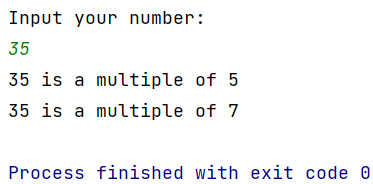
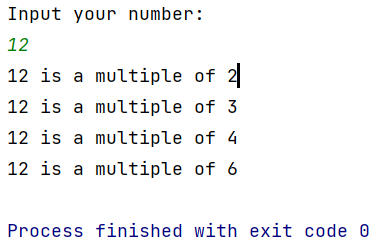
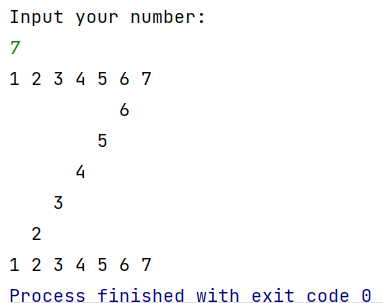
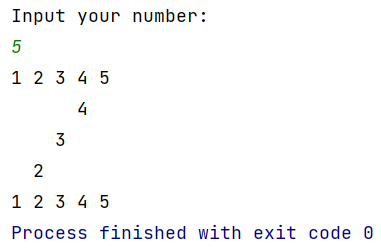
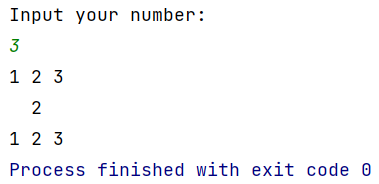
1. Write a program to determine if the given positive integer number (1~100) is a multiple of [**2 to 9**]. The program should take an input number from the user. (10pts)  
     
   **Output example)**



1. Write a program that prints out number-Z! The program should take the height of “Z” from the user. (15pts)

* Assume that we only take the number >= 3 from the user.
* No hard-coded solutions are allowed.

**Output example)**



1. Write a program to determine whether the given number is an Armstrong number or not. The program should take an input number from the user (15pts)

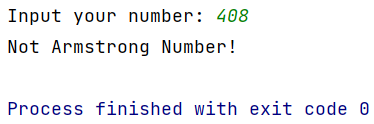
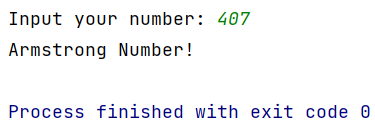
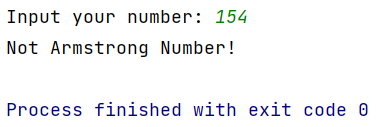
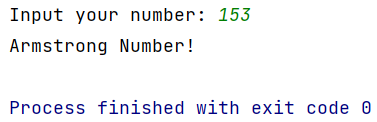
* Amstrong number: A number that is equal to the sum of the cubes of its own digits
  + 153 = 1^3 + 5^3 + 3^3
  + 370 = 3^3 + 7^3 + 0^3
  + 371 = 3^3 + 7^3 + 1^3
  + 407 = 4^3 + 0^3 + 7^3
* Assume that we only take a 3digits number from the user.
* Use the following template: main method cannot be modified

public class Q3 {  
 public static void main(String[] args) {  
 Scanner scn = new Scanner(System.*in*);  
 System.*out*.println("Input your number: ");  
 int num = scn.nextInt();  
  
 boolean isArm = *isArmstrong*(num);  
  
 if(isArm) System.*out*.println("Armstrong Number!");  
 else System.*out*.println("Not Armstrong Number!");  
 }  
  
 static boolean isArmstrong(int num){

// you must implement this method

}  
}

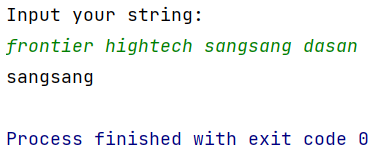
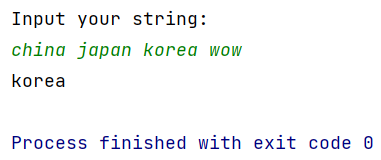
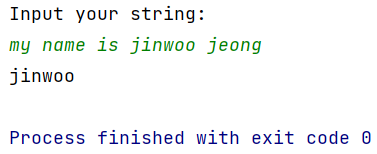
**Output example**



1. Write a program to find the longest word from the given string. The program should take an input string from the user. (15pts)

* If there exist multiple candidate words with the longest length, **print the word that appears last.**
* Assume that the string consists of alphabets only.
* Assume that only a single (white)space will be used as a delimiter.

**Output example)**

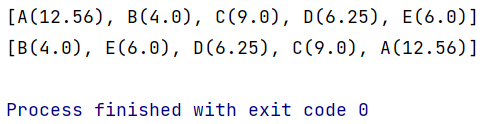


1. Complete the blanks in the following code template: (20pts)

* In the main method of *AreaSorter* class, we add a set of *Shape* instances to the list. Then, we invoke *sort()* method of *Collections* class with this list instance. By doing this, we can get a new list **sorted based on the area** of elements in it.
* *Circle* takes “radius” and “name” as constructor parameters
* *Rectangle* takes “width”, “height”, and “name” as constructor parameters
* *Square* takes and “side-length” and “name” as constructor parameters
* When printing the list instance, the program should be able to display **the name and the area of each element** in the list instance.
* Use the given template code below.
* You **CANNOT** modify “**main**”method of AreaSorter class.

**Output example)**

Program’s output just consists of the following two lines: 1) elements of the list before sorting, 2) elements of the list after sorting.



* Code template)

import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
  
class Shape implements Comparable<Shape>{

double area=0;  
String name="";

public Shape(String name) {  
 this.name = name;  
 }

// Implement here

}  
  
// Implement Circle, Rectangle, Square classes here

public class AreaSorter{  
 public static void main(String[] args) {  
 List<Shape> list = new ArrayList<Shape>(); // creation of the list<Shape>  
 list.add(new Circle(2.0, "A")); // adding elements to the list  
 list.add(new Rectangle(2.0,2.0,"B"));  
 list.add(new Square(3.0,"C"));  
 list.add(new Square(2.5,"D"));  
 list.add(new Rectangle(2.0,3.0,"E"));  
  
 System.*out*.println(list); *// before sorting* Collections.*sort*(list);  
 System.*out*.println(list); *// after sorting* }  
}

1. Write a program to search for lines **containing a specific string** in the file. (25pts)

* The program should take an input string to search from the user
* Use the given “file.txt” file
* **Case-insensitive search** must be supported
* **Multi-word search** must be supported
* If the target string does not exist in the file, a message indicating that “it is not present in the file” must be printed out.

**Example output)**

