西安交通大学实验报告

课程名称：算法设计与问题求解 实验名称：判断指数、多项式除法等

学 院：机械工程学院 实验日期 2020 年 9 月 19 日

班 级：机械97班 姓 名：杨逢诜 学号：2193712613

**请写出题目、源程序和运行结果，其中运行结果要截图，程序要文字版。请注意排版要整齐，测试要充分。**

**文件命名格式："实验1-"+学号+“.doc”,如：实验1-2170010011.doc**

**通过本页下方的文件框提交。注意截止时间。**

#### 一、实验内容和结果

* **题目1**

写出判断整数n是否为质数的函数，在主函数中验证。

【源程序】

#include<stdio.h>

void PrimeJudge(int n) //空类型函数声明：质数判断

{

int i; //声明辅助变量

for(i=2;i\*i<=n;i++) //从质数2开始逐个自然数进行试验，直到指定数的开方为止

{

if(n%i==0) //只要以上诸数中其中有一个数能够将指定数整除，则该数不是质数，判断并输出为“NO”

{

printf("NO");

break; //找到该数的一个因子之后，该数必不是质数，则循环无必要继续进行，可以直接终止

}

}

if(i\*i>=n) //若在整个循环的检验过程中没有找到质因子，则该数是质数，判断并输出为“YES”

{

printf("YES");

}

}

int main() //主函数开始

{

int n; //声明待判断的整数变量

printf("Please input an integer:"); //构造提示，并输入变量

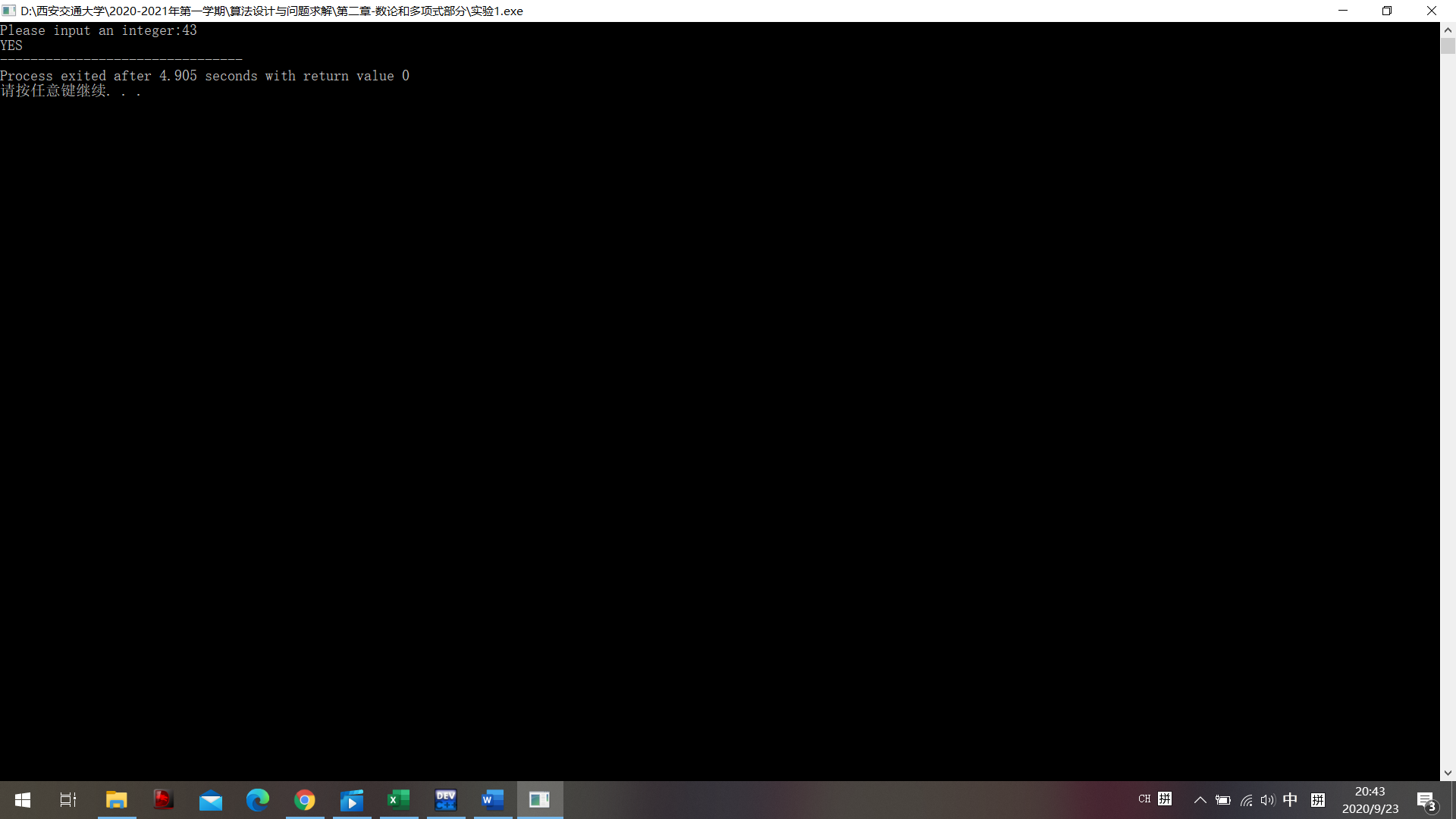
scanf("%d",&n);

