Cairo University, Faculty of Computers
and Artificial Intelligence

FACULTY OF COMPUTERS AND INFROMATION, CAIRO UNIVERSITY

CS213: Programming II
Fall 2024

First Semester

Problem Sheet 1 – Version 2.0

Course Instructors:

Dr. Mohammad El-Ramly

Revision History

Version 1.0	By Dr Mohammed El-Ramly	1 Oct. 2022	Main Doc
Version 2.0	By Dr Mohammed El-Ramly	11 Oct. 2023	Corrections
Version 2.0	Dr. Mohammad El-Ramly	1 Oct. 2024	Corrections

CS213: Object Oriented Programming
Problem Sheet 1 – C++ Revision

Cairo University, Faculty of Computers
and Artificial Intelligence

Objectives

This sheet includes programming problems for training on C++ Revision.

Strings, Arrays, Pointers, Vectors and Structures

1. Write a program that reads in a sentence of up to 100 characters and outputs the sentence with spacing corrected and with letters corrected for capitalization. In other words, in the output sentence, all strings of two or more blanks should be compressed to a single blank. The sentence should start with an uppercase letter but should contain no other uppercase letters. Do not worry about proper names; if their first letters are changed to lowercase; that is acceptable.

   Treat a line break as if it were a blank, in the sense that a line break and any number of blanks are compressed to a single blank. Assume that the sentence ends with a period and contains no other periods.

   For example, the input:

   ```
   the      Answer to  life,    the Universe, and everything IS 42.
   ```

   should produce the following output:

   ```
   The answer to life, the universe, and everything is 42.
   ```

2. Write a program that converts male speech to inclusive speech that includes both male and female. The program will ask for a sentence, read the sentence into a string variable, and replace all occurrences of masculine pronouns with both-gender pronouns. For example, it will replace "he" with "he or she". Thus, the input sentence.

   For example, the input:

   ```
   See an adviser and talk to him. He will guide you.
   ```

   will produce the output:

   ```
   See an adviser and talk to him or her. He or she will guide you.
   ```

   This will be a long program that requires a good deal of patience in order for your algorithm to consider all corner cases. Your program should not replace the string "**`he`**" when it occurs inside another word, such as "**`here`**". A word is any string consisting of the letters of the alphabet and delimited at each end by a blank, the end of the line, or any other punctuation letter, e.g. "**`Did you ask him`**?" is going to be "**`Did you ask him or her`**?" but "**`HE2 is a new store.`**" will stay the same. You do not need to differentiate between cases when you put "**`her`**" and cases when you put "**`hers`**" but you can try.

3. Given the following header:

   ```
   vector<string> split(string target, string delimiter);
   ```

   implement the function split so that it returns a vector of the strings in target that are separated by the string delimiter. For example:

   ```
   split("10,20,30", ",")
   ```

   should return a vector with the strings "10", "20", and "30". Similarly,

   ```
   split("do re mi fa so la ti do", " ")
   ```

   should return a vector with the strings "do", "re", "mi", "fa", "so", "la", "ti", and "do".

CS213: Object Oriented Programming
Problem Sheet 1 – C++ Revision

Cairo University, Faculty of Computers
and Artificial Intelligence

4. Prime numbers are very important in the field of security. See https://www.scienceabc.com/innovation/how-are-prime-numbers-used-in-cryptography.html

   In the 3rd century B.C., the Greek astronomer Eratosthenes developed an algorithm to find all the prime numbers up to some upper limit N. To apply the algorithm, you write down a list of the integers between 2 and N. For example, if N were 20, you would begin by writing down the following list:

   - 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

   Next you circle the first number in the list, indicating that you have found a prime. You then go through the rest of the list and cross off every multiple of the value you have just circled, since none of those multiples can be prime. Thus, after executing the first step of the algorithm, you will have circled the number 2 and crossed off every multiple of two, as follows:

   - 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~

   From this point, you simply repeat the process by circling the first number in the list that is neither crossed off nor circled, and then crossing off its multiples. In this example, you would circle 3 as a prime and cross off all multiples of 3 in the rest of the list, which would result in the following state:

   - 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~

   Eventually, every number in the list will either be circled or crossed out, as shown in this diagram:

   - 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~

   The circled numbers are the primes; the crossed-out numbers are composites. This algorithm for generating a list of primes is called the sieve of Eratosthenes. Write a program that uses the sieve of Eratosthenes to generate a list of the primes between 2 and a given number *n*.

