ASSIGNMENT 2

COMP202, Winter 2021

Due: Friday, March 12^{th}, 11:59pm

Please read the entire PDF before starting. You must do this assignment individually.

Question 1:	10 points
Question 2:	35 points
Question 3:	55 points
	100 points total

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment, in some cases through automated tests. While these tests will never be used to determine your entire grade, they speed up the process significantly, which allows the TAs to provide better feedback and not waste time on administrative details. Plus, if the TA is in a good mood while he or she is grading, then that increases the chance of them giving out partial marks. :)

Up to 30% can be removed for bad indentation of your code as well as omitting comments, or poor coding structure.

To get full marks, you must:

- Follow all directions below.
  - In particular, make sure that all file names and function names are **spelled exactly** as described in this document. Otherwise, a 50% penalty will be applied.
- Make sure that your code runs.
  - Code with errors will receive a very low mark.
- Write your name and student ID in a comment at the top of all `.py` files you hand in.
- Name your variables appropriately.
  - The purpose of each variable should be obvious from the name.
- Comment your work.
  - A comment every line is not needed, but there should be enough comments to fully understand your program.
- Avoid writing repetitive code, but rather call helper functions! You are welcome to add additional functions if you think it can increase the readability of your code.
- Lines of code should NOT require the TA to scroll horizontally to read the whole thing. Vertical spacing is also important when writing code. Separate each block of code (also within a function) with an empty line.

Part 1 (0 points): Warm-up

Do **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then we suggest that you consult the TAs during their office hours; they can help you and work with you through the warm-up questions. You are responsible for knowing all of the material in these questions.*

Warm-up Question 1 (0 points)

Write a function `swap` which takes as input two int values `x` and `y`. Your function should do 3 things:

1. Print the value of `x` and `y`
2. Swap the values of the variables `x` and `y`, so that whatever was in `x` is now in `y` and whatever was in `y` is now in `x`
3. Print the value of `x` and `y` again.

For example, if your function is called as follows: `swap(3,4)` the effect of calling your method should be the following printing

```
inside swap: x is:3 y is:4
inside swap: x is:4 y is:3
```

Warm-up Question 2 (0 points)

Consider the program you have just written. Create two global integer variables in the main body of your program. Call them `x` and `y`. Assign values to them and call the swap function you wrote in the previous part using `x` and `y` as input parameters.

After calling the `swap()` function —inside the main body— print the values of `x` and `y`. Are they different than before? Why or why not?

Warm-up Question 3 (0 points)

Create a function called `counting` that takes as input a positive integer and counts up to that number. For example:

```
>>> counting(10)
Counting up to 10: 1 2 3 4 5 6 7 8 9 10
```

Warm-up Question 4 (0 points)

Modify the last function by adding an additional input that represents the step size by which the function should be counting. For example:

```
>>> counting(25, 3)
Counting up to 25 with a step size of 3: 1 4 7 10 13 16 19 22 25
```

Warm-up Question 5 (0 points)

Write a function `replace_all` which takes as input a string and two characters. If the second and third input string do not contain exactly one character the function should raise a `ValueError`. Otherwise, the function returns the string composed by the same characters of the given string where all occurrences of the first given character are replaced by the second given character. For example, `replace_all("squirrel", "r" , "s")` returns the string `"squissel"`, while `replace_all("squirrel", "t", "a")` returns the string `"squirrel"`. Do not use the method `replace` to do this.

Warm-up Question 6 (0 points)

