UEE1302: Final Examination

-- Programming Part --

FULL SCORES:

100%

EXAMINATION TIME:

120 minutes

INSTRUCTIONS:

Problems are all with score 25 points.

You are allowed open any notes or books but prohibited to browse on the Internet to search answers directly. Read carefully the statements and requirements of each problem. Once you complete your program for one problem, please raise your hand and TA will come to your desk for testing. Please note that no credit will be given if your program fails to fully meet the requirement in each problem. Good luck!

Student ID(學號):_____ Name(姓名):_____

Problem #	TA's Signature	Problem #	TA's Signature

Score:_____

PROBLEM 01(25%) DIFFERENTIATION OF POLYNOMIALS

Given an arbitrary polynomial with a degree n

$$f(x) = c_n x^n + c_{n-1}x^{n-1} + \ldots + c_1 x^1 + c_0 \text{ , } n \text{ and } c_i \in Z,\ i \in [0,n]$$

Please write a program to read a file containing a polynomial and find the differentiation of the polynomials. Write the result to the output file. The command-line usage of the program is shown below

>./pg01 input_file.in ouput_file.out

The required format for a sample input file "01.in" is shown as follows.

2X^2+X+1

Note that the representation of polynomial may not be in descending order. It can be X+1+2X^2 in above example.

Your result should be written to user-specified output file. The sample output for "01.in" is file "01.out" with the content shown as below.

4X+1

PROBLEM 02 (25%) ENCRYPTION

Given an encrypted string, how to restore the original password?
If we know that the encryption method is as follows:

(1)Repeat the password twice

(2)Then add a number of any characters before and after it.

Note that exact password is the longest one are out of all possible solutions.

The command-line usage of the program is shown below

>./pg02 input_file.txt ouput_file.txt

The required format for a sample input file "ti02a.in" is shown as follows.

abcdefghijklmnopqruvwxyzthispasswordwordthispasswordword
abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz
tmtthetmttmtntmtotmtntmtotmtn

The input file may contain more than one test data, and each test data takes one line. Each line represents the message after encryption. Characters in the string can only be the letter 'A' ~ 'Z', 'a' ~ 'z', as well as the number of '0 '~ '9'.

Your result should be written to user-specified output file. The sample output for "ti02a.in" is file "ti02a.out" with the content shown as below. There are two items in each line, the first is the exact password, and the second is the length of it.

thispasswordword 16
abcdefghijklmnopqrstuvwxyz 26
tmtntmto 8

PROBLEM 03 (25%) NICE'N EASY FACTORIALS

Write a C++ program to read an integer n ($1 < n < 1000$) and compute $n!$. The result should be written to an output file named n.txt, where each line contains 60 digits.

The command-line usage of the program is shown below

```
>./pg03 n
```

The sample output for "n = 100" is file "100.out" with the content shown as below.

```
933262154439441526816992388562667004907159682643816214685929
638952175999932299156089414639761565182862536979208272237582
511852109168640000000000000000000000000
```

Hint: Use the following structure to manipulate extremely large numbers:

```
typedef struct{
    int length;
    char *data;
} NUMBER;
```

PROBLEM 04(25%) FINDING ROOTS FOR POLYNOMIALS

Given an arbitrary polynomial with a degree n,

$$c_n x^n + c_{n-1}x^{n-1} + \ldots + c_1 x^1 + c_0 = 0$$

each root x can be classified into two cases:

- an integer $p \in Z$ *iff.* $f(p) = 0$ or
- a range $x \in [p, q]$ where $p \le x < q$, $p,q \in Z$ *iff.* $f(p) \times f(q) < 0$

Please write a program to read a file containing multiple polynomials and search from $-r$ to $+r$ to enumerate all roots with the resolution up to the d decimal places for each given polynomial. The command-line usage of the program is shown below

```
>./pg04 input_file.txt ouput_file.txt
```

The required format for a sample input file "ti04a.in" is shown as follows.

```
3 12                    // polynomial 1: degree d1 and range r1
1 5.5 -13 -17.5         // polynomial 1: x^3+5.5 x^2-13x-17.5=0
2 5                     // polynomial 2: degree d2 and range r2
1 0 2                   // polynomial 2: x^2+2=0
3 7                     // polynomial 3: degree d3 and range r3
2 -41.5 230 -150        // polynomial 3: 2x^3-41.5 x^2+230x-150=0
```

Each question takes two lines: the first line specifies the degree d for the polynomial followed by the checking range $\pm r$ and the second line describes the coefficients c_i for each term in the polynomial**.** Note that the comments will not actually appear in the files.

