ENGG1340 / COMP2113, Assignment 1

Due Date: Mar 18, 2022 23:59

If you have any questions, please post to the Moodle discussion forum on Assignment 1.

Total marks: 100 marks

- 5 marks for proper code comments and indentation
- 95 marks for program correctness

A maximum of 5 marks will be deducted if you fail to follow the submission instructions strictly.

General Instructions

Read the instructions in this document carefully.

In this assignment you will solve 4 problems and a tester would automatically test your submitted program. So if your submitted files and program outputs do not conform to our instructions given here, your programs cannot be evaluated and you will risk losing marks totally.

Sample test cases are provided with each problem in this document. Note that the test cases may or may not cover all boundary cases for the problem. It is also part of the assessment whether you are able to design proper test cases to verify the correctness of your program. We will also use additional test cases when marking your submission.

Input and output format

Your C++ programs should read from the **standard input**. Also, your answer should be printed through the **standard output**. If you failed to follow this guide, the tester may not able to give a score for your program. Additionally, you should strictly follow the sample output format (including space, line breaker, etc.), otherwise, your answer might be considered as wrong.

How to use the sample test cases

Sample test cases in text file formats are made available for you to check against your work to avoid formatting errors which might fail the tester. Here's how you may use the sample test cases. Take Problem 2 test case 3 as an example. The sample input and the expected output are given in the files `input2_3.txt` and `output2_3.txt`, respectively. Suppose that your program is named "2", do the followings at the command prompt of the terminal to check if there is any difference between your output and the expected output.

```
./2 < input2_3.txt > myoutput.txt
diff myoutput.txt output2_3.txt
```

Testing against the sample test cases is important to avoid making formatting mistakes. The additional test cases for grading your work will be of the same formats as the sample test cases.

Coding environment

You must make sure that your program can compile, execute and generate the required outputs on our standard environment, namely, the gcc C++11 environment we have on the CS Linux servers (academy*). Make sure that the following compilation command is used to compile your programs:

```
g++ -pedantic-errors -std=c++11 [yourprogram].cpp
```

As a programmer/developer, you should always ensure that your code can work perfectly as expected on a target (e.g., your client's) environment, not only on yours.

While you may develop your work on your own environment, you should always try your program (compile & execute & check results) on our standard environment before submission.

Submission

Name your C++ programs as in the following table and put them together into one directory. Make sure that the folder contains only these source files (`*.cpp`) and no other files. **Compress this directory as a `[uid].zip` file where [uid] is your university number** and check carefully that the correct zip file have been submitted. We suggest you to download your submitted file from Moodle, extract them, and check for correctness. **You will risk receiving 0 marks for this assignment if you submit incorrect files.** Resubmission after the deadline is not allowed.

Problem	Code templates provided	Files to Submit
1	-	`1.cpp`
2	`2.cpp`	`2.cpp`
3	`3.cpp`	`3.cpp`
4	`4.cpp`	`4.cpp`

Late submission

If submit within 3 days after the deadline, 50% deduction. After that, no mark.

Evaluation

Your code will be auto-graded for technical correctness. In principle, we use test cases to benchmark your solution, and you may get zero marks for not being able to pass any of the test cases. Normally partial credits will not be given for incomplete solution, as in many cases the logic of the programs are not complete and an objective assessment could be difficult. However, your work may still be considered on a case-by-case basis during the rebuttal stage.

Academic dishonesty

We will be checking your code against other submissions in the class and from the Internet for logical redundancy. Please be reminded that no matter whether it is providing your work to others, assisting others to copy, or copying others will all be considered as committing plagiarism and we will follow the departmental policy to handle such cases. Please refer to the course information notes for details.

Getting help

You are not alone! If you find yourself stuck on something, post your question to the course forum. We want this assignment to be rewarding and instructional, not frustrating and demoralizing. But we don't know when or how to help unless you ask.

Please be careful not to post spoilers. Please don't post any code that is directly related to the assignments to the discussion forums or share your work to any public domain. However you are welcome and encouraged to discuss general ideas on the discussion forums.

Problem 1: Palindromic Numbers

A palindromic number is one that reads the same both ways, from the beginning or from the end. For example, 525 and 8448 are palindromic numbers. Note that 525 and 8448 are also products of two 2-digit numbers: 525 = 15 * 35 and 8448 = 96 * 88. On the other hand, 7777 is a palindromic number but not a product of two 2-digit numbers.

Write a C++ program that prompts the user for two integers `M` and `N` and output numbers in the range of `M` and `N` (inclusively) which are either (1) palindromic only; (2) product of two 3-digit numbers only; or (3) both palindromic and product of two 3-digit numbers.