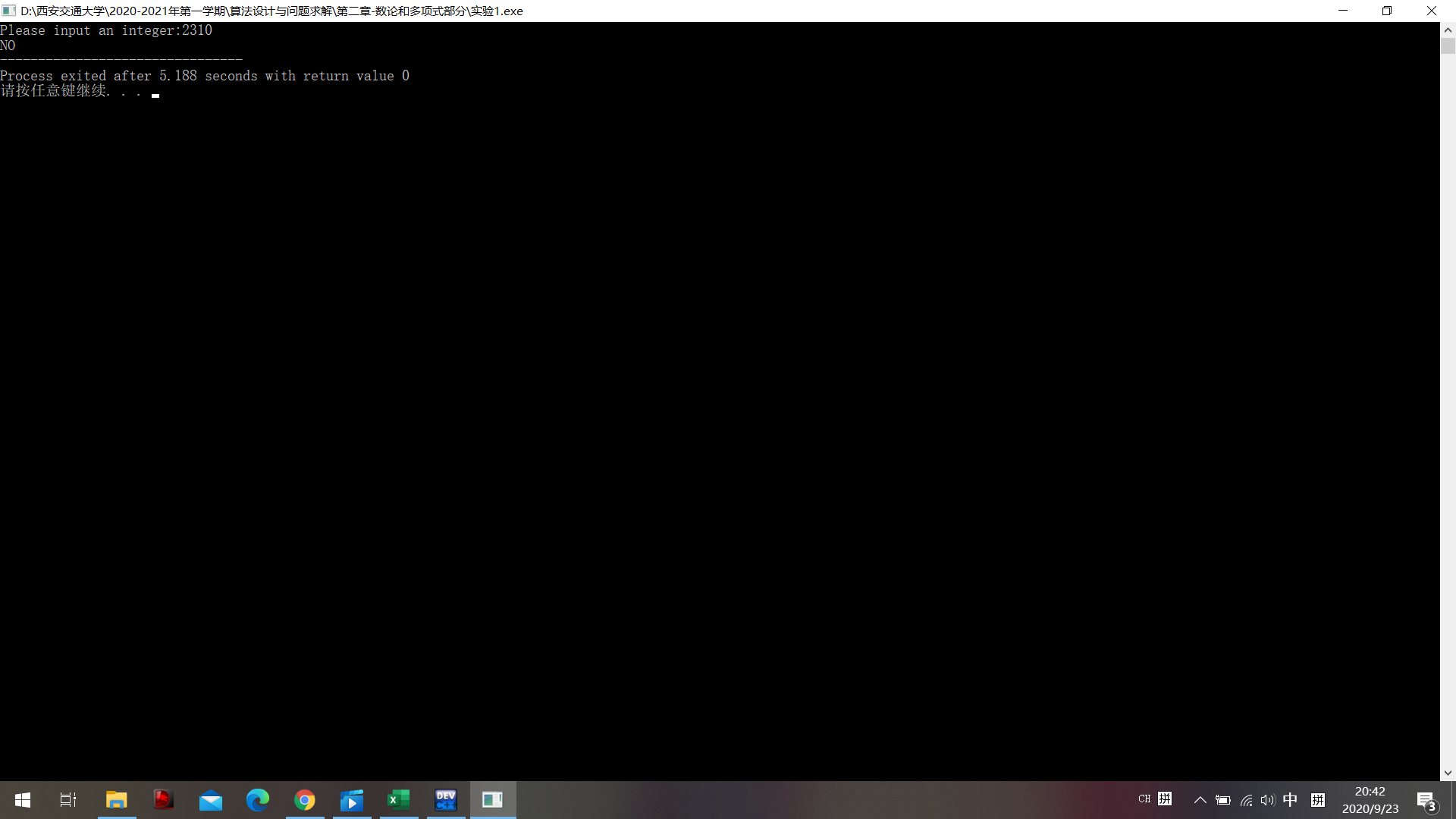
PrimeJudge(n); //调用质数判断函数

return 0; //主函数结束

}

【运行结果】





* **题目2**

编程列举出整数N的所有质因子。

【源程序】

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

void QualityFactor(int n) //声明空类型函数：质因数解算

{

printf("This natural number has the following prime factors:");

int nr=n; //为解算质因数，必须遍历从2到n之间的全部自然数，因此我们须保持形参不变以作为一个判据，而声明另一个初值与形参相同的整型变量，以进行下一步的操作

int i=2; //声明辅助变量

for(;i<=n;i++) //从2起到传入的形参值n为止，逐个检验是否为质因数

{

if(nr%i==0) //若形参能够被循环中的某个数整除，则该数必定是传入的形参的质因数

{

printf("%d ",i); //输出该质因数，并消去形参中的这一质因子

nr=nr/i;

}

while(nr%i==0) //若在消去这一质因子后该数仍能被这一质因子整除，则继续消去这一质因子，进行除法运算直至不能继续整除为止

{

nr=nr/i;

}

}

} //由于每个合数都能够被表示为若干个比该合数小的质因数的乘积，因此即使传入函数的形参能够被某个合数整除，这一合数因子也会被分解为比他小的质因子而被前面已经判定的质因子消去，这一程序本身结构即决定它本身不可能输出合数