Your result should be written to user-specified output file with the proper description. The sample output for "ti04a.in" is file "ti04a.out" with the content shown as below.

```
1: -7 -1 [2,3) are roots for X^3+5.5X^2-13X-17.5=0 in [-12,+12].
2: No root exists for X^2+2=0 in [-5,+5].
3: [0,1) is the only root for 2X^3-41.5X^2+230X-150=0 in [-7,+7].
```

Hint:

1. You may use **pow(x,y)** in **<cmath>** to ease the computation of x^k.
2. Please refer to Lab 11 exercises for displaying the polynomial.

PROBLEM 05(25%) THROW THE TWO BALLS

One man throw up the first ball at time ***0 sec*** and then throw up the second ball at time ***t sec,*** vertically. Given that gravity ***g*** is ***9.8 N/m***. There are two cases might happen. Case one is the second ball collide with the first, while case two without collision. The command-line usage of the program is shown as follows:

```
>./throw throw.in throw.out
```

The required format for a sample input file and output file is shown as follows. The result should be written to user-specified output file.

Note: It's required to check the time step by step using a time interval ***0.01s***. (check t=0.01, t=0.02 …..)

Case 1: The example is two balls with collision.

```
10 12   1// the initial velocity for ball 1 and 2 is 10 and 12,(m/s)
          //The second ball are throwing when t = 1.
```

Then, the output file should report information as follows:

```
Collide at time 1.45
Ball 1 is falling. Ball 2 is rising.
```

Case 2: The example is two balls without collision.

```
200 100 2// the initial velocity for ball 1 and 2 is 200 and 100,(m/s)
           //The second ball are throwing when t = 2.
```

Then, the output file should report message as follows:

```
There is no collision.
```

Hint: $h = v_0 \times t + 0.5 \times a \times t^2$, where v_0 is initial velocity, t is time, a is acceleration

PROBLEM 06 (25%) GAUSSIAN ELIMINATION

Given a square matrix ***A*** with dimension ***n*** and a vector ***y*** with ***n*** elements, the following equation can be solved by Gaussian elimination.

$$\boldsymbol{Ax} = \boldsymbol{y}$$

For example, given $\boldsymbol{A} = \begin{bmatrix} 2.0 & 1.0 \\ 1.0 & 4.0 \end{bmatrix}$ and $\boldsymbol{y} = \begin{bmatrix} 7.0 \\ 14.0 \end{bmatrix}$, you can model the formula as $\begin{pmatrix} 2.0 & 1.0 & 7.0 \\ 1.0 & 4.0 & 14.0 \end{pmatrix}$ and use Gaussian elimination to reduce the formula as $\begin{pmatrix} 1.0 & 0.0 & 2.0 \\ 0.0 & 1.0 & 3.0 \end{pmatrix}$. Then you can obtain the solution $\boldsymbol{x} = \begin{bmatrix} 2.0 \\ 3.0 \end{bmatrix}$ of $\boldsymbol{Ax} = \boldsymbol{y}$.

Please write a program to read a file containing dimension, context of matrix ***A*** and vector ***y***. The command-line usage of the program is shown below

```
>./pg06 input_file.txt ouput_file.txt
```

The required format for a sample input file "ti06a.in" is shown as follows.

```
2              // dimension of matrix A and vector y
2.0 1.0   // context of matrix A, A[0]
1.0 4.0   // context of matrix A, A[1]
7.0          // context of vector y, y[0]
14.0         // context of vector y, y[1]
```

Your result should be written to user-specified output file. The sample output for "ti06a.in" is file "ti06a.out" with the content shown as below.

```
2.0
3.0
```

Hint: The detail steps of Gaussian elimination of above example are as follows.

R_1 is the first row and R_1 is the first row in the formula $\begin{pmatrix} 2.0 & 1.0 & 7.0 \\ 1.0 & 4.0 & 14.0 \end{pmatrix}$.

1. $R_1 = R_1 \times (\frac{-1}{2})$. The formula becomes $\begin{pmatrix} 1.0 & 0.5 & 3.5 \\ 1.0 & 4.0 & 14.0 \end{pmatrix}$.
2. $R_2 = R_1 \times (-1) - R_2$. The formula becomes $\begin{pmatrix} 1.0 & 0.5 & 3.5 \\ 0.0 & 3.5 & 10.5 \end{pmatrix}$

3. $R_2 = R_2 \times (\frac{1}{3.5})$. The formula becomes $\begin{pmatrix} 1.0 & 0.5 & 3.5 \\ 0.0 & 1.0 & 3.0 \end{pmatrix}$

4. $R_1 = R_2 \times (\frac{-1}{2}) - R1$. The formula becomes $\begin{pmatrix} 1.0 & 0.0 & 2.0 \\ 0.0 & 1.0 & 3.0 \end{pmatrix}$

PROBLEM 07 (25%) PLAYFAIR CIPHER

The Playfair cipher is a manual symmetric encryption technique and was the first literal digraph substitution cipher invented in 1854 by Charles Wheatstone.