Input:

- A single line containing two integers `M`, `N`, and a char `opt` for the display option.

- You may assume that `M` <= `N` and both numbers are within the range 10000 to 999999 (inclusively), and `opt` must be one of the chars `'p'`, `'t'`, `'b'`.

Output:

- If the input `opt` is the char `'p'`, then output the palindromic numbers in the range of `M` and `N` (inclusively) **in increasing order**.
- If the output `opt` is the char `'t'`, then output the numbers which are a product of two 3-digit numbers in the range of `M` and `N` (inclusively) **in increasing order**.
- If the input `opt` is the char `'b'`, then output the numbers which are both palindromic and a product of two 3-digit numbers in the range of `M` and `N` (inclusively) **in increasing order**.
- If there is no number within the range that matches the criteria, no line will be output.

Requirement:

- Your program must implement the following two helper functions:
    - `bool isPalindrome(int x)` which returns whether the parameter `x` is a palindromic number.
    - `bool isProduct(int x)` which returns whether the parameter `x` is a product of two 3-digit numbers.
- Call the helper functions in your `main` function to accomplish the task.
- You can ONLY use the simple data types `char`, `bool`, `int`, `double`. In other words, you are not allowed to use other data types or data structures such as arrays, strings or STL containers (e.g., vectors), etc.

Sample Test Cases

User inputs are shown in blue.

1_1

```
10000 10300 p
10001
10101
10201
```

1_2

```
10000 10300 t
10000
10100
10200
10201
10300
```

1_3

```
10000 10300 b
10201
```

1_4

```
99000 110000 b
99099
99199
99299
99599
99699
99799
99899
99999
101101
102201
105501
106601
108801
```

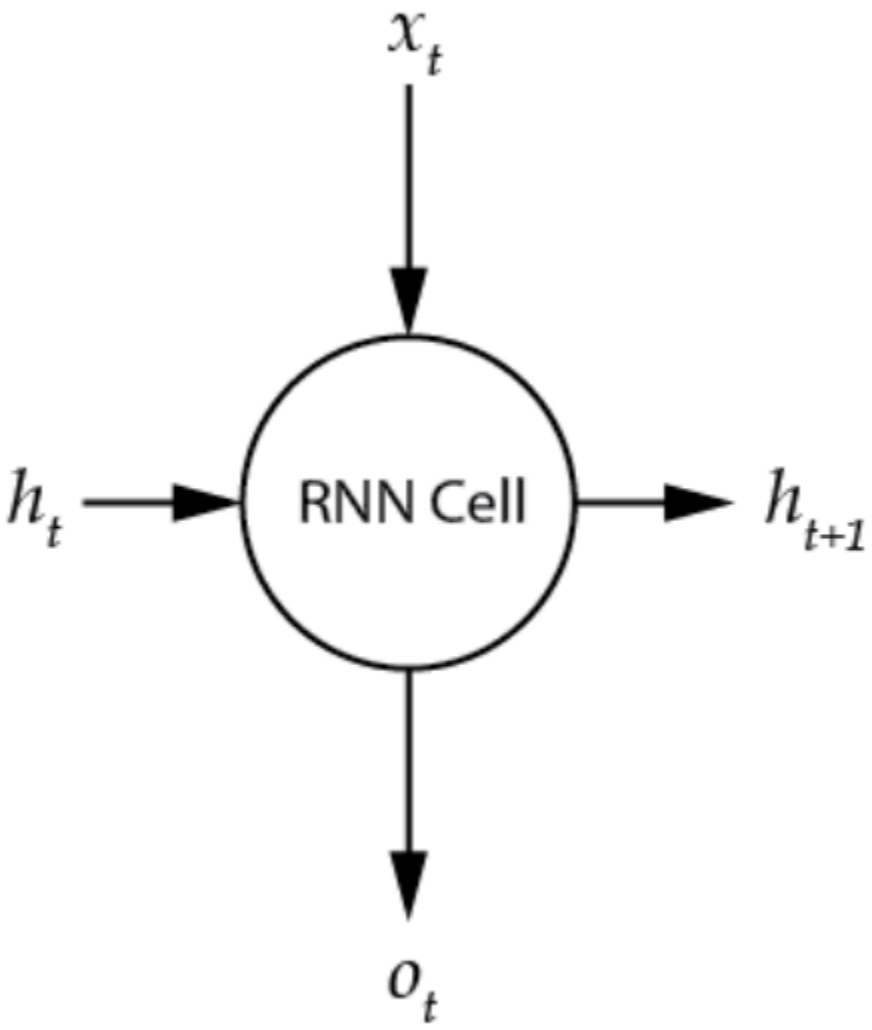
1_5 (*Note: there is no output in this test case*)

```
100000 101000 b
```

