

STACK EXERCISES SOLUTIONS 1

```

main.c X
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  #define MAX 10
6
7  int top = -1, ch, i;
8  int stk[MAX], ele;
9
10 void Push()
11 {
12     if (top == (MAX - 1))
13     {
14         printf("\nThe stack is full");
15     }
16     else
17     {
18         printf("Enter an element: ");
19         scanf("%d", &ele);
20         top++;
21         stk[top] = ele;
22         printf("\n\nElement pushed successfully\n");
23     }
24 }
25 void Pop()
26 {
27     if (top == -1)
28     {
29         printf("\nThe stack is empty");
30     }
31     else
32     {
33         ele = stk[top];
34         top--;
35         printf("\nThe deleted element is: %d\n", ele);
36     }
37 }
38 void Top()
39 {
40     if (top == -1)

```

```

41 {
42     printf("\nThe stack is empty");
43 }
44 else
45 {
46     printf("The top element of the stack is: %d\n", stk[top]);
47 }
48 }
49 void Display()
50 {
51     if (top == -1)
52     {
53         printf("\nThe stack is empty");
54     }
55     else
56     {
57         printf("\nThe elements in the stack are:");
58         for (i = top; i >= 0; i--)
59         {
60             printf("\n%d", stk[i]);
61         }
62     }
63 }
64 int main()
65 {
66     int flag = 1;
67     do
68     {
69         printf("\n****MENU****");
70         printf("\n1. Push\n2. Pop\n3. Top\n4. Display\n5. Exit");
71         printf("\nEnter your Choice: ");
72         scanf("%d", &ch);
73         switch (ch)
74         {
75             case 1:
76                 Push();
77                 break;
78             case 2:
79                 Pop();
80                 break;
81             case 3:
82                 Top();
83                 break;
84             case 4:
85                 Display();
86                 break;
87             case 5:
88                 flag = 0;
89                 break;
90             default:
91                 printf("Enter correct Choice\n");
92                 break;
93         }
94     }
95     while (flag);
96     return 0;
97 }

```

```

****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: 4

The elements in the stack are:
33
22
11

```

```

****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: 3

The top element of the stack is: 33

```

```

****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: 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 time
Press any key to continue.

```

STACK EXERCISES SOLUTIONS WEEK 6 (1)

```

main.c X
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdbool.h>
4  #define STACK_SIZE 1000
5
6  typedef struct
7  {
8      char data[STACK_SIZE];
9      int top;
10 } Stack;
11
12 void init_stack(Stack *stack)
13 {
14     if (stack == NULL) return;
15     stack->top = -1;
16 }
17
18 bool is_empty(Stack *stack)
19 {
20     if (stack == NULL) return true;
21     return stack->top == -1;
22 }
23
24 bool is_full(Stack *stack)
25 {
26     if (stack == NULL) return true;
27     return stack->top >= STACK_SIZE - 1;
28 }
29
30 void push(Stack *stack, char item)
31 {
32     if (stack == NULL) return;
33     if (is_full(stack))
34     {
35         fprintf(stderr, "\nStack Error: pushing on a full stack\n");
36         return;
37     }
38     stack->data[++stack->top] = item;
39
40 char pop(Stack *stack)

```

```

39 {
40     if (stack == NULL || is_empty(stack))
41     {
42         fprintf(stderr, "\nStack Error: Popping an empty stack\n");
43         return '\0';
44     }
45     return stack->data[stack->top--];
46 }
47
48 int main()
49 {
50     Stack equation;
51     init_stack(&equation);
52     char ch;
53     char popped;
54     bool good = true;
55     int read_result;
56
57     printf("Enter an equation followed by an s:\n");
58     while ((read_result = scanf(" %c", &ch)) == 1)
59     {
60         if (ch == 's') break;
61
62         if (ch == '{' || ch == '[' || ch == '(')
63         {
64             push(&equation, ch);
65         }
66         else if (ch == '}' || ch == ']' || ch == ')')
67         {
68             if (!is_empty(&equation))
69             {
70                 popped = pop(&equation);
71                 if (!((popped == '{' && ch == '}') ||
72                     (popped == '[' && ch == ']') ||
73                     (popped == '(' && ch == ')')))
74                 {
75                     good = false;
76                 }
77             }
78         }
79     }
80
81     if (good)
82     {
83         printf("\nYes, it matched\n");
84     }
85     else
86     {
87         printf("\nNo, it was bad!\n");
88     }
89     return 0;
90 }