You need to uses a **5 by 5 table** containing a **key word** or **phrase**. Memorization of the keyword and 4 simple rules was all that was required to create the 5 by 5 table and use the cipher.

1. Fill in the spaces in the table with the letters of the keyword (dropping any duplicate letters)
2. The key is written from the top rows of the table, from left to right.
3. Fill the remaining spaces with the rest of the letters of the alphabet in order omitting "Q" to reduce the alphabet.
4. The keyword together with the conventions for filling in the 5 by 5 table constitutes the cipher key.

For example, if the key word is *THIS IS A KEY*, the table will be

THISA

KEYBC

DFGJL

MNOPR

UVWXZ

To encrypt a message, one would break the message into digraphs (groups of 2 letters) such that, for example, "**HELLOWORD**" becomes "**HE LL OW OR LD**", and map them out on the key table. The two letters of the digraph are considered as the opposite corners of a rectangle in the key table. Note the relative position of the corners of this rectangle. Then apply the following 4 rules, in order, to each pair of letters in the plaintext:

1. If both letters are the same (or only one letter is left), add an "X" after the first letter. Encrypt the new pair and continue. For example of "**HE LL OW OR LD**", based on the first rule, it will be "**HE LX OW OR LD**".
2. If the letters appear on the same row of your table, replace them with the letters to their immediate right respectively (wrapping around to the left side of the row if a letter in the original pair was on the right side of the row). For example of "**HE LX OW OR LD**", "**OR**" and "**LD**" are in the same row, they will be replaced by "**PM**" and "**DF**", respectively.
3. If the letters appear on the same column of your table, replace them with the letters immediately below respectively (wrapping around to the top side of

the column if a letter in the original pair was on the bottom side of the column). For example of "**HE LX OW OR LD**", "**HE**" and "**OW**" are in the same column, they will be replaced by "**EF**" and "**WI**", respectively.

4. If the letters are not on the same row or column, replace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important – the first letter of the encrypted pair is the one that lies on the same **row** as the first letter of the plaintext pair. For example of "**HE LX OW OR LD**", "**LX**" is not in the same row or column, it will be replaced by "**JZ**".

Therefore, "**HE LL OW OR LD**" will be encrypted into "**EF JZ WI PM DF**".

Please write an encryption program to read a file that need to encrypted and report the encrypted results to the specified output file. The command-line usage of the program is shown below

```
>./pg07 input_file.txt –o ouput_file.txt
```

The required format for a sample input file "ti07a.in" is shown as follows.

```
THIS PROGRAM IS A TEST FILE TO BE ENCRYTED BY PLAYFAIR CIPHER
```

Then the program will ask for key word

```
>Please input the key word:
```

Your result should be written to user-specified output file. The sample output for "ti07a.in" is file "ti07a.out" with the content shown as below.

```
HISARMWOZCOTATHKAHGHFCIMCYFVLZKIKFCBRJICLHAOYANSCN
```

PROBLEM 08(25%) COMPLEX ARITHMETIC

Please write a C++/C program to operate the arithmetic of complex number. You have to read two complex numbers from "***complex.txt.***" and output the arithmetical result of the two complex numbers to "***result.txt***".
The example files content are as follow.

The command-line usage of the program is shown below

```
>./pg08 complex.txt result.txt
```

The required format for a sample input file "ti08a.in" as "***complex.txt***" is shown as follows.

```
1.5+6i
-2-10i
```

In the ***"complex.txt"*** file, the first line and the second line indicates the complex number A and B, respectively. The representation of complex number is a+bi, where a means the real part and b means the image part. If a is equal to zero, it can be written as 0+bi instead of bi. For the same reason, it should be written as a+0i while the complex number has no image part.

Your result should be written to user-specified output file. The sample output for "ti08a.in" is file "ti08.out" with the content shown as below.

```
-0.500–4.000i      // A + B
3.500+16.000i      // A - B
57.000–27.000i     // A * B
-0.606+0.029i      // A / B
```