Robotique CRC Robotics

CRC

# PROGRAMMING COMPETITION

KRYPTIK

## PRELIMINARY PROBLEM 1

## A FEW NOTES

- The complete rules are in section 4 of the rulebook.
- You have until **Sunday, December 3, 11:59 pm** to submit your code
- Feel free to use the programming forum on the CRC discord to ask questions and discuss the problem with other teams. That's what it's there for!
- **We are giving you quick and easy-to-use template files for your code and the tests. Please use them.**

## USING THE TEMPLATE FILE

- Basically, the template tests calls the function to test and takes the output and allows you to quickly check if it did what was expected or not. **All your code (except additional functions and classes, which should go right above it in the file) should be written in the function of the part of the problem you are solving.**
- Points given in the document are indications of how difficult the section is and how many points you will get if you complete it. This preliminary problem is going to be 2% of the main challenge towards the global score of the programming competition and for more points related information consult the rulebook.

## STRUCTURE

Every problem contains a small introduction like this about the basics of the problem and what is required to solve it. Points distribution is also given here for the preliminary problems.

### Input and output specification:

In these two sections, we specify what the inputs will be and what form they will take, and we also say what outputs are required for the code to produce and in what format they shall be.

### Sample input and output:

In these two sections, you will find a sample input (that often has multiple entries itself) in the sample input and what your program should produce for such an input in the sample output.

### Explanation of the first output:

Sometimes, the problem might still be hard to understand after those sections, which is why there will also be a usually brief explanation of the logic that was used to reach the first output from the first input.

# It’s cryptic, isn’t it?

## Part 1: Find the secret code (4 points)

You have found an encoded document in your computer and you really want to decipher it. You only know that the key to decrypt it is located inside the document. You will need to find it in order to eventually decrypt it. To help you find the key, a hint is that the key is a number sandwiched between two “|” (124 in ascii base 10 or 7C in base HEX) on both sides of the number. This number is the key you are looking for!

### Input specification:

You will receive as an input an encoded message with words that are complete gibberish. In this message will be the key number between the characters “|”.

### Output specification:

For the output, you will need to return the key found in the text, which is the number found between the characters “|”.

### Sample input:

```
ti kzk mab cvm kwuxmbqbqwv bzma|18| ncv

Zcfsa wdgia rczcf gwh oash, qcbgsqhshif orwdwgqwbu szwh. Qfog qifgig dzoqsfoh hcfhcf bsq toqwzwgwg. Bizzoa jsz sfoh zsc. Gigdsbrwggs bcb bwgz sbwa. Aosqsbog oq sfoh sush sfoh gcrozsg hsadig. Bizzo toqwzwgw. Dszzsbhsgeis toqwzwgwg|12| hwbqwribh bibq, dfshwia hwbqwribh rczcf jsbsbohwg wb. Bizzoa woqizwg oqqiagob zsc bcb sttwqwhif. Aosqsbog crwc ofqi, hfwghweis.
```

### Sample output:

```
18

12
```

### Explanation of the first output:

At the end of the first input we have the sequence “|18|”. This sequence is the key we are looking for. We only have to extract the number and return it.

## Part 2: Decrypting Caesar (8 points)

Once you have retrieved the key you can decrypt the file! The file is encoded using Caesar cipher. Caesar cipher is an encryption algorithm that works by shifting letters. We need to shift the letters from the alphabet by a specific number given by a key. If we have the key 3, a becomes d, b becomes e, c becomes f and the same applies until y becomes b and z becomes c. You will need to shift all the letters and only the letters in the message. No other special character in that message should be shifted.

### Input specification:

You will receive an encrypted message containing gibberish and a decryption key. The decryption key will be a number between -26 and 26.

### Output specification:

You will need to shift the letters in the message and return the decrypted message.