Write a module with the following global variables:

```
lower_alpha = "abcdefghijklmnopqrstuvwxyz"
upper_alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

In this module write a function `make_lower` which takes a string as input and returns a string containing the same characters as the input string, but all in lower case. For example, `make_lower("AppLE")` returns the string `"apple"`. Do not use the method `lower` to do this. Hint: note that characters from the English alphabet appear in the same position in the two global variables.

Warm-up Question 7 (0 points)

Create a function called `generate_random_list` which takes an input an integer `n` and returns a list containing `n` random integers between 0 and 100 (both included). Use `random` to do this.

Warm-up Question 8 (0 points)

Write a function `sum_numbers` which takes as input a list of integers and returns the sum of the numbers in the list. Do not use the built-in function `sum` to do this.

Part 2

The questions in this part of the assignment will be graded.
The main learning objectives for this assignment are:

- Correctly define and use simple functions.
- Solidify your understanding of the difference between `return` and `print`.
- Generate and use random numbers inside a program.
- Understand how to test functions that contain randomness.
- Correctly use loops and understand how to choose between a while and a for loop.
- Solidify your understanding of how to work with strings: how to check for membership, how to access characters, how to build a string with accumulator patterns.
- Begin to use very simple lists.
- Create a program with more than one module.
- Learn how to use functions you have created in a different module.

Note that the assignment is designed for you to be practicing what you have learned in the videos up to and including Week 8.2. For this reason, you are NOT allowed to use anything seen after Week 8.2 or not seen in class at all. You will be heavily penalized if you do so.

For full marks on the following three questions, in addition to the points listed on page 1, make sure to add the appropriate documentation string (docstring) to *all* the functions you write. The docstring must contain the following:

- The type contract of the function.
- A description of what the function is expected to do.
- At least 3 examples of calls to the function (except when the function has only one possible output, in which case you can provide only one example). You are allowed to use *at most* one example per function from this pdf.

Examples

For each question, we provide several **examples** of how your code should behave. All examples are given as if you were to call the functions from the shell.

When you upload your code to codePost, some of these examples will be run automatically to check that your code outputs the same as given in the example. However, **it is your responsibility to make sure your code/functions work for any inputs, not just the ones shown in the examples.** When the time comes to grade your assignment, we will run additional, private tests that may use inputs not seen in the examples.

Furthermore, please note that your code files for this question and all others **should not contain any function calls in the main body of the program** (i.e., outside of any functions). Code that does not conform in this manner will automatically fail the tests on codePost and **be heavily penalized**. It is OK to place function calls in the main body of your code for testing purposes, but if you do so, make certain that you remove them before submitting. Please review what you have learned in video 5.2 if you'd like to add code to your modules which executes only when you run your files.

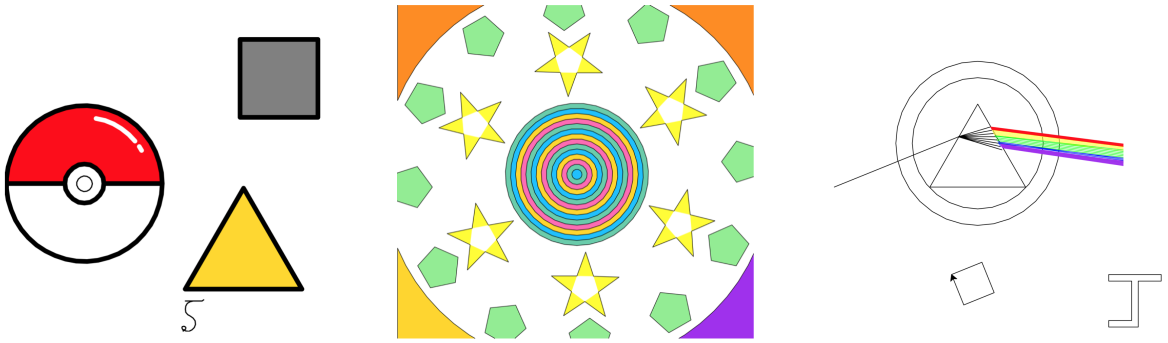
Safe Assumptions

For all questions in this assignment, you can safely assume that the **type** of the inputs (both to the functions and those provided to the program by the user) will always be correct. For example, if a function takes as input a list, you can assume that a list will always be provided. At times you will be required to do some

input validation, but this requirement will always be clearly stated. Otherwise, your functions should work with any possible input that respect the function's description. For example, if the description says that the function takes as input a positive integer, then it should work with all integers greater than 0. If it mentions an integer, then it should work for *any* integer. Make sure to test your functions for edge cases!

Code Repetition

One of the main principles of software development is DRY: Don't Repeat Yourself. One of the main ways we can avoid repeating ourselves in code is by writing functions, then calling the functions when necessary, instead of repeating the code contained within them. Please pay careful attention in the questions of this assignment to not repeat yourself, and instead call previously-defined functions whenever appropriate. As always, you can also add your own helper functions if need be, with the intention of reducing code repetition as much as possible.



Question 1: Turtle Art (10 points)

This question asks you to write a function `my_artwork()` in a file called `artwork.py`. The function should draw a picture using the Turtle module. You are free to draw what you like, but your code/drawing must satisfy at least the following requirements:

- the drawing must include at least three shapes
- at least one shape must be drawn using a for loop