int main() //主函数开始

{

int n; //声明变量：待分解的整数

printf("Please input an integer:"); //给出提示，将该整数输入机器

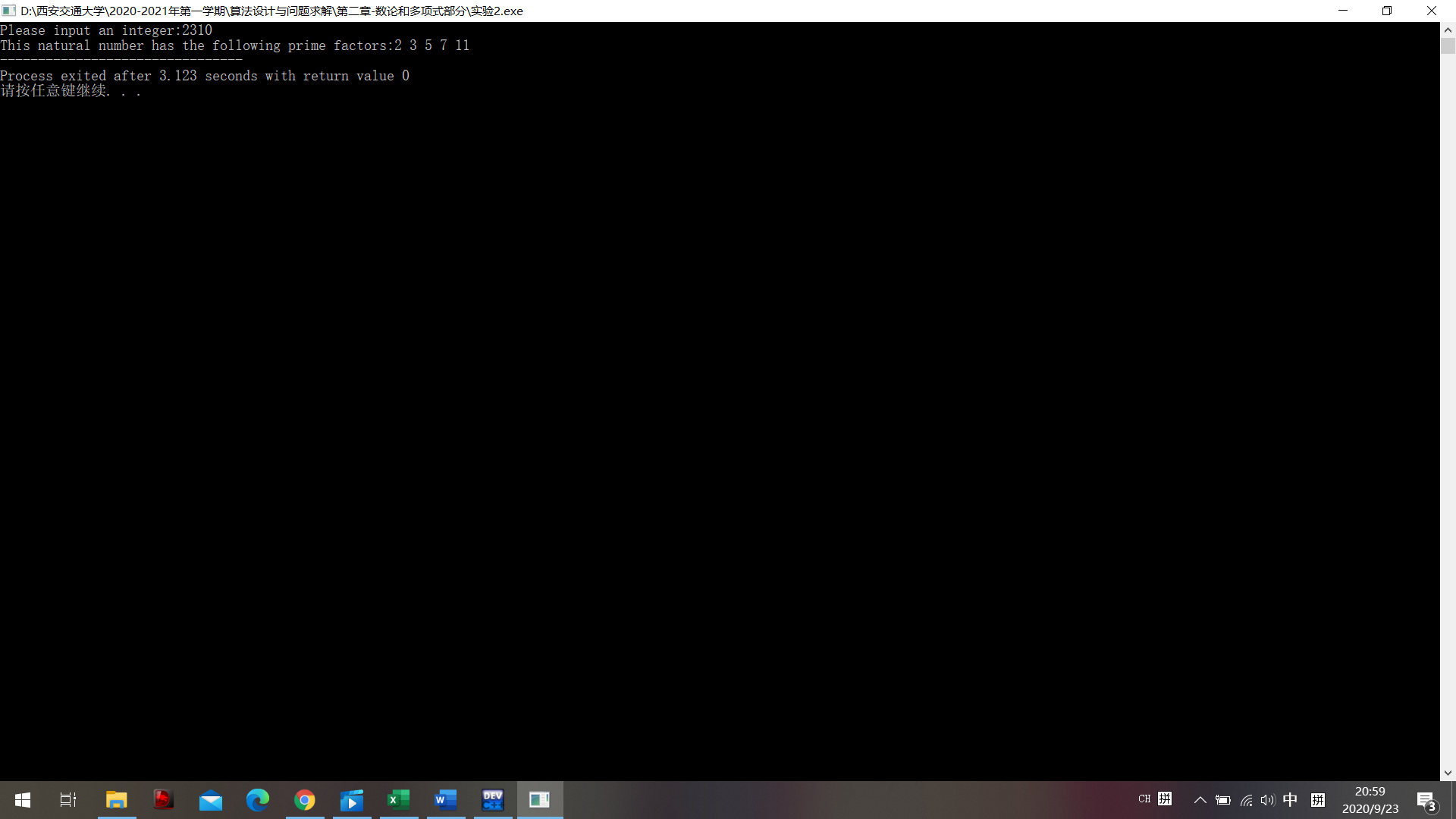
scanf("%d",&n);

QualityFactor(n); //调用质因数解算程序

return 0; //主函数结束

}

【运行结果】



* **题目3**

将书中筛法求质数表的函数抄写下来，对其中每个for循环的作用加上注释。

【源程序】

#include<stdio.h>

#define MAX 500000

int isprime[MAX];

void PrimeTable(int M)

{

int i,j;

for(i=2;i<=M;i++) //判定预备：将所有的介于2和输入之间的数字对应的数组元素全部赋为1

isprime[i]=1;

for(i=2;i<=M;i++) //判定开始：遍历数组中所有有定义的单元

{

if(isprime[i])

{

for(j=i+i;j<=M;j+=i) //若一个数字为质数（即数字对应的数组单元值为1），则令该数字的所有倍数对应的数组单元全部赋0，指代这些数字的合数身份

isprime[j]=0;

}

}

}

int main()

{

int i,m;

printf("请输入m(1<m<500000):");

scanf("%d",&m);

printf("2到%d的整数中质数有：\n",m);

PrimeTable(m);

for(i=2;i<=m;i++) //循环输出：凡是判定为质数（即，数组中对应数不为0） 的数字，全部输出并制表

{

if(isprime[i])

printf("%d\t",i);

}

return 0;

}

（以下部分是我自己写的）

#include<stdio.h>

int main() //主函数开始

{

int n; //声明变量：质数列举的上限

printf("Please input the integer:"); //给出提示，输入变量

scanf("%d",&n);

int nums[n+1]; //声明数组，用以判定质数

for(int i=2;i<=n;i++) //初始化数组：从第二个存储单元之后的所有存储单元，使之存储的数值与下标算符相等（我们不引入0和1，因为它们会引起代数上的麻烦）