### Sample input:

```
ti kzk mab cvm kwuxmbqbqwv bzma ncv!
18

Zcfsa wdgia rczcf gwh oash, qcbgsqhshif orwdwgqwbu szwh. Qfog qifgig
dzoqsfoh hcfhcf bsq toqwzwgwg. Bizzoa jsz sfoh zsc. Gigdsbrwggs bcb
bwgz sbwa. Aosqsbog oq sfoh sush sfoh gcrozsg hsadig. Bizzo toqwzwgw.
Dszzsbhsgeis toqwzwgwg hwbqwribh bibq, dfshwia hwbqwribh rczcf
jsbsbohwg wb. Bizzoa woqizwg oqqiagob zsc bcb sttwqwhif. Aosqsbog
crwc ofqi, hfwghweis.
12
```

### Sample output:

```
la crc est une competition tres fun!

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras cursus
placerat tortor nec facilisis. Nullam vel erat leo. Suspendisse non
nisl enim. Maecenas ac erat eget erat sodales tempus. Nulla facilisi.
Pellentesque facilisis tincidunt nunc, pretium tincidunt dolor
venenatis in. Nullam iaculis accumsan leo non efficitur. Maecenas
odio arcu, tristique.
```

### Explanation of the first output:

In the first output, we have the decryption key that is 18. We will now take each letter and shift it by 18 places in the alphabet. Once every letter is shifted, we will have obtained the initial message. Please note that the punctuation hasn't been modified by the decryption.

## Part 3: Lost in a sea of words (8 points)

The decrypted text in the first part maybe seems a bit random and meaningless. The real message is hidden in the sea of useless words. To find the code you will have to find the 5 most common words of the text and order them from the most frequent to the least frequent. Those five words are the message you are looking for!

### Input specification:

You will receive a message with words of different frequencies.

### Output specification:

As an output, you will need to return the 5 most common words in order starting with the most frequent.

### Sample input:

```
chien le chat maison voiture message arbre secret soleil fleurs
musique rive montagne est ordinateur plage le cell lune livre mer
secret ceci restaurant amis le voyage chanson bateau hiver automne
neige amour sport bonheur le art danse message avion jardin livre
parc est cadeau table chapeau porte lunettes secret chemin message
commande fort pont or argent fourchette message couteau tasse montre
chapeau guitare le piano balance le nuage parapluie est balle
chocolat ceci message chaussettes radio silence plume pinceau secret
tableau nuage le bougie chemin miroir porte montre papier le lampe
sac message bonjour le sable
```

```
chocolat la pluie est la plage est pomme la sapin la musique fun
vraiment ordinateur prog prog cauchemar la montagne bonjour arbre
prog amour livre vraiment prog couleur danse guitare voyage chanson
est horizon fleur fun prog livre lune ballon vraiment ami prince la
plume prog vent porte la table aventure la neige est chaussure
histoire bateau
```

### Sample output:

```
le message secret est ceci
```

```
la prog est vraiment fun
```

### Explanation of the first output:

We look at the frequency of every word of the text and the one that is the most frequent is “le”. The next most common is “message” followed by “secret est ceci”. All of those words combined give us the message: “le message secret est ceci”. (meaning this is the secret message).

## Part 4: Word cascade (12 points)

To print the message in a more sophisticated manner we will organize the message in a beautiful cascade of words. We will take the input of 5 words and we are going to set them up in a cascade. For every word, each of its letters are going to be one row down and one space to the right. Every word will be separated by a space. If a word is shorter than the other ones, the missing letters have to be replaced by spaces.

### Input specification:

As an input, you will receive a string containing 5 words separated by spaces.

### Output specification:

As an output, you will need to return a list of strings containing all the levels of your cascade.

### Sample input:

```
voici un petit texte test

la prog est vraiment fun
```

### Sample output:

```
[
"v u p t t",
" o n e e e",
"  i   t x s",
"   c   i t t",
"    i   t e  "]
[
"l p e v f",
" a r s r u",
"    o t a n",
"     g   i  ",
"          m  ",
"           e  ",
"            n  ",
"             t  "]
```

### Explanation of the first output:

We take the letters from each word and shift it once while going down to create a cascade.

```
v u p t t
 o n e e e
  i   t x s
   c   i t t
    i   t e
```

## Part 5: Can you do better? (8 points)

You will need to create your own interesting formatting such as in **section 4** using 5 random words. To make sure everything works you will need to write your own test cases using pytest. To correct this part, we will look at both your code and your tests, and make sure it is properly working. Impress us with your best formatting!

### Input specification:

As an input, you will receive a string containing 5 words separated by spaces.

### Output specification:

Dealer’s choice!

### Sample input:

```
voici un petit texte test
```

### Sample output:

```
???
```

### Explanation of the first output:

Be creative!