5. Write a program that manages a list of up to 10 players and their high scores in the computer's memory. Use suitable arrays, vector and/or structures to manage the list. The program will support these features:
   (a) Add a new player and score. If it is one of the top 10 scores then add it to the list of scores and remove the player with the smallest score. The same name and score can appear multiple times. For example, if Bill played 3 times and scored 100, 100, and 99, and Bob played once and scored 50, then the top scores would be Bill 100, Bill 100, Bill 99, Bob 50.
   (b) Print the top 10 names and scores to the screen sorted by score with the highest score first.
   (c) Allow the user to enter a player name and output that player's highest score if it is on the top 10 list or a message if the player's name has not been input or is not in the top 10.
   (d) Create a menu system that allows the user to select which option to invoke.

Recursion

6. You are asked to write 2 printing functions with these specifications.
   a) **One Binary Number**. Write a method with this specification:

   ```
   public static void binaryPrint(int n)
   ```

   The number **n** is non-negative. The method prints the value of **n** as a BINARY number. If **n** is zero, then a single zero is printed; otherwise no leading zeros are printed in the output.
   The `'\n'` character is NOT printed at the end of the output.

   Examples:

   ```
   n = 0    Output:0
   n = 4    Output:100
   n = 27   Output:11011
   ```

   NOTE: Your recursive implementation must not use any local variables.

Cairo University, Faculty of Computers
and Artificial Intelligence

b) **Printing Many Numbers**. Write a method with this specification:

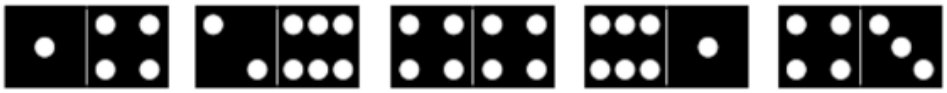
```
public static void numbers(String prefix, int k);
```

The number **k** is non-negative. The argument called prefix is a String of 0's and 1's. The method prints a sequence of binary numbers. Each output number consists of the prefix followed by a suffix of exactly **k** more binary digits (0's or 1's). All possible combinations of the prefix and **k**-digit suffix are printed. For example, if prefix is the string "00101" and **k** is 2, then the method would print the prefix followed by the 4 possible suffixes shown on the side.

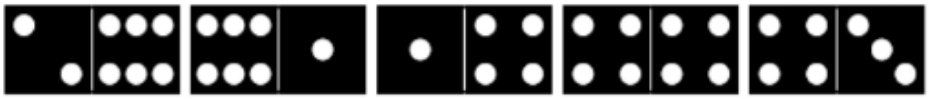
```
0010100
0010101
0010110
0010111
```

The stopping case occurs when **k** reaches zero (in which case the prefix is printed once by itself followed by nothing else).

7. **The game of dominos** is played with rectangular pieces composed of two connected squares, each of which is marked with a certain number of dots. For example, each of the following five rectangles represents a domino:



Dominos can be connected end-to-end to form chains, subject to the condition that two dominos can be linked together only if the numbers match. For example, you can form a chain consisting of all five of these dominos by connecting them in the following order:



Note: In traditional dominos, you can rotate a domino by 180˚ so that its numbers are reversed. As a simplification in this problem, this operation is not allowed, which means that the dominos must appear in their original orientation.

Dominos can, of course, be represented in C++ using the following structure type:

```
struct dominoT {
   int leftDots;
   int rightDots;
};
```

(a) Given this abstract domino type, write a recursive function

```
bool FormsDominoChain(vector<dominoT> & dominos);
```

that returns **true** if it possible to build a chain consisting of every domino in the vector. For example, if you initialized the variable **dominoSet** to contain the five **dominoT** values shown at the top of the page, calling **FormsDominoChain(dominoSet)** would return **true** because it is possible to form the chain shown in the second diagram. On the other hand, if you remove the first domino, calling **FormsDominoChain(dominoSet)** returns false because there is no way to form a domino chain if the 1-4 domino is missing.

