CS2315 Computer Programming Assignment Two (2022-23 Sem. A)

Deadline: **5-Nov-2022 23:59** (Week 10 Sat)   No Late submission will be accepted

Problem description

A prime number is a natural number which is divisible by exactly two factors – 1 and itself. Prime number is not that common, for example there are only 168 prime numbers in the range 1..1000. However from time to time, we could find numbers which *"look like"* prime numbers. In this assignment, you will work on a special type of number called "**xPrime**", which is defined as:

> *A composite number which will become a prime if two digits of the number at different positions get swapped.*

For example, 130 is an **xPrime**, because when '3' and '0' get swapped, it yields 103 which is a prime number. 20 is also an **xPrime**, because swapping the two digits yields 02 which is a prime. However, 37 is not an **xPrime**. You cannot treat the number as 037 and yield 307 by swapping. *(i.e. the length of input number should be preserved)*.

Occasionally, an **xPrime** could yield more than one prime when swapping different positions. For instance, 1357 is an xPrime, which could yield the prime 1753 *(by swapping '3' and '7')* as well as yielding the prime 7351 *(by swapping '1' and '7')*.

In this assignment, you will write a program which read in an non-negative integer and report whether the number is an xPrime.

- The input number is always in the simplest form. *(e.g. 3 will not be typed as 03)*
- The input number is no longer than 8 digits.
- No error checking is needed. *(e.g. it will not be non-number)*
- If the input is an **xPrime**, the program should print out all distinct prime number(s) formed by swapping, in ascending order of magnitude.
- If the input is not an **xPrime**, the program should print the **xPrime** which is closest to the input *(which may be greater or smaller)*. If there exists two **xPrimes** with the same distance. Print the smaller one.
- PASS will enforce a maximum runtime of 2 seconds per testcase.

In this assignment you may use *only* the functions from **<iostream>** and **<iomanip>** whenever appropriate. *(i.e. no other libraries like **<cmath>**, **<string>**, **<cstring>**, **<algorithm>** or **<cstdlib>**…etc.)*.

Sample Input / Output:

Example 1: *(User Input is underlined)*

```
Input an integer: 130
130 is xPrime formed by swapping 2 digits of:
031
103
```

Example 2: *(User Input is underlined)*

```
Input an integer: 20
20 is xPrime formed by swapping 2 digits of:
02
```

Example 3: *(User Input is underlined)*

```
Input an integer: 1357
1357 is xPrime formed by swapping 2 digits of:
1753
7351
```

Example 4: *(User Input is underlined)*

```
Input an integer: 37
37 is not xPrime
The nearest xPrime is: 38
```

Example 5: *(User Input is underlined)*

```
Input an integer: 44
44 is not xPrime
The nearest xPrime is: 38
```

Example 6: *(User Input is underlined)*

```
Input an integer: 99999998
99999998 is xPrime formed by swapping 2 digits of:
89999999
99899999
99998999
99999989
```

Marking criteria

Submitted program will be tested repeatedly with PASS. Marks will be graded objectively based on the number of correct outputs reported by PASS.

- If the program is not compilable, zero mark will be given.
- Make sure that the output format (spelling, punctuations, spacing…etc) follows *exactly* the sample output above otherwise PASS will consider your answer incorrect. Note that the marker will make NO manual attempt to check or re-mark program output.

Unlike tutorial exercises, for assignment, PASS will NOT make ALL test cases visible online. *(i.e. there are hidden test cases)*. Queries regarding hidden test case(s) will not be entertained. It is the responsibility of a programmer to design own test cases in order to verify the correctness of the written program.

Note that the marking in PASS is automatic, therefore the following situations may lead to extremely low or zero marks:

- Input/output does not match the requirements as defined.
  (e.g. incorrect spacing, incorrect spelling, letter cases or punctuations)
- Submission of non **.cpp** files *(e.g. .exe or .zip files)*
- Submission with the "**test**" function rather than the "**submit**" function
  (Students may test or submit many times, only the last "submit" will be marked)

Please also note that the PASS plagiarism check will also be turned on. **The same disciplinary action will be applied without distinguishing which one is the source / copier**. Please safeguard your files if you work on your assignment using public computers (e.g. CityU Lab)

Testing and Submission

Students should submit cpp file to PASS before the deadline. **No other submission method (e.g. hardcopy, email…etc) is accepted**. Students may use whatever IDE/compiler for development, but programming using non-standard, platform-specific features (which is not compilable in PASS/MS Visual Studio) will lead to zero mark. The marker will make no attempt to fix the syntax error and make the code compilable.

If you observe that the answer / runtime in PASS is different from the one you get in your PC, **most likely it is caused by programming bugs like uninitialized variables**. The result by PASS will be used for marking without consideration of what you observe in your local PC.

It is advised that students test the programs with PASS *(using the* Test *function)* before final submission *(with the "submit" button)*. Be warned that the system could be extremely busy and may become sluggish in responding near deadline. Only **.cpp** files are accepted. Do not submit **.obj** , **.exe** or any other files.

To protect yourself, it is advised that you write down your particulars (full name, student number, eid…) in the beginning of your source code *as comment*.