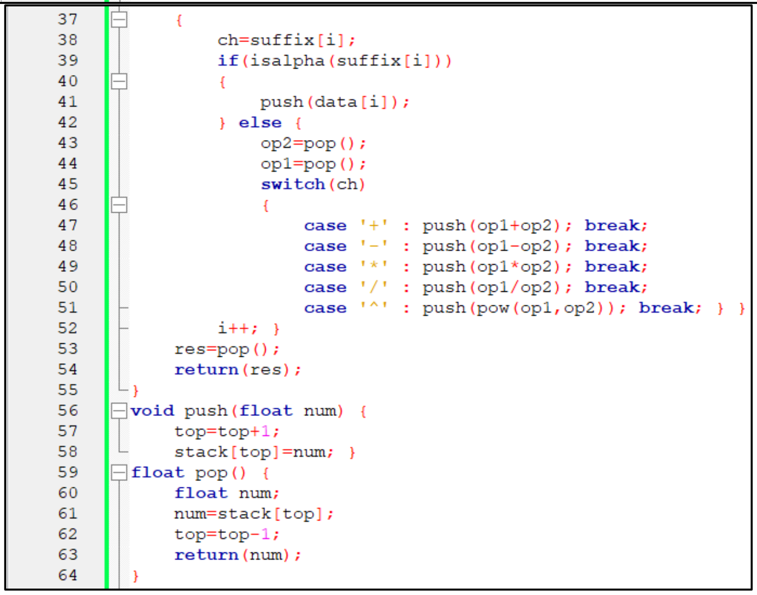
```
Enter operator or operand: 3
Value 3 is pushed into stack
Enter operator or operand: 4
Value 4 is pushed into stack
Enter operator or operand: 3
Value 3 is pushed into stack
Enter operator or operand: *
Value 3 is popped
Value 4 is popped
Value 12 is pushed into stack
Result 12 is pushed into stack
Enter operator or operand: +
Value 12 is popped
Value 3 is popped
Value 15 is pushed into stack
Result 15 is pushed into stack
Enter operator or operand: x

The result is: 15
```

STACK EXERCISES SOLUTIONS WEEK 7 (2)

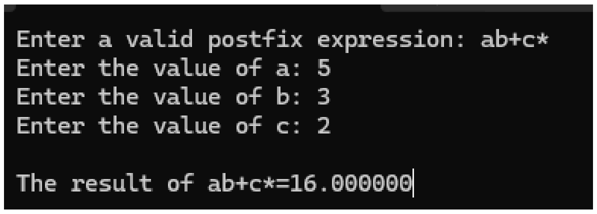




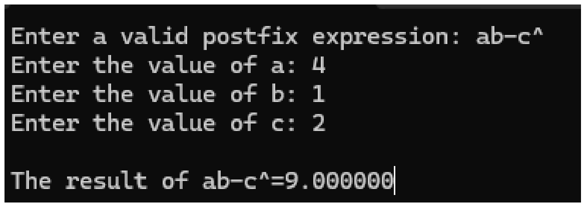
```c
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
#include<math.h>

float stack[10];
int top =- 1;
void push(float);
float pop();
float eval(char [],float[]);

void main()
{
    int i=0;
    char suffix[20];
    float value[20],result;
    printf("Enter a valid postfix expression: ");
    gets(suffix);
    while (suffix[i]!='\0')
    {
        if(isalpha(suffix[i]))
        {
            fflush(stdin);
            printf("Enter the value of %c: ",suffix[i]);
            scanf("%f",&value[i]); }
        i++; }
    result=eval(suffix,value);
    printf("\nThe result of %s=%f",suffix,result);
    getch();
}
float eval(char suffix[],float data[])
{
    int i=0;
    float op1,op2,res;
    char ch;
    while(suffix[i]!='\0')
```



```c
    {
        ch=suffix[i];
        if(isalpha(suffix[i]))
        {
            push(data[i]);
        } else {
            op2=pop();
            op1=pop();
            switch(ch)
            {
                case '+' : push(op1+op2); break;
                case '-' : push(op1-op2); break;
                case '*' : push(op1*op2); break;
                case '/' : push(op1/op2); break;
                case '^' : push(pow(op1,op2)); break; } }
        i++; }
    res=pop();
    return(res);
}
void push(float num) {
    top=top+1;
    stack[top]=num; }
float pop() {
    float num;
    num=stack[top];
    top=top-1;
    return(num);
}
```



```
Enter a valid postfix expression: ab+c*
Enter the value of a: 5
Enter the value of b: 3
Enter the value of c: 2

The result of ab+c*=16.000000
```



```
Enter a valid postfix expression: ab-c^
Enter the value of a: 4
Enter the value of b: 1
Enter the value of c: 2

The result of ab-c^=9.000000
```