第二章作业

1.（1）是

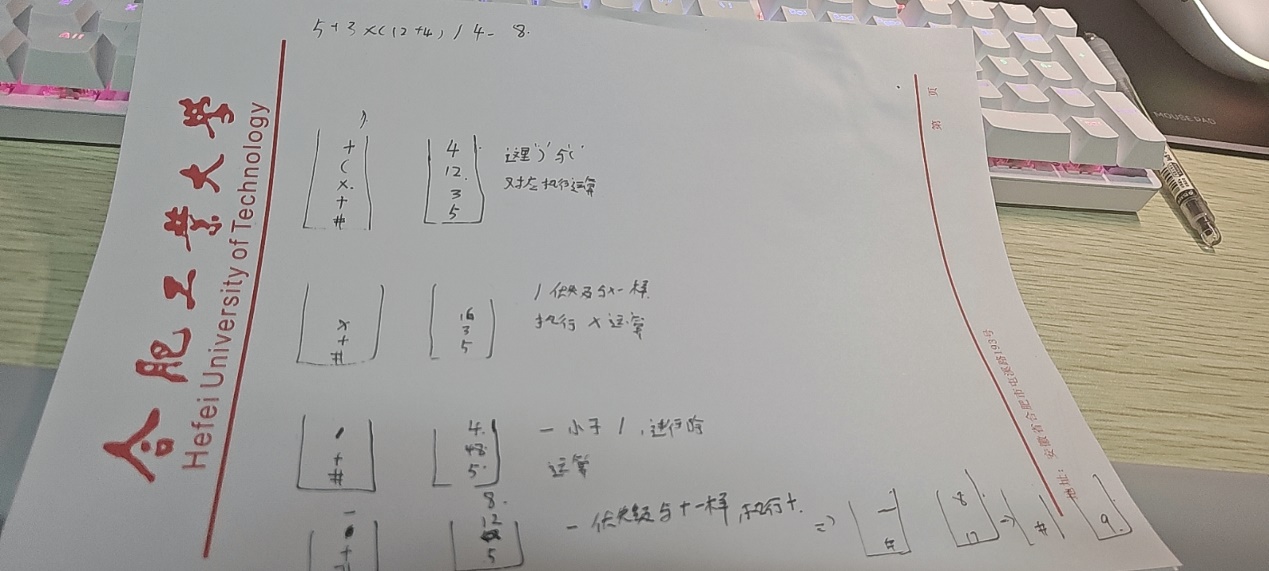
（2）太多了

（3）判断元素是否再出现过，出现了把这两个栈重复的删掉，再来比，如果同时栈中元素为零，那就可以。

（4）元素的排列组合。有Ann种。

2．入栈操作基本不变cout不用++了，先开辟一个新空间，再让top指向它。出栈操作也没怎么变，cout不用—了，先用临时指针保存当前top ，再更新top使之指向上一个数组的地址。,然后再删除临时指针。

取栈顶元素，直接\*top来用。判断是否为空，看top是不是等于栈底的地址。是否未满就看是不是数组最后元素的地址。

4. 

5

#include<stdio.h>

#include<stdlib.h>

#include<stack>

#include<iostream>

#include<string.h>

using namespace std;

stack<int> q1;//数字栈

stack<char> q2;//符号栈

int flag = 0;//用来判断第一位是‘-’的情况

int flag2 = 0;//用来判断第一位是‘-’的情况

int flag1 = 0;//用来判断除数是0的情况

//取出数字栈顶两个数字进行s运算函数，需要注意！因为我们需要对数字栈进行入

//栈出栈操作，改变了数字栈的结构，所以我们需要在栈名前加“&”

void calculate(stack<int>& q1, char s)

{

int x, y, z;

switch (s)

{

case '+':

{

x = q1.top();

q1.pop();

y = q1.top();

q1.pop();

z = x + y;

q1.push(z);

break;

}

case '-':

{

x = q1.top(); q1.pop();

y = q1.top(); q1.pop();

z = y - x;

q1.push(z);

break;

}

case '\*':

{

x = q1.top(); q1.pop();

y = q1.top(); q1.pop();

z = x \* y;

q1.push(z);

break;

}

case '/':

{

x = q1.top(); q1.pop();

y = q1.top(); q1.pop();

if (x == 0)

{

flag1 = 1;

break;

}

z = y / x;

q1.push(z);

break;

}

}

}

//用数字来表示符号优先级

int fuhao(char s)

{

if (s == '+' || s == '-')

return 1;

if (s == '\*' || s == '/')

return 2;

if (s == '(' || s == ')')

return 3;

if (s == '#')

return -1;

}

//对两个栈进行入栈出栈以及计算的函数

void function(char a[100])

{

q2.push('#');//在符号栈先入一个“#”，防止后序找栈顶为空的时候发生异常

int len = strlen(a);

if (a[0] == '-')//如果第一个数字是负号的情况，要单独考虑

{

flag2 = 1;

a[0] = '0';

}

for (int i = 0; i < len; i++)//主循环

{

if (a[i] == '(')

{

q2.push(a[i]);

}

else if (a[i] == ')')

{

while (q2.top() != '(')

{

calculate(q1, q2.top());

q2.pop();

}

q2.pop();

}

else if (a[i] >= '0' && a[i] <= '9')

{

int num = 0;

//数字不一定只有一位，如果是多位数字要进行处理

while (a[i] >= '0' && a[i] <= '9')

{

num \*= 10;

num += (a[i] - '0');

i++;

}

i--;

if (flag2 == 1 && flag == 0)//对第一位是负数的单独处理

num \*= -1;

q1.push(num);

flag = 1;

}

else

{

if (a[i] == '+' || a[i] == '-')

{

if ((fuhao(q2.top()) >= fuhao(a[i])) && q2.top() != '(')

{

while (fuhao(q2.top()) >= fuhao(a[i]) && q2.top() != '(')

{

calculate(q1, q2.top());

q2.pop();

}

q2.push(a[i]);

}

else

{

q2.push(a[i]);

}

}

else if (a[i] == '\*' || a[i] == '/')

{

if (fuhao(q2.top()) == fuhao(a[i]))

{

calculate(q1, q2.top());

if (flag1 == 1)

{

printf("error");

return;

}

q2.pop();

q2.push(a[i]);

}

else

{

q2.push(a[i]);

}

}

}

}

//对栈中剩余数字进行处理，当字符栈空了自然运算就结束了

while (q2.top() != '#')

{

calculate(q1, q2.top());

if (flag1 == 1)

{

printf("error");

return;

}

q2.pop();

}

printf("%d", q1.top());

}

int main()//主函数

{

char a[100];

scanf\_s("%s", a, 100);

function(a);

return 0;

}

