

# Laboratory Assignment 3

**Important Note:** In this lab (and thereafter) you should NOT use `input()` function in any of your problems unless it is explicitly told in the problem. You will get your inputs as function parameters.



## Problem 1 - Find the King [10 pts]

Your function should have a parameter that will hold a string. This string will be the cards from a deck of cards. Return True if there is a King in the cards, False if not.

- String will not be ordered, can be any order.
- Function parameter name does not matter, but the function should only expect 1 parameter.
- inputs :  $a = \{ x: x \in \{1,2,3,4,5,6,7,8,9,0,J,Q,K\} \text{ and } \text{len}(x) \in [1:52] \}$
- output : bool

```
>>> problem1('12313')
False
```

```
>>> problem1('K713')
True
```

```
>>> problem1('Q')
False
```

```
>>> problem1('631J43K146')
True
```

## Problem 2 - Minimum of the four [10 pts]

Write a function that expects four parameters as numbers, and returns the minimum of these numbers.

- Function parameter names do not matter, but the function should expect 4 parameters.
- inputs :  $a, b, c, d \in \mathbb{R}$
- output : float

```
>>> problem2(2.3, -2.4, -1, 4.3)
-2.4
```

## Problem 3 - Round to the nearest [10 pts]

Write a function that expects two parameters as numbers, rounds the first number **closer to the second number** and returns the result.

- You should always round the first number.
- Function parameter names do not matter, but the function should expect 2 parameters.

- In the case of indecisivity, round up.
- inputs :  $a, b \in \mathbb{R}$
- output : integer

```
>>> problem3(2.6, 1)
2
```

```
>>> problem3(2.6, 2)
2
```

```
>>> problem3(2.6, 3)
3
```

```
>>> problem3(3, 10)
3
```

```
>>> problem3(3, -10)
3
```

```
>>> problem3(2.6, 5)
3
```

```
>>> problem3(-0.7, 10)
0
```

```
>>> problem3(-0.7, -10)
-1
```

```
>>> problem3(0.7, 0.6)
1
```

Explanation: between two numbers (0, 1), 1 is closer to 0.6.

```
>>> problem3(0.7, 0.9)
1
```

```
>>> problem3(0.7, 0.3)
0
```

```
>>> problem3(0.7, 0.5)
1
```

Explanation: between two numbers (0, 1), both 0 and 1 have the same distance to 0.5, this is a case of indecisivity, so we round up to 1.

## Problem 4 - Volume of a cylinder [10 pts]

Write a function that will calculate and return the volume of a cylinder.

- This function should expect three parameters. First two parameters will denote radius and height, and the last parameter will denote pi.
- First two parameters should be named radius and height, and do not have a default value.
- Last parameter, named pi, should have a default value as: 3.1415
- **These parameter names are important for your problem to be counted as correct.**
- inputs : radius, height, pi  $\in \mathbb{R}$
- output : integer / float

```
>>> problem4(2, 3)
37.698
```

```
>>> problem4(2, 3, pi=3)
36
```

```
>>> problem4(2, 3, 3)
36
```

```
>>> problem4(radius=2, height=3)
37.698
```

## Problem 5 - Volume of a cylinder with error check [10 pts]

Write a function that will calculate and return the volume of a cylinder.

- This function should expect three parameters. First two parameters will denote radius and height, and the last parameter will denote pi.
- First two parameters should be named radius and height, and do not have a default value.
- Last parameter, named pi, should have a default value as: 3.1415
- These parameter names are important for your problem to be counted as correct.
- **You need to make sure the parameters are integer/float types that you can calculate. If the parameter is not one of these, you should return -1 instead.**
- inputs : radius, height, pi : { x : type(x)  $\in$  {integer, float, string, NoneType, bool}}
- output : integer / float

```
>>> problem5(2, 3)
37.698
```

```
>>> problem5(2, 3, pi=3)
36
```

```
>>> problem5(2, 3, 3)
```

36

```
>>> problem5(radius=2, height=3)
37.698
```

```
>>> problem5(radius='test', height=3)
-1
```

```
>>> problem5(2, 'test', 3)
-1
```

```
>>> problem5(2, 2, True)
-1
```

```
>>> problem5(2, None, 3)
-1
```

## Problem 6 - Mr. Lonely [18 pts]

Write a function that will find and return the only character that does not have a duplicate in a given string. The returned character needs to be a string.

- There can be multiple copies of a given character.
- The given string will have exactly **one character** without a pair.
- String will not be ordered, can be any order.
- Function parameter name does not matter, but function should expect 1 parameter.
- inputs :  $a = \{ x: x \in \text{printable characters except whitespaces and } \text{len}(x) \in [1:200] \}$
- output : string

```
>>> problem6('12a31a333')
2
```

Explanation: There are four 1's, two 3's and two a's. Only 2 does not have a pair.

```
>>> problem6('7b3b17C3C1b7z3')
z
```

Explanation: There are two 7's, two 3's, three b's, 2 C's and two 1's. Only z does not have a pair.

```
>>> problem6('t')
t
```

```
>>> problem6('00631543146')
5
```

## Problem 7 - Ascending ordered string [15 pts]

Write a program that will find and return if the characters are in ascending order. (abcde...z).

- The letters do not need to be consecutive, look only for ascending order. (i.e. 'az' is fine)
- Same letter might repeat multiple times.
- inputs :  $a = \{ x: x \in \text{lowercase English letters and } \text{len}(x) \in [1:200] \}$
- output : boolean

```
>>> problem7('abcd')
```

```
True
```

```
>>> problem7('abbbbc')
```

```
True
```

```
>>> problem7('adz')
```

```
True
```

```
>>> problem7('bbcdd')
```

```
True
```

```
>>> problem7('bacd')
```

```
False
```

```
>>> problem7('ra')
```

```
False
```

```
>>> problem7('abcdefzg')
```

```
False
```

## Problem 8 - Unique characters [17 pts]

Write a program that will find **if** the characters in a string are **all unique**. Meaning, there are no duplicate characters.

- inputs :  $a = \{ x: x \in \text{printable characters except whitespaces and } \text{len}(x) \in [1:200] \}$
- output : boolean

```
>>> problem8('abcd')
```

```
True
```

```
>>> problem8('asb@!@dz')
```

```
False
```

```
>>> problem8('bbcdd')
```

```
False
```

```
>>> problem8('abacd')
```

```
False
```

```
>>> problem8('r#aczxb')
```

```
True
```

```
>>> problem8('abgcdefzg')
```

```
False
```