Problem 2: Recurrent Neural Networks

Recurrent neural networks (RNNs) are deep-learning architectures which are particularly powerful in dealing with time-series (e.g., financial) and natural language data. In this problem, you are going to write a program to simulate the basic forward propagation mechanism in an RNN using simple 1D arrays. **Note that you do not need to understand what an RNN is to solve this problem; you will find all the necessary information that you need to solve this problem in the following description.**

An RNN cell at time t takes an input x_t and a hidden state h_t from the previous cell, and computes a next state h_{t+1} and an output o_t.



Specifically, each RNN cell computes o_t and h_{t+1} using the following two functions:

$o_t = \text{sigmoid}(0.1\, x_t + 1.5\, h_t)$

$h_{t+1} = \tanh(0.5\, x_t - 2\, h_t)$

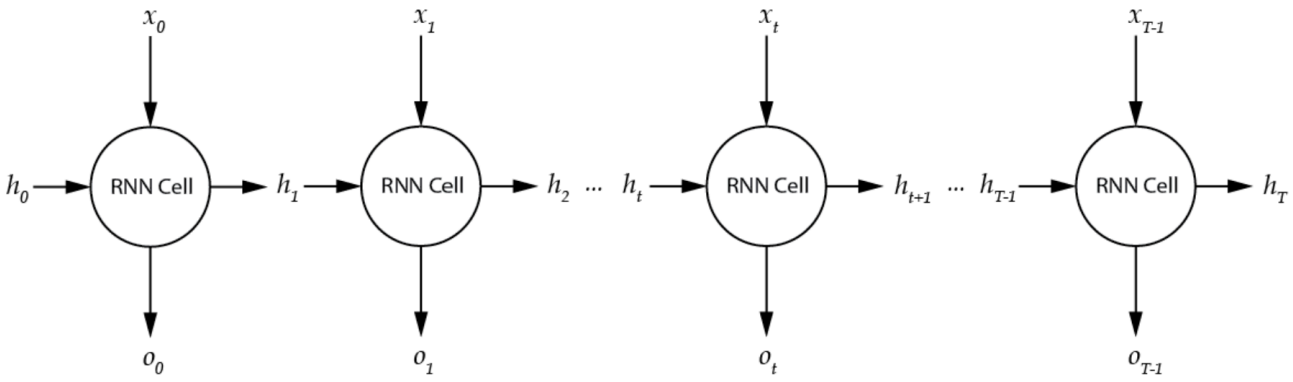
where

$\text{sigmoid}(x) = 1/(1 + e^{-x})$

$\tanh(x) = 2 \times \text{sigmoid}(2x) - 1$

and $e = 2.72$.

Multiple RNN cells can then be connected to form a network. The following figure depicts an RNN of T cells that accepts an input sequence x_0, x_1,...,x_{T-1} and an initial hidden state h_0 as inputs, and outputs a sequence o_0, o_1, ..., o_{T-1}.



Write a C++ program that implement the above simple RNN.

Input:

- the first line contains two numbers: an integer T denoting the recurrent times ($1 \leq T \leq 100$) and a floating-point number h_0 denoting the initial hidden state; and
- the second line contains the input sequences which are T floating-point numbers x_0, x_1,...,x_{T-1}.

Output:

- Your program should display the hidden state sequences h_1, h_2, ..., h_T in the first line and the output sequences o_0, o_1, ..., o_{T-1} in the second line.

- Floating-point numbers are displayed with 10 decimal places. Use `setprecision(n)` defined in the header `<iomanip>` to set the precision parameter of the output stream.

Requirement:

- You will need to complete the following functions in the provided template `2.cpp`. Read the code in the template carefully to see what have been provided for you. You will find details of the function prototypes in the in-code comments too.
    - `sigmoid()` for `sigmoid` activation function
    - `tanh()` for `tanh` activation function
    - `ComputeH()` for computing the next hidden value in RNN cell
    - `ComputeO()` for computing the output value at current time step
    - `PrintSeqs()` for printing the values in a 1D array to screen
    - `main()` for main function
- Use 1D arrays to store the sequences for x_i, h_i and o_i.
- You can ONLY use the simple data types `char`, `bool`, `int`, `double` and arrays. In other words, you are not allowed to use other data types or data structures such as strings or STL containers (e.g., vectors), etc.

Sample Test Cases

User inputs are shown in blue.