(b) Print the solution of the domino chain that you generated above in simple format (no need to have pictures). For example:

```
2|6 - 6|1 - 1|4 - 4|4 - 4|3
```

8. **A Fractal Pattern**. Examine this pattern of asterisks and blanks, and write a recursive method that can generate patterns such as this.

```
*
* *
  *
* * * *
    *
    * *
      *
* * * * * * * *
        *
        * *
          *
        * * * *
            *
            * *
              *
```

With recursive thinking, the method needs only seven or eight lines of code (including two recursive calls). Your method should look like this:

```
public static void pattern(int n, int i)
// Precondition: n is a power of 2 greater than zero.
// Postcondition: A pattern based on the above example has been
// printed. The longest line of the pattern has
// n stars beginning in column i of the output. For example,
// The above pattern is produced by the call pattern(8, 0).
```

9. **Teddy Bear Picnic.** Build a game with teddy bears. It starts when I give you some bears. You can then give back some bears, but you must follow these rules (**n** is the number of bears that you have):

   1. If **n** is even, then you may give back exactly **n / 2** bears.
   2. If n is divisible by 3 or 4, then you may multiply the last two digits of n and give back this many bears. (By the way, the last digit of n is **n % 10**, and the next-to-last digit is **((n % 100) / 10).**
   3. If n is divisible by 5, then you may give back exactly 42 bears.

   The game goal is to end with **exactly** 42 bears. E.g., if we start with 250 bears, we can make the moves:

   - Start with 250 bears.
   - Since 250 is divisible by 5, you may return 42 of the bears, leaving you with 208 bears.
   - Since 208 is even, you may return half of the bears, leaving you with 104 bears.
   - Since 104 is even, you may return half of the bears, leaving you with 52 bears.
   - Since 52 is divisible by 4, you may multiply the last two digits (resulting in 10) and return these 10 bears. This leaves you with 42 bears.
   - You have reached the goal!

   Write a recursive method to meet this specification:

```
public static boolean bears(int n)
// Postcondition: A true return value means it is possible to win
// the bear game by starting with n bears. A false value means that
// it is not possible to win the bear game starting with n bears.
// Examples:
// bear(250) is true (as shown above)
// bear(42) is true
// bear(84) is true
// bear(53) is false
// bear(41) is false
```

CS213: Object Oriented Programming
Problem Sheet 1 – C++ Revision

Cairo University, Faculty of Computers
and Artificial Intelligence

Files

10. **Message Altering to Avoid Censorship.** In time of conflict, as currently, social media platforms usually have bias to one side and censor content or posts supporting the other side. Write a C++ programs that takes a file in Arabic with some message or post. Then, it changes words that could be censored by the platform due to the view they represent with other alternatives. Program should have a look up table loaded from another file that has a list of words and the alternative(s) for each word.
    - **Hint:** You need to learn how to handle Unicode in C++.
    - **Hint:** You can allow multiple alternatives for each word and pick from them randomly.
    - **Hint:** If you do not speak Arabic, you can do it for English messages.

11. **File Comparison.** You are asked to write a file comparison facility that asks the user for the names of two files to compare. It also asks the user if he wants a character by character comparison or word by word comparison.
    (a) In case of character by character comparison, the program will display if the files are identical or it will display the number and the content of the first line that is different. White spaces are compared.
    (b) In case of word by word, white spaces, tabs and new lines are ignored and files are compared word by word. The result is either identical or the first different word and the line that contains it.

12. **Phishing Scanner**. Phishing is a form of identity theft in which, in an e-mail, a sender posing as a trustworthy source attempts to acquire private information, such as your user names, passwords, credit-card numbers and social security number. Phishing e-mails claiming to be from popular banks, credit-card companies, auction sites, social networks and online payment services may look quite legitimate. These fraudulent messages often provide links to spoofed (fake) websites where you're asked to enter sensitive information.

    Visit **`www.snopes.com`** and other websites to find lists of the top phishing scams.

    Also checkout the Anti-Phishing Working Group

    `www.antiphishing.org/`

    and the FBI's Cyber Investigations website

    `https://www.fbi.gov/about-us/investigate/cyber/cyber`

    where you'll find information about the latest scams and how to protect yourself.

    Create a list of 30 words, phrases and company names commonly found in phishing messages. Assign a point value to each based on your estimate of its likeliness to be in a phishing message (e.g., one point if it's somewhat likely, two points if moderately likely, or three points if highly likely). Write a program that scans a file of text for these terms and phrases. For each occurrence of a keyword or phrase within the text file, add the assigned point value to the total points for that word or phrase. For each keyword or phrase found, output one line with the word or phrase, the number of occurrences and the point total. Then show the point total for the entire message. Does your program assign a high point total to some actual phishing e-mails you've received? Does it assign a high point total to some legitimate e-mails you've received?