Question Pool

This document contains questions in different topics. Answers are given at the end of the document. Some exercises have stars (*); the number of stars is proportional to the difficulty of the question.

1. Introduction

No extra questions yet.

2. Variables – Integer & Float

No extra questions yet.

3. Conditionals

Exercise 1: Every morning at 10 am, Lucy looks at the thermometer and the humidity meter and decides what to eat or drink according to the temperature and the humidity levels. The following table shows what Lucy eats or drinks in what conditions. For example, if the temperature is 45°F and the humidity is 53% then Lucy eats cake.

	Temperature < 60°F	**Temperature ≥ 60°F**
Humidity < 40%	Lucy drinks coffee	Lucy drinks iced tea
Humidity ≥ 40%	Lucy eats cake	Lucy eats ice cream

Write a program that reads two numbers, the temperature and the humidity accordingly, from the input file and writes what Lucy does to the output file.

Sample:

input:	*output*:
`45 53`	`Lucy eats cake`

Exercise 2: Write the same program with the conditions below.

	Temperature < 60°F	**Temperature ≥ 60°F**
Humidity < 40%	Lucy drinks coffee	Lucy drinks iced tea
40% ≤ Humidity < 60%	Lucy plays tennis	Lucy plays golf
Humidity ≥ 60%	Lucy eats cake	Lucy eats ice cream

Exercise 3: Consider the same example, but this time you will have `temperature` and `humidity` information for more days. You will output the days Lucy plays tennis or golf. In the input file, the first line will provide `N`, the number of days. Upcoming N lines will provide the `temperature` and `humidity` information for these days.

Sample:

```
 input:                            output:
-----------------------------------------------------------------
 4                                 Day 2: Lucy plays tennis
 45 83                             Day 4: Lucy plays golf
 58 58
 77 32
 77 40
-----------------------------------------------------------------
```

4. Looping

Exercise 1: Write a program which prints all two-digit numbers that can be composed of {0, 1, 2, 3, 4, 5}.

Exercise 2: Write a program which prints all two-digit numbers that can be composed of {0, 1, 2, 3, 4, 5} in the following order:

50 51 52 53 54 55 40 41 42 43 44 45 … 10 11 12 13 14 15

Be careful that 1s digit increases while 10s digit decreases.

* *Exercise 3*: Write a program which prints all two-digit odd numbers that can be composed of {0, 1, 2, 3, 4, 5, 6}.

* *Exercise 4*: Write a program which prints all three-digit numbers that can be composed of {0, 1, 2, 3, 4}.

** *Exercise 5*: Write a program that reads two numbers N and M from the input. Then, your program will print a figure on the screen that has N number of rows and M number of columns. The first row will have with M number of 1s, and the second row will have M number of 2s. It will continue until the Nth row will have M number of Ns. See the example below for clarity.

Sample:

```
 input:                            output:
```

```
3 4                         1111
                            2222
                            3333
```

** *Exercise 6*: Write a program that reads a number N from the input. Then, your program will print a figure on the screen that has `N` rows in triangle from. The first row will have only 1, and the second row will have 1 and 2. It will continue until the `N`th row will have the numbers from 1 to N. See the example below for clarity. Be careful about the blank between numbers.

Sample:

input: *output*:

```
4                           1
                            1 2
                            1 2 3
                            1 2 3 4
```

** *Exercise 7*: Write a program that reads a number N then N numbers from the input. Your program will print a figure on the screen that has N rows. The numbers given in the input specify the number of ‘*’ in their row consequently. See the example below for clarity.

Sample:

input: *output*:

```
5                           ***
3 4 6 0 3                   ****
                            ******

                            ***
```

* *Exercise 8*: Write a program which prints the output below when 4 and 7 are given as input. The program reads two numbers; N and M from keyboard.