{

nums[i]=i;

}

for(int i=2;i<=n;i++) //筛：从2开始到n所有的自然数都拿来跑一次

{

if(!nums[i]==0) //如果一个数还没有被筛掉（对应存储单元里面不是0），那它就是质数，就可以拿来筛（当然，起始的数字2是最小的质数）

{

for(int j=i+1;j<=n;j++) //筛去这个质数的全部倍数，具体方法为将该数对应的存储单元内部赋值为0

{

if((nums[j]%i==0)&&(!nums[j]==0))

{

nums[j]=0;

}

}

}

}

printf("These are the prime numbers prior to it:\n");

for(int i=2;i<=n;i++) //输出所有的质数（也就是对应存储单元不是0的数）并制表，主函数结束

{

if(!nums[i]==0)

{

printf("%d\t",i);

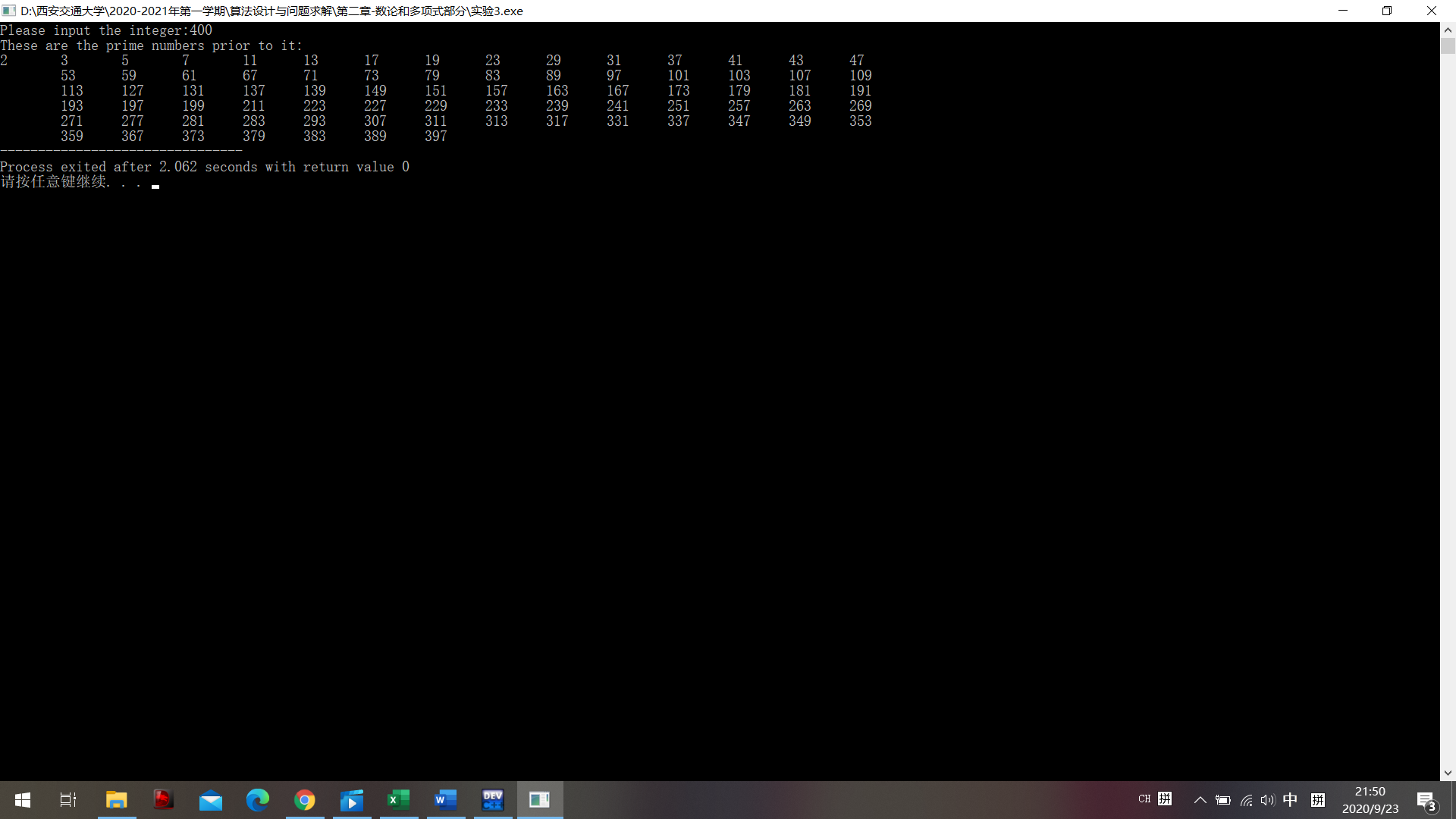
}

}

return 0;

}

【运行结果】



* **题目4**

按书上对一元多项式的存储形式，写出一元多项式除法的程序，并将书上除法的例子数据（ 除以 ）代入，输出结果。

【源程序】

#include<stdio.h>

int main() //主函数开始

{

int m,n; //这一模块的作用是声明并输入多项式的诸参数：首先是两个多项式的次数，其次是第一个多项式的系数，最后是第二个多项式的系数；系数的输入是从高次到低次的

scanf("%d %d",&m,&n);

float mc[m+1],nc[n+1];

for(int i=m;i>=0;i--)

{

scanf("%f",&mc[i]);

}

for(int i=n;i>=0;i--)

{

scanf("%f",&nc[i]);

}

float d[m-n+1]; //声明数组，用以存储多项式除法的运算结果

for(int i=m-n;i>=0;i--) //这一模块的作用是进行除法运算，即循环执行“最高次项系数与最高次项系数相除，分子减去分母与两最高次项系数之比的乘积”直至分子上不具有比分母最高次数更高的项

{

d[i]=mc[n+i]/nc[n];

for(int j=n;j>=0;j--)

{

mc[n+i-j]=mc[n+i-j]-d[i]\*nc[n-j];

}

}

for(int i=0;i<=m-n;i++) //这一模块的作用是输出除法运算结果：第一行输出不带余式的除法结果，第二行输出余式，输出顺序是从高次到低次，中间以空格分隔

{

printf("%02f",d[i]);

if(i<m-n)

{

printf(" ");

}

}

printf("\n");

for(int i=0;i<=n;i++)

{

printf("%02f",mc[i]);

if(i<n)

{

printf(" ");

