CSC3002 Complete Assignment 2

Important Notes:

1. The assignment is an individual project, to be finished on one's own effort.
2. **Submission deadline is 23:59pm Friday, 22 March, 2024 (Week #8). After the deadline, we will reopen OJ for late submission for more 3 days.** Late submission will lead to some penalty – 10% deduction for each day of the delay and no submission would be accepted after 3 days of the due date.

3. Please also submit your final **four program**s and **your one page document**. The four programs should be renamed as "StudentID_A2_Encrypt.cpp", "StudentID_A2_CSV.cpp", "StudentID_A2_Credit_card.cpp", and "StudentID_A2_Hanoi.cpp", respectively on the blackboard. For example, a student whose Student ID is "120000001" should submit **four programs** named as "120000001_A2_Encrypt.cpp", "120000001_A2_CSV.cpp", "120000001_A2_Credit_card.cpp", and "120000001_A2_Hanoi.cpp". Furthermore, prepare **one page document to highlight your ideas of your solutions to the given four problems**. The one page document should be named as "StudentID_A2_solutions.pdf".

4. OJ website: http://10.26.200.13/. After opening this website, go the contest "**CSC3002 24Spring Assingment 2"**. Please u**se your student ID** as the user name for registration. We score your assignment only using your student ID. If you are off campus, please use VPN to access the OJ.

5. Each student is only permitted to submit code to OJ **up to 100 times for each problem**. Only the **last submission** will be used in evaluation of assignment marks.

6. Plagiarism is strictly forbidden, regardless of the role in the process. Notably, ten consecutive lines of identical codes are treated as plagiarism. Depending on the seriousness of the plagiarism, 30%-100% of marks will be deducted.

Marking Criterion:

1. The full score of the assignment is 100 marks.
2. **We have 4 problems in this assignment with 5 test cases each. Each test case has 5 marks.**
3. Zero mark is given if: there is no submission; a normal submission fails all test cases.

Running Environment:

1. The submissions will be evaluated in the course's OJ system running **C++17** and Linux platform.
2. All students will have an opportunity to test their programs in the OJ platform prior to the official submission.
3. In the test, each program is required to finish within 10 **seconds**, with no more than **64MB memory**. **This is a strict requirement measured in the server environment!**

Submission Guidelines:

1. Inconsistency with or violation from the guideline leads to marks deduction.
2. It is the students' responsibility to read this assignment document and submission guidelines carefully and in detail. No argument will be accepted on issues that have been specified clearly in these documents.

Problem 1 Description (Encrypt and decrypt text file using C++):

- Background

  Encryption is a process by which a plain text or a piece of information is converted into a text (also known as Ciphertext) which can only be decoded by the receiver for whom the information was intended.

  Decryption is the process of converting a meaningless message (Ciphertext) into its original form (Plaintext). It works by applying the conversion algorithm opposite of the one that is used to encrypt the data.

- Method
  - A class *encdec* is defined with two member functions: *encrypt()* and *decrypt()*. The name of the file to be encrypted is the member variable of the class.
  - *encrypt()* function is used to handle the encryption of the input file. The file handling code is included in the *encrypt()* function to read the file and write to the file. A new encrypted file called *encrypt.txt* is generated with all the encrypted data in it. The encrypted file is encrypted using a key that is being inputted by the user.
  - *decrypt()* function is used to read the encrypted file and decrypt the data and generate a new file *decrypt.txt*. To decrypt a file, a key is requested from the user. If the correct key is entered, then the file is successfully decrypted.
  - Caesar cypher algorithm is adopted in this problem. The concept is to replace each alphabet by another alphabet which is 'shifted' by some fixed number (referred to as '*key*") between -200 and 200. You need to do mod(key, 26), then the shifted number should locate between 0-25.

  For example, the key is 3. The mapping alphabet should follow:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

  Move characters 3 letters to the left. If the key is negative, the characters will move 3 letters space to right.

We would have **5** test cases for evaluation.

For each test case, you should output its result.

Input: Here is an illustration of the input math expression at a line.

```
3
encrypt
tutorial
/*Here is a blank line.*/
```

1. The first line indicates the shifted key number. The second line indicates the function we use,

and it would be either '*encrypt*' or '*decrypt*'. The remaining part is the text we need to encrypt or decrypt and it can have multiple lines, including a-z, A-Z and spalce. No special characters. There is a blank new line at the end.