Sample:

input: *output*:

```
4 7                         1 2 3 4 5 6 7
                            2 3 4 5 6 7 8
                            3 4 5 6 7 8 9
                            4 5 6 7 8 9 10
```

*** *Exercise 9*: Write a program which finds the primes that are less than the given number.

Sample:

input:	*output*:
`20`	`2 3 5 7 11 13 17 19`

** *Exercise 10*: Write a program that finds all the 3-digit numbers that composed of {0, 1, 2, 3, 4, 5} and whose digits are distinct.

Exercise 11: Write a program that reverses the given number.

Sample:

input:	*output*:
`2431`	`1342`

Exercise 12: Write a program that converts a number N from base M to base 10. Read N and M from the keyboard.

Sample:

input:	*output*:
`12345 3`	`179`

$12345_3 = 1 \text{ x } 3^4 + 2 \text{ x } 3^3 + 3 \text{ x } 3^2 + 4 \text{ x } 3^1 + 5 \text{ x } 3^0 = 81 + 54 + 27 + 12 + 5 = 179$

* *Exercise 13*: Write a program that converts a number N from base 10 to base M. Read N and M from the keyboard.

Sample:

input:	*output*:
`179 3`	`12345`

$12345_3 = 1 \text{ x } 3^4 + 2 \text{ x } 3^3 + 3 \text{ x } 3^2 + 4 \text{ x } 3^1 + 5 \text{ x } 3^0 = 81 + 54 + 27 + 12 + 5 = 179$

5. Arrays

** *Exercise 1*: Write a program that reads *n* < 100 then *n* numbers each of which is between 1 and 1000 inclusively from the input. Then, your program will find the numbers that exist more than once and write them on the screen.

** *Exercise 2*: Write a program that reads *n* < 100 then *n* distinct numbers each of which is between 1 and 1000 inclusively from the input. Then, your program will write the numbers on the screen in sorted order from smaller to larger.

*** *Exercise 3*: Write a program that reads *n* < 100 then *n* numbers each of which is between 1 and 1000 inclusively from the input. Then, your program will write the numbers on the screen in sorted order from smaller to larger. Note that numbers may not be distinct.

** *Exercise 4*: Write a program that outputs the intersection of two sets of sizes *n* < 100 and *m* < 100. Each number in the array is between -1000 and 1000 inclusively. Read *n*, then *n* numbers; read *m*, then *m* numbers.

Sample:

input: *output*:

```
5                                    30 40 50
30 -111 40 50 28
6
892 30 40 50 0 -892
```

** *Exercise 5*: Write a program that gives a sorted output of numbers given in two sorted arrays of sizes *n* < 100 and *m* < 100. Each number in the array is between -1000 and 1000 inclusively. Read *n*, then *n* numbers; read *m*, then *m* numbers. Write the sorted output on the screen.

Sample:

input: *output*:

```
6                                    -18 3 4 9 12 12 36 66 156 876 991
3 4 9 12 36 991
5
-18 12 66 156 876
```

*** *Exercise 6*: Write a program that gives a sorted output of numbers given in two sorted arrays of sizes *n* < 100 and *m* < 100. The numbers in the array may not be between -1000 and 1000. Read *n*, then *n* numbers; read *m*, then *m* numbers. Write the sorted output on the screen.

Sample:

input:	*output*:

`6`
`3 4 9 12 36 2191`
`5`
`-18 12 66 156 876`
	`-18 3 4 9 12 12 36 66 156 876 2191`

6. Variables – Char & String

Exercise 1: Lucy wants to calculate her bonus points she earned while shopping. She knows how much she spent in her last N ≤ 100 shopping, and all of them are *bakery*, *produce*, or *drinks*.

- If they are *bakery*, she earns 5 points per dollar.
- If they are *produce*, she earns 2 points per dollar.
- If they are *drinks*, she earns 1 point per dollar.

She earns bonus points when she spends more than M dollars in one shopping. You will first read N then N numbers corresponding to the money Lucy spend. At the end, you will read M then the type of shopping items. Write the total bonus points on the screen.