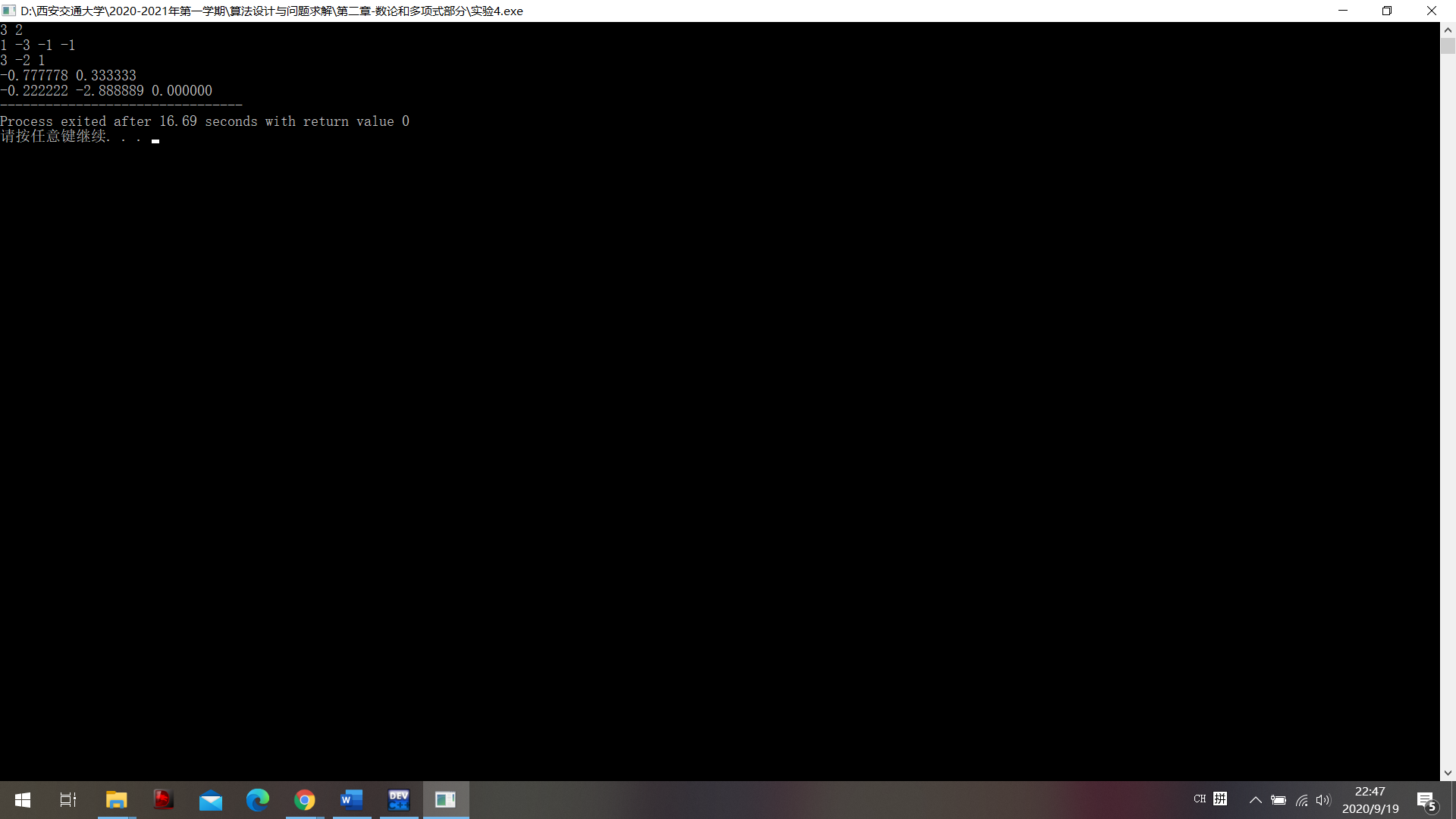
}

}

return 0; //主函数结束

}

【运行结果】



* **题目5**

求N！的尾部有连续多少个零。

【源程序】

#include<stdio.h>

int main() //主函数开始

{

int n; //声明变量：待求尾部0数的阶乘数

printf("Please input the factorial number:"); //给出提示，输入待求0数的阶乘数

scanf("%d",&n);

int zn=0,aux=1; //声明变量：尾部0数，辅助变量

for(int i=1;;i++) //由一次开始，逐次计算5的各次幂在阶乘表达式中出现的次数，不设特定上限

{

aux=aux\*5; //在每次循环开始之处，利用辅助变量构造5的乘幂

zn=zn+n/aux; //阶乘数与对应乘幂相除所得的整数（不计余数），则是具有这一乘幂因子的数在阶乘表达式中出现的次数，我们将这个次数计入到尾部0数中

if(aux>zn) //如果计算出的5的乘幂大于阶乘数，则这一乘幂的最小倍数不会出现在阶乘表达式中，接下来的判定则无意义，循环终止

{

break;

}

}

printf("There are %d zeroes at the tail.",zn); //输出尾部0的总数，主程序结束

return 0;

} //本题的核心在于寻找阶乘结果中10因子的数目，而10又可以拆成5和2两个因子，其中2因子出现的比5因子频繁得多，因此可以认为5因子决定了阶乘结果中的10因子数目；具有5因子的数是5及5乘幂的倍数，且5的n次幂具有n个5因子；这一算法是基于这一原理设计的

//由于5的高次乘幂在之前计算5和5的低次乘幂的倍数数量的时候已经被重复计算，因此在计算高次乘幂对应的尾部0数增量时，并没有乘以n，而是直接相加

【运行结果】