Output: For the output, please output the text you encrypted or decrypted.

```
wxwruldo
/*Here is a blank line.*/
```

Example 1: the input a math expression is as follows:

```
194
decrypt
Lo dCAyrxzVGwbXwGhbktvBcoxisKlcRkAraXxRXDvzbYrAoniCxO bozPft
```

The corresponding output is as follows:

```
Zc rQOmflnJUkpLkUvpyhjPqclwgYzqFyOfoLlFLRjnpMfOcbwQlC pcnDth
```

Example 2: the input a math expression is as follows:

```
-166
encrypt
j
F
C
E
m
T
u
```

The corresponding output is as follows:

```
z
V
S
U
c
J
k
```

Example 3: the input a math expression is as follows:

```
-49
decrypt
```

```
TxwJhAuzeNaiBxzIHl YbxyMyWPGJCm
NZdwRPPFyleajLWoJUJA PTtNXjkmce
zoNTZTc ueNIuSCyFiaYrGihSwdWUJG
lONWbVVDBgasevLXsSKREPbXaTnnnJZ
Ype orjzHbHftThbseBDwvQSBRWAoJz
DaBneqwpKVVhmUHmzJvtxlUStBnGVuH
yTQuNflx qIQwtRtzYTCjFnLCnTcZxl
EMpfybjeevO PkngfTnodEKHauDATVz
RLzKfWoGMnzyZHbvCXGyBcUsPnxPZfP
```

The corresponding output is as follows:

```
QutGeXrwbKxfYuwFEi VyuvJvTMDGZj
KWatOMMCvibxgITlGRGX MQqKUghjzb
wlKQWQz rbKFrPZvCfxVoDfePtaTRGD
iLKTySSAYdxpbsIUpPHOBMyUxQkkkGW
Vmb logwEyEcqQeypbYAtsNPYOTXlGw
AxYkbntmHSSejREjwGsquiRPqYkDSrE
vQNrKciu nFNtqOqwVQZgCkIZkQzWui
BJmcvygbbsL MhkdcQklaBHExrAXQSw
OIwHcTlDJkwvWEysZUDvYzRpMkuMWcM
```

Problem 2 Description (CSV data file management using C++)

- Background
  - *CSV* is a simple file format used to store tabular data such as a spreadsheet or a database.
  - *CSV* stands for *Comma Separated Values*. The data fields in a CSV file are separated/delimited by a comma **(', ')** and the individual rows are separated by a newline *('\n')*.
- Method
  - The manipulation of records in a CSV file includes the processes of creation, updating, and deletion.
  - The sequence starts with the "CREATE" operation, which defines a function with a series of codes to handle file by opening an existing csv file or creating a new file , managing the records of data and saving the file.
  - To access a specific record in a data file and retrieve the desired column data by accessing the row.
  - To update a record by reading data from a file and comparing it with the user input. Prompt the user to enter new values for the record that needs to be updated. Update the specific column field, referred to as the index, with the new data. Write the updated record and all other records to a new file to save the updated data file.
  - To remove a record by performing the following steps: 1) Retrieve data from a file and compare it with the user input, as described in the read and update operation; 2). Save all the updated records, excluding the data to be deleted, in a new file; and

3) Delete the old file and rename the new file with the same name as the old file.

- Requirement

  Develop your own C++ program to implement CSV data file management with the functions such as creation, updating and deletion.

Input: Here is an illustration of the input cases.

```
This,is,the,second,assignment
This,is,the,second,assignment
create
update 1 0 updatedValue
remove 0
/*Here is a blank line.*/
```

The first two lines are the content of the CSV file you need to create and edit. The content can be **multiple** lines. The next line is '*create'*,which means you need to create a csv file with the above content. Then, the next line is '*update 1 0 updatedValue*' means you need to update the csv file. The first number is the row index, the second number is the column index. In the case, we need to update the word at (1, 0) to *updatedValue.* Then, the next line is '*remove 0*', which means to remove the first raw. The number indicates the row index we need to remove.

Only a-z and A-Z will be used in test case, no space or special letters.

Output:

```
updatedValue,is,the,second,assignment
/*Here is a blank line.*/
```

Example 1: the input is as follows:

```
TCdwMlGHfz,iPCgItlPWf,cxvFXUQfgY,OsPauSAnwr
ZQxKPlcmZT,GDIBcKXjSe,ivoWeLmGbG,PbeJGmkIkQ
create
update 1 1 updatedValue
remove 1
```

The corresponding output is as follows:

```
TCdwMlGHfz,iPCgItlPWf,cxvFXUQfgY,OsPauSAnwr
```

Example 2: the input is as follows:

```
fSCBHgNppk,rncwyAKUKa,RhgYKCsqMR,lqVwpETDVZ,KATIStyFdY
oHUfCzKKXD,SCJuOOYWfk,nsUVyyZJgx,fPMYWkVBtU,ZGoVmewvyE
create
update 1 0 updatedValue
remove 0
```

