**2.Stack - Parenthesis**

**1.**

#include<stdio.h>

#include<stdlib.h>

const int N = 20;

struct stack {

int arr[N];

int size;

};

void stackInit(struct stack& koko) {

koko.size = 0;

}

int empty(struct stack& koko) {

if (koko.size == 0)

return 1;

else

return 0;

}

void push(struct stack& koko, int x) {

if (koko.size == N) {

printf("stack full!");

}

if (koko.size < N && koko.size > -1) {

koko.arr[koko.size] = x;

koko.size++;

}

}

int stacktop(struct stack& koko) {

if (empty(koko) == 0)

return koko.arr[koko.size - 1];

}

int pop(struct stack& koko) {

if (empty(koko) == 0) {

int x = koko.arr[koko.size - 1];

koko.size = koko.size - 1;

return x;

}

}

void print(struct stack& koko) {

for (int i = 0; i < koko.size; i++) {

printf("%d,", koko.arr[i]);

}

}

void Delete(struct stack& koko, int x) {

struct stack s2;

stackInit(s2);

int counter = 0;

if (empty(koko) == 0)

{

while (stacktop(koko) != x) {

push(s2, pop(koko));

counter++;

}

pop(koko);

while (counter > 0) {

push(koko, pop(s2));

counter--;

}

}

else

printf("the stack is empty!");

}

int search(struct stack& koko, int x) {

struct stack s2;

stackInit(s2);

int counter = 0;

if (empty(koko) == 0) {

while (stacktop(koko) != x) {

push(s2, pop(koko));

counter++;

}

}

int j = counter;

while (counter > 0) {

push(koko, pop(s2));

counter--;

}

return j + 1;

}

void main() {

struct stack s;

stackInit(s);

char exp[20];

scanf\_s("%s", exp, 20);

int i = 0;

while (exp[i] != '\0') {

char c = exp[i];

if (c == '+' || c == '-' || c == '\*' || c == '/') {

int x = pop(s);

if (c == '+') {

push(s, pop(s) + x);

}

if (c == '-') {

push(s, pop(s) - x);

}

if (c == '\*') {

push(s, pop(s) \* x);

}

if (c == '/') {

push(s, pop(s) / x);

}

}

else {

int variable = atoi(&c);

push(s, variable);

}

i++;

}

print(s);

}

**2.**

#include<stdio.h>

#include<stdlib.h>

const int N = 20;

struct stack {

int arr[N];

int size;

};

void stackInit(struct stack& koko) {

koko.size = 0;

}

int empty(struct stack& koko) {

if (koko.size == 0)

return 1;

else

return 0;

}

void push(struct stack& koko, int x) {

if (koko.size == N) {

printf("stack full!");

}

if (koko.size < N && koko.size > -1) {

koko.arr[koko.size] = x;

koko.size++;

}

}

int stacktop(struct stack& koko) {

if (empty(koko) == 0)

return koko.arr[koko.size - 1];

}

int pop(struct stack& koko) {

if (empty(koko) == 0) {

int x = koko.arr[koko.size - 1];

koko.size = koko.size - 1;

return x;

}

}

void print(struct stack& koko) {

for (int i = 0; i < koko.size; i++) {

printf("%d,", koko.arr[i]);

}

}

void Delete(struct stack& koko, int x) {

struct stack s2;

stackInit(s2);

int counter = 0;

if (empty(koko) == 0)

{

while (stacktop(koko) != x) {

push(s2, pop(koko));

counter++;

}

pop(koko);

while (counter > 0) {

push(koko, pop(s2));

counter--;

}

}

else

printf("the stack is empty!");

}

int search(struct stack& koko, int x) {

struct stack s2;

stackInit(s2);

int counter = 0;

if (empty(koko) == 0) {

while (stacktop(koko) != x) {

push(s2, pop(koko));

counter++;

}

}

int j = counter;

while (counter > 0) {

push(koko, pop(s2));

counter--;

}

return j + 1;

}

void main() {

struct stack s;

stackInit(s);

char exp[20];

scanf\_s("%s", exp, 20);

int i = 0;

int start = 0;

int close = 0;

int zug = 0;

while (exp[i] != '\0') {

char c = exp[i];

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

push(s, c);

start++;

}

if (c == ')' || c == '}' || c == ']') {

if (c == ')') {

close++;

if (stacktop(s) == '(') {

pop(s);

zug++;

}

}

if (c == '}') {

close++;

if (stacktop(s) == '{') {

pop(s);

zug++;

}

}

if (c == ']') {

close++;

if (stacktop(s) == '[') {

pop(s);

zug++;

}

}

}

i++;

}

if (start == close && zug == start) {

printf("all good!");

}

else {

printf("Error");

}

}