2_1:

```
1 1.0
1.0
-0.9053193842
0.8321596401
```

2_2:

```
5 0.0
1.0 2.0 3.0 4.0 5.0
0.4623655914 0.0751742883 0.8741722520 0.2466235712 0.9645899021
0.5249949450 0.7097382221 0.6018123354 0.8471395064 0.7048466172
```

2_3:

```
3 1.0
0.0 0.0 0.0
-0.9641167571 0.9586890814 -0.9578009493
0.8177157977 0.1904499845 0.8082908051
```

Problem 3: Shift Cipher

Write a C++ program which encrypts and decrypts a sequence of input characters using an algorithm adapted from the Vignère cipher as described below.

We first consider the algorithm **to encrypt/decrypt a single character**:
To encrypt (decrypt) a letter c (within the alphabet A-Z or a-z) with a shift of k positions:

1. Let x be c's position in the alphabet (0 based), e.g., position of `B` is 1 and position of `g` is 6.
2. For encryption, calculate $y = x + k$ modulo 26;
   for decryption, calculate $y = x - k$ modulo 26.
3. Let w be the letter corresponding to position y in the alphabet. If c is in uppercase, the encrypted (decrypted) letter is w in lowercase; otherwise, the encrypted (decrypted) letter is w in uppercase.

A character which is not within the alphabet A-Z or a-z will remain unchanged under encryption or decryption.

Example. Given letter `B` and k = 3, we have x = 1, y = 1 + 3 mod 26 = 4, and w = `E`. As `B` is in uppercase, the encrypted letter is `e`.

Now, to encrypt/decrypt a sequence of characters:

The number of positions, k, used to shift a character is determined by a key V of n characters. For example, if V is a 4-character key 'C', 'O', 'M', 'P', and their positions in the alphabet is 2, 14, 12 and 15, respectively. To encrypt a sequence of characters, we shift the first character by +2 positions, the second by +14, the third by +12, the fourth by +15 and repeat the key, i.e., we shift the fifth character by +2, the sixth by +14, until we encrypt all the characters in the input sequence.

Example. Consider the input sequence of characters 'H','e','l','l','o','W','o','r','l','d' and a 4-character key 'C','O','M','P':

character	**H**	**e**	**l**	**l**	**o**	**W**	**o**	**r**	**l**	**d**
k	+2	+14	+12	+15	+2	+14	+12	+15	+2	+14
x	7	4	11	11	14	22	14	17	11	3
y	9	18	23	0	16	10	0	6	13	17
w	j	s	x	a	q	k	a	g	n	r
enrypted character	j	S	X	A	Q	k	A	G	N	R

Note: For decryption, we will shift by $-k$ instead of k.

Input:

- first line of input $s\ c_1\ c_2\ c_3\ \ldots$, where
  - s is either the character `e` for encryption, or the character `d` for decryption
  - $c_1\ c_2\ c_3\ \ldots$ is a sequence of space separated characters, ended by `!`, to be encrypted or decrypted
  - you may assume that the input number of characters to encrypt or decrypt (including `!`) is no greater than 50.
- second line of input $n\ v_1\ v_2\ \ldots\ v_n$, where n is the number of characters in the key, and v_1,v_2,...,v_n are the characters in the key.
  - you may assume that all characters in the key is in uppercase and the number of characters in the key is at least one and no greater than

10.

Output:

- the encrypted/decrypted message ended by `!`. There should be no space between two consecutive characters.

Requirement:

- You are provided with a template program `3.cpp`. Read the code in the template carefully to see what have been provided for you.
- You can ONLY use the simple data types `char`, `bool`, `int`, `double` and arrays. In other words, you are not allowed to use other data types or data structures such as strings or STL containers (e.g., vectors), etc.

Sample Test Cases

User inputs are shown in blue.

3_1

```
e !
3 A B C
!
```

3_2

```
e a B c D e !
2 X Y
XzZbB!
```

3_3

```
d X z Z b B !
2 X Y
aBcDe!
```

3_4

```
e H e l l o   E N G G 1 3 4 0 / C O M P 2 1 1 3 !
4 C O M P
jSXAQszvi1340/odod2113!
```

3_5

```
d f 3 O 3 B 3 I _ P Q R 3 _ Q K _ X 3 D I J K 3 V _ W B F 3 !
9 C O D E I S F U N
D3l3t3d_cod3_is_d3bugg3d_cod3!
```

Problem 4: A 5-Card Hand

Write a C++ program to generate a random hand of FIVE poker cards from a deck of 52 poker cards, and determine the type of poker hand it is. The input and output of your program as are follows:

Input:

- An integer x that you would use as input parameter to `srand()` for initializing the random number generator (RNG).