The corresponding output is as follows:

```
updatedValue,SCJuOOYWfk,nsUVyyZJgx,fPMYWkVBtU,ZGoVmewvyE
```

Example 3: the input is as follows:

```
ONOEAbzYwS,nzHOqnGHYG
WbAgUagCgT,mSpVknIvfG
hhNDeluncE,ABymayTzEN
yfVwMDcJsF,yfDwAhWaSC
EbnoDpwhGC,hdQRfjHsco
qlXbRCLqHe,dixJFMoadk
LotHmYLFSU,wISzqnVTmy
create
update 3 1 updatedValue
remove 3
```

The corresponding output is as follows:

```
ONOEAbzYwS,nzHOqnGHYG
WbAgUagCgT,mSpVknIvfG
hhNDeluncE,ABymayTzEN
EbnoDpwhGC,hdQRfjHsco
qlXbRCLqHe,dixJFMoadk
LotHmYLFSU,wISzqnVTmy
```

Problem 3 Description (Credit card validation by Luhn algorithm using C++)

- Background

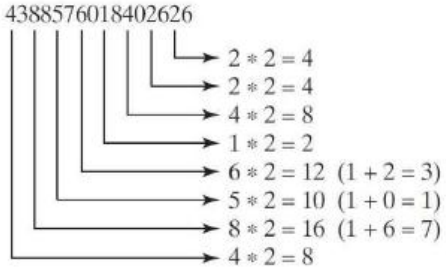
  Credit card numbers follow certain patterns. It must have between 13 and 16 digits, and the number must start with:

  - 4 for Visa cards
  - 5 for MasterCard credit cards
  - 37 for American Express cards
  - 6 for Discover cards

- Method

  In 1954, Hans Luhn of IBM proposed an algorithm for validating credit card numbers. The algorithm is useful to determine whether a card number is entered correctly or whether a credit card is scanned correctly by a scanner. Credit card numbers are generated following this validity check, commonly known as the *Luhn check* or the *Mod 10 check*, which can be described as follows (for illustration, consider the card number 4388576018402626):

  - Step 1: Double every second digit from right to left. If doubling of a digit results in a two digit number, add up the two digits to get a single-digit number.



- Step 2: add all single-digit numbers from Step 1

  $$4+ 4 + 8 + 2 + 3 + 1 + 7 + 8 = 37$$

- Step 3: Add all digits in the odd places from right to left in the card number

  $$6 + 6 + 0 + 8 + 0 + 7 + 8 + 3 = 38$$

- Step 4: Sum the results from Steps 2 and 3

  $$37 + 38 = 75$$

- Step 5 If the result from Step 4 is divisible by 10, the card number is valid; otherwise, it is invalid. For example, the number 4388576018402626 is invalid, but the number 4388576018410707 is valid.

- Requirement

  Develop your own C++ program that prompts the user to enter a credit card number as an integer. Display whether the number is valid or invalid.

Input: Here is an illustration of the input cases.

4388576018410707

The input will be only one line with the number you need to check.

Output:

Valid

The output will be either '*Valid*' or '*Invalid*'.

Example 1: the input is as follows:

5354416281323658

The corresponding output is as follows:

Valid

Example 2: the input is as follows:

4441338692943977

The corresponding output is as follows:

Invalid

Example 3: the input is as follows:

6342904238914169

The corresponding output is as follows:

Valid

Problem 4 Description (C++ Program for the Tower of Hanoi puzzle)

- Background

  The *Tower of Hanoi* is a mathematical game or puzzle. It consists of three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape. The following figure shows the initial state of the Tower of Hanoi with 5 disks.

- Method

  The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:
  - Step 1: Only one disk can be moved at a time.
  - Step 2: Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
  - Step 3: No disk may be placed on top of a smaller disk.

- Requirement

  Assume that initially all the disks are placed on rod A. Develop your own C++ program to print out the steps to move all the disks from rod A to rod C via rod B

Input:

```
2 1 0
E
E
```

We use numbers to simulate the disk. For example, we are solving the problem of the Tower of Hanoi with 3 disks. The 3 disks are simulated to 3 numbers, 2 means the biggest disk at the bottom, and 0 means the smallest dist at the top. The next two lines are 'E', which means empty. The total number of lines is 3, which means the three rods.

Output:

```
E
E
2 1 0
```

Example 1: the input is as follows:

```
0
E
```

```
E
```

The corresponding output is as follows:

```
E
E
0
```

Example 2: the input is as follows:

```
4 3 2 1 0
E
E
```

The corresponding output is as follows:

```
E
E
4 3 2 1 0
```

Example 3: the input is as follows:

```
3 2 1 0
E
E
```

The corresponding output is as follows:

```
E
E
3 2 1 0
```