# Assignment 4

## Instructions

- Max score is 10.
- Deadline is 11:59PM, Oct 7$^{th}$ , Saturday.
- Extra credits are added only if score is less than 10.
- Try to attempt every question and keep practising from other online sites.
- Upload all assignments to a single repository named ‘Assignments’.
- Don’t use any direct methods for this assignment.
- Contact us if any assistance is needed.

## Questions

1. Now you are given a string S, which represents a software license key which we would like to format. The string is composed of alphanumerical characters and dashes. The dashes split the alphanumerical characters within the string into groups.(i.e If they are M dashes, the string is split into M+1 groups). The dashes in the given string are possibly misplaced.

   We want each group of characters to be of length K (except for possibly the first group, which could be shorter, but still must contain at least one character). To satisfy this requirement, we will reinsert dashes. Additionally, all the lower case letters in the string must be converted to upper case.

   So, you are given a non-empty string S, representing a license key to format, and an integer K. And you need to return the license key formatted according to the description above. (Score 2)

   Example 1:
   Input: S = “2-4A0r7-4k”, K = 4
   Output: “24A0-R74K”
   Explanation: The string S has been split into two parts, each part has 4 characters.

   Example 2:
   Input: S = “2-4A0r7-4k”, K = 3
   Output: “24-A0R-74K”
   Explanation: The string S has been split into three parts, each part has 3 characters, except the first part as it could be shorter as said above.

   Note:
   The length of string S will not exceed 12,000, and K is a positive integer. String S consists only of alphanumerical characters (a-z and/or A-Z and/or 0-9) and dashes(-).String S is non-empty.

2. Implement a class called Tool. It should have an int field called strength and a char field called type. You may make them either private or protected. The Tool class should also contain the function void setStrength(int), which sets the strength for the Tool.

   Create 3 more classes called Rock, Paper, and Scissors, which inherit from Tool. Each of these classes will need a constructor which will take in an int that is used to initialise the strength field. The constructor should also initialise the type field using ‘r’ for Rock, ‘p’ for Paper, and ’s' for Scissors.

   These classes will also need a public function bool fight(Tool) that compares their strengths in the following way:

   Rock’s strength is doubled (temporarily) when fighting scissors, but halved (temporarily) when fighting paper. In the same way, paper has the advantage against rock, and scissors against paper. The strength field shouldn’t change in the function, which returns true if the original class wins in strength and false otherwise. You may also include any extra auxiliary functions and/or fields in any of these classes. Run the program without changing the main function, and verify that the results are correct. (Score 2)

```
class Tool{

    //add your code here
}

/*

    Implement class Scissors

*/

/*

    Implement class Paper

*/

/*

    Implement class Rock

*/

class RockPaperScissorsGame{

    public static void main(String args[]){

        Scissors s = new Scissors(5);
        Paper p = new Paper(7);
        Rock r = new Rock(15);

        System.out.println(s.fight(p) + " , "+ p.fight(s) );
        System.out.println(p.fight(r) + " , "+ r.fight(p) );
        System.out.println(r.fight(s) + " , "+ s.fight(r) );

    }
}
```

3. Every computer on the Internet has a unique identifying number, called an Internet protocol (IP) address. To contact a computer on the Internet, you send a message to the computer's IP address. Here are some typical IP addresses:

   216.27.6.136
   224.0.118.62

   There are different formats for displaying IP addresses, but the most common format is the dotted

decimal format. The above two IP addresses use the dotted-decimal format. It's called "dotted" because dots are used to split up the big IP address number into four smaller numbers. It's called "decimal" because decimal numbers are used (as opposed to binary) for the four smaller numbers.

Each of the four smaller numbers is called an octet because each number represents eight bits (oct means eight). For example, the 216 octet represents 11011000 and the 27 octet represents 00011011.

Implement an IpAddress class that stores an IP address as a dotted-decimal string and as four octet ints.

You must implement all of the following:

Instance variables:
dottedDecimal – a dotted-decimal string. Example value: "216.27.6.136" firstOctet, secondOctet, thirdOctet, fourthOctet – four int variables that store the octets for an IP address

Constructor:
This constructor receives one parameter, a dotted-decimal string. You may assume that the parameter's value is valid (i.e., no error checking required). The constructor initialises the instance variables with appropriate values. There are many ways to solve the problem of extracting octets from the given dotted-decimal string. We recommend that you use String methods to extract the individual octets as strings, and then use parseInt method calls to convert the octet strings to ints.

getDottedDecimal method:
This is a standard accessor method that simply returns the dottedDecimal instance variable's value.

getOctet method:
This method receives the position of one of the octets (1, 2, 3, or 4) and returns the octet that's at that position.

Provide a driver class that tests your IpAddress class. Your driver class should contain this main method: (Score 2)

```
public static void main(String args[]){

    IpAddress ip = new IpAddress("216.27.6.136");
    System.out.println(ip.getDottedDecimal());
    System.out.println(ip.getOctet(4));
    System.out.println(ip.getOctet(1));
    System.out.println(ip.getOctet(3));
    System.out.println(ip.getOctet(2));

}
```

Using the above main method, your program should generate the following output.

Sample output:

```
216.27.6.136
136
216
6
27
```

4. Design a simple registration system that allows Student to register in a course using 2 classes: class Student & class Course. Implement the scenarios in class Test's main method.

   Each student has a name and an id variables. Each object of class Student is initialised using values of name and id passed to constructor. Class Student has accessor methods for its instance variables

   Each Course has a name, and a variable numberOfStudent representing the number of registered students. A course can have a maximum number of 10 students registered in it. Class Course store the registered students in students which is an array of type Student. When a student register in a course, he is added to the array. Each object of class Course is initialised using the title. Class Course has the following methods: method getStudents(): return the array of registered students; method boolean isFull(): return true if the course is full, accessor method for the title and numberOfStudent field, method registerStudent (Student student): if the course is not full, register a student in course. (Score 2)

5. Given an integer, convert it to a roman numeral. Input is guaranteed to be within the range from 1 to 3999. (Score 2)

   Here is the prototype you can work with

```
    public String intToRoman(int num) {

    }
```

## Extra credit

1. There are two sorted arrays nums1 and nums2 of size m and n respectively. Find the median of the two sorted arrays. (Score 2)
   Example 1:

```
nums1 = [1, 3]
nums2 = [2]

The median is 2.0
```

Example 2:

```
nums1 = [1, 2]
nums2 = [3, 4]

The median is (2 + 3)/2 = 2.5
```

Here is the prototype you can work with.

```
    public double findMedianSortedArrays(int[] nums1, int[] nums2) {

    }
```