Output:

- the first line displays the five cards in the hand. (Note that there is a space at the end of the first output line.)
- the second line tells the type of the 5-card hand, in the following categories:
    - four of a kind (e.g., A♠ A♥ A♣ A♦ 2♣)
    - full house (e.g., 8♠ 8♥ 8♦ 4♥ 4♣)
    - flush (e.g., K♠ 10♠ Q♠ 5♠ 3♠)
    - three of a kind (e.g., 10♥ 10♦ 10♣ 5♥ J♠)
    - two pair (e.g., 9♦ 9♣ 2♥ 2♣ 3♥)
    - one pair (e.g., 7♠ 7♦ A♠ Q♠ 6♠)
    - others

      (Check the wiki page "list of poker hands" for the detailed definition of

each type.)

Requirement:

- You are provided with a template program `4.cpp`. Read the code in the template carefully to see what have been provided for you.
- Represent each card of the deck by an integer from 0 to 51 in the following ways:
  - A♠, 2♠, 3♠, ..., 10♠, J♠, Q♠, K♠ are represented by 0, 1, 2, ..., 12, respectively
  - A♥, 2♥, 3♥, ..., 10♥, J♥, Q♥, K♥ are represented by 13, 14, 15, ..., 25, respectively
  - A♣, 2♣, 3♣, ..., 10♣, J♣, Q♣, K♣ are represented by 26, 27, 28, ..., 38, respectively
  - A♦, 2♦, 3♦, ..., 10♦, J♦, Q♦, K♦ are represented by 39, 40, 41, ..., 51, respectively
- The array `cards` of 5 integers defined in the main function is used for storing a 5-card hand.
- Use the `rand()` function in `<cstdlib>` to generate random numbers. However, the RNG ONLY guarantees to generate the same sequence of random numbers given the same seed on the same OS & platform only. The sample test cases are generated on the CS academy* servers, so **you have to run the program there in order to generate the same results**.
- You may assume that the first 5 random numbers generated by `rand()` are all different.
- You can ONLY use the simple data types `char`, `bool`, `int`, `double`, `string` and arrays. In other words, you are not allowed to use other data types or data structures such as STL containers (e.g., vectors), etc.
- **You program MUST contain the following 8 functions:**

```
void DealHand(int cards[])
```

- The function `DealHand()` should read from user input an integer x and use it as seed to the RNG for generating five random numbers, each representing a card in a 52-card deck. The first five numbers generated by the RNG should be stored in the array `cards[]` in order.

```
void PrintHand(int cards[])
```

- The function `PrintHand()` should print a hand stored in `cards[]`. For example, if `cards[0] == 0`, `cards[1] == 25`, `cards[2] == 14`, `cards[3] == 50`, `cards[4] == 36`, the line `A♠ K♥ 2♥ Q♦ J♣` should be output to the screen.
- You may refer to the following program on how to display the suit symbols in UTF-8 encoding (on Linux/MacOS):

```
#include <iostream>

// include the following 4 lines in your program
// these define the UTF-8 encoding of the suit symbols
#define SPADE   "\xE2\x99\xA0"
#define CLUB    "\xE2\x99\xA3"
#define HEART   "\xE2\x99\xA5"
#define DIAMOND "\xE2\x99\xA6"

using namespace std;

int main()
{
  cout << SPADE << CLUB << HEART << DIAMOND << endl;
  return 0;
}
```

The other 6 functions that you must have in your program are:

```
bool IsFourOfAKind(int cards[])   // return if the hand is a four of a kind
bool IsFullHouse(int cards[])     // return if the hand is a full house
bool IsFlush(int cards[])         // return if the hand is a flush
bool IsThreeOfAKind(int cards[])  // return if the hand is a three of a kind
bool IsTwoPair(int cards[])       // return if the hand is a two pair
bool IsOnePair(int cards[])       // return if the hand is a one pair
```

These 6 functions should take a 5-card hand as input and return whether the hand is of a certain type. Note that these functions do not take the array size as an input parameter as we assume a hand always contain 5 cards in this problem (and a constant is defined in the code template).

Sample Test Cases

User inputs are shown in blue.

4_1

```
0
A♦ 10♥ Q♣ 5♦ 2♠
others
```

4_2

```
3
7♠ 9♥ 10♦ 4♥ 7♦
one pair
```

4_3

```
540
10♣ 6♠ 10♥ 3♠ 10♦
three of a kind
```

4_4

```
13
8♦ 8♣ 9♦ J♥ J♣
two pair
```