Sample:

input:	*output*:

`5`
`41 13 88 50 72`
`50 bakery`
	`800`

Lucy receives bonus points for shopping more than $50 hence $88 and $72 are qualified for bonus. Her shopping was on *bakery* which brings 5 points per dollar. In total, she got (88 + 72) * 5 = 800 bonus points.

* *Exercise 2*: Write a program that outputs the number of letters in a given string.

Sample:

input:	*output*:
`I8-10caKes!`	`6`

I, c, a, K, e, and s are the letters.

* *Exercise 3*: Write a program that converts all letters to uppercase in a given string.

Sample:

input:	*output*:
`I8-10caKes!`	`I8-10CAKES!`

** *Exercise 4*: Write a program that checks if two given strings are the same regardless of the case differences. If the strings are same, write ‘`same`’ otherwise ‘`not same`’ on the screen.

Sample:

input:	*output*:
`Trivial Trival`	`not same`

input:	*output*:
`tRiViAl TrIvIaL`	`same`

Exercise 5: Write a program that reverses a given string.

Sample:

input:	*output*:
`Reverse!`	`!esreveR`

* *Exercise 6*: Complete the following program so that it reverses a given string. Note that **you cannot declare another string**.

```
#include <iostream>
using namespace std;

int main()
{
  string str;
  cin >> str;

  // write your code here

  cout << str << endl;
}
```

Sample:

input:	*output*:
`Reverse!`	`!esreveR`

* *Exercise 7*: Write a program that reads two strings from the keyboard and checks if the number of vowels in these strings are equal.

Sample:

input:	*output*:
`ThReE VoWeLs`	`Same number of vowels.`

input:	*output*:
`twO VoWeLs`	`Not same number of vowels.`

In the first input, both strings have two vowels whereas in the second one, the first string has only one vowel.

* *Exercise 8*: Write a program that reads a string and writes the number of digits in that string on the screen.

Sample:

input:	*output*:
`I8-10caKes!`	`3`

3 of them are digits; 8, 1, and 0.

Exercise 9: Write a program that reads a number *x* and a character *c* from the keyboard. The program outputs a square with the size *x* filled with character *c*.

Sample:

input:	*output*:
`5 #`	`#####` `#####` `#####` `#####` `#####`

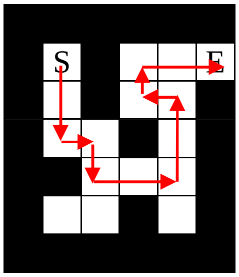
* *Exercise 10*: Write a program that reads a number *x* and a character *c* from the keyboard. The program outputs an empty square with the size *x* and border with character *c*.

Sample:

input:	*output*:

`5 #`	`#####`
	`#   #`
	`#   #`
	`#   #`
	`#####`

** *Exercise 11*: You will be given a map of a maze and a path. Check if the path is a correct way to the exit. You will start at position (1, 1), and exit is at (1, *m* – 1). An example maze of size (7, 6) is given below. Starting and exit locations are shown as 'S' and 'E'. A path is shown with red color to the exit.



Sample:

input:

```
7 6
######
#.#...
#.#..#
#..#.#
##...#
#..#.#
######
ddrdrruulurr
```

output:

```
Right path!
```

You will first read the size of the maze (*n*, *m*). Then the maze will be given; '#'s are walls, '.'s are empty locations. The size of the maze will not be more than (20, 20). The path is given as a string composed of four different letters; 'l' means 'left', 'r' means 'right', 'u' means 'up', and 'd' means 'down'.

You will write '`Right path!`' if the path leads to the exit from the starting location, otherwise write '`Wrong path!`' on the screen.

*** *Exercise 12*: write a program that reads a number $n \leq 20$ from the keyboard and prints the spiral of asterisks on the screen as shown below.

Sample:

n = 5

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

n = 8

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