```

```

77     else
78     {
79         good = false;
80     }
81 }
82
83 if (read_result != 1)
84 {
85     fprintf(stderr, "\nError reading input\n");
86     return 1;
87 }
88 if (!is_empty(&equation))
89 {
90     good = false; // Unmatched opening brackets
91 }
92 if (good)
93 {
94     printf("\nYes, it matched\n");
95 }
96 else
97 {
98     printf("\nNo, it was bad!\n");
99 }
100 return 0;
101 }

```

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

Yes, it matched

Enter an equation followed by an s:
{a{b}c}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 X
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4  #define MAX 10
5
6  int stack[MAX];
7  int top = -1;
8
9  void push(int value) {
10     if (top == MAX - 1) {
11         printf("Stack is full\n");
12     } else {
13         stack[++top] = value;
14         printf("Value %d is pushed into stack\n", value);
15     }
16 }
17 int pop() {
18     if (top == -1) {
19         printf("Stack is empty\n");
20         return -1;
21     } else {
22         int poppedValue = stack[top--];
23         printf("Value %d is popped\n", poppedValue);
24         return poppedValue;
25     }
26 }
27 int evaluate(int operand1, int operand2, char operator) {
28     switch (operator) {
29         case '+': return operand1 + operand2;
30         case '-': return operand1 - operand2;
31         case '*': return operand1 * operand2;
32         case '/': return operand1 / operand2;
33         default: return 0;
34     }
35 }
36 int main() {
37     char ch;
38     int operand1, operand2, result;
39
40     while (1) {
41         printf("Enter operator or operand: ");
42         scanf(" %c", &ch);
43         if (ch == 'x') {
44             break;
45         } else if (isdigit(ch)) {
46             push(ch - '0');
47         } else if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
48             operand2 = pop();
49             operand1 = pop();
50             result = evaluate(operand1, operand2, ch);
51             push(result);
52             printf("Result %d is pushed into stack\n", result);
53         } else {
54             printf("Invalid input\n");
55         }
56     }
57     printf("\nThe result is: %d\n", stack[top]);
58     return 0;
59 }
```

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)

```

Gabon_Johnrey_Stack_Exercises_And_Solutions_Prog2_Week7_2
Sources
main.c
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <conio.h>
4  #include <math.h>
5
6  float stack[10];
7  int top = -1;
8  void push(float);
9  float pop();
10 float eval(char [], float[]);
11
12 void main()
13 {
14     int i=0;
15     char suffix[20];
16     float value[20], result;
17     printf("Enter a valid postfix expression: ");
18     gets(suffix);
19     while (suffix[i]!='\0')
20     {
21         if (isalpha(suffix[i]))
22         {
23             fflush(stdin);
24             printf("Enter the value of %c: ", suffix[i]);
25             scanf("%f", &value[i]);
26             i++;
27         }
28         result = eval(suffix, value);
29         printf("\nThe result of %s=%f", suffix, result);
30         getch();
31     }
32     float eval(char suffix[], float data[])
33     {
34         int i=0;
35         float op1, op2, res;
36         char ch;
37         while (suffix[i]!='\0')
38         {
39             ch = suffix[i];
40             if (isalpha(suffix[i]))
41             {
42                 push(data[i]);
43             } else {
44                 op2 = pop();
45                 op1 = pop();
46                 switch (ch)
47                 {
48                     case '+': push(op1+op2); break;
49                     case '-': push(op1-op2); break;
50                     case '*': push(op1*op2); break;
51                     case '/': push(op1/op2); break;
52                     case '^': push(pow(op1, op2)); break;
53                 }
54                 i++;
55             }
56             res = pop();
57             return (res);
58         }
59     }
60     void push(float num) {
61         top = top + 1;
62         stack[top] = num;
63     }
64     float pop() {
65         float num;
66         num = stack[top];
67         top = top - 1;
68         return (num);
69     }

```

```

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
```