- at least one shape must be drawn using a function, with the function having at least two parameters that modify the shape being drawn in some way. (You cannot simply copy one of the functions we have written in our lectures - you must write your own function that is not similar in function to the ones written in class.)
- the drawing must include at least two different colors
- at least one random number must be used to create the drawing in some way
- everything must fit into the standard Turtle window (without having to make the window larger)
- the first letter of your first name must appear somewhere (you must sign your artwork!)
- there should be **no** calls to the `input()` function
- **As always, do not call any functions (including `turtle.Turtle()`) in the main body.**

Any submission meeting these requirements will obtain full marks, but you are encouraged to go beyond them. The most creative submissions (as judged by our TAs) will be shown in class. As St. Patrick's Day is right around the due date of the assignment, you are encouraged to take inspiration from that holiday. ☘ ☘ ☘

Note: Recall that you can import the `speed` function from turtle and then call `speed("fastest")` to speed up the drawing routines, so that you don't waste time when testing your code.

Also, a reminder to please only use the functions from the Turtle module that we have seen in class, or you will lose marks. There is one exception: you can use the `circle` function from Turtle module. `circle(r)` takes a radius `r` as argument and draws a circle of the given radius. You can also specify a second integer argument for the extent of the circle to draw, e.g., `circle(r, 90)`, which will draw only a quarter of a circle (90 degrees).

Some submissions from students of previous years (with their size scaled down to fit) can be found at the top of this page.

Question 2: Return of the COMP202COIN (35 points)

It is the year 3019 on the planet Orion. COMP202COIN has become the most popular currency on the planet. As much time has passed, the COMP202COIN that exist now are not like the ones you have seen previously. These COMP202COIN are represented in a special base known as 'base 202', which is similar to base 8 (octal). Base 8 uses the digits 0 through 7 to represent quantities. Base 202 is similar to base 8, as it uses only 8 characters, but instead of `0, 1, 2, 3, 4, 5, 6,` and `7`, it uses the following: `0, C, 2, O, M, P, I,` and `N`. A `0` (zero) and a `2` (two) are the same in both bases, but a `1, 3, 4, 5, 6,` or `7` of a base 8 string do not exist in base 202 and are instead represented by a `C, O, M, P, I,` and `N`, respectively.

Another important note about amounts of COMP202COIN is that they are always represented using 10 characters. The first two characters are always `0c` or `0C`. The next eight characters represent the amount of COMP202COIN in base 202, with `0`'s padding the amount in front if it does not need the full eight characters. For example, a solitary COMP202COIN would be represented as the string `0c0000000C` (as a `1` in base 202 is `C`). Note that the letters are not case-sensitive, so a `c` or `C`, `n` or `N`, etc., can be used interchangeably.

You may assume for the purposes of this question that we will never deal with an amount greater than 8 characters in base 202 (i.e., the maximum value would be `0cNNNNNNNN`).

Everything has been going great on the planet and major banks are now starting to use COMP202COIN as their main currency. But, bad news. Attack ships were spotted off the shoulder of Orion two days ago. Yesterday, they detonated an electromagnetic charge in the atmosphere, which has scrambled communications systems. Customers logging into their banks today have discovered their monthly statements of COMP202COIN transactions have been totally garbled. The banks are asking for help to get their customer transactions in order. Can you help restore order?

We will approach this problem by writing code in a file called `coins.py`. First, define two global variables at the top of your file:

- `BASE8_CHARS`: a string containing the characters of base 8: `0, 1, 2, 3, 4, 5, 6,` and `7`.
- `BASE202_CHARS`: a string containing the characters of base 202: `0, C, 2, O, M, P, I,` and `N`.

Use them as appropriate in your code.

Then, define the following functions:

- `base10_to_202(amt_in_base10)`: a function that takes an integer representing an amount in base 10, and returns the corresponding amount in base 202 as a 10-character string (as described above). (If it helps, you may use the `oct()` built-in function which takes an integer in base 10 as argument and returns a string representing the integer in base 8 and beginning with the characters `0o` to represent it is an octal string.)
  EXAMPLE:

```
>>> base10_to_202(202)
'0c00000OC2'
```

- `base202_to_10(amt_in_base202)`: a function that takes a string representing an amount in base 202 (as described above) as argument, and returns the corresponding amount in base 10 as an integer.
  EXAMPLE:

```
>>> base202_to_10('0c00000OC2')
202
```

When clients view their banking information on OrioNet, all their transactions are jumbled up into a single paragraph. We have to write some functions to identify particular transactions/amounts of money in the paragraph.

- `is_base202(text)`: takes a string as argument, and returns `True` if the string is a valid 10-character COMP202COIN string, and `False` otherwise. Note that the string could be any length and contain any characters.
  Examples:

```
>>> is_base202('1cCOMPCOIN')
False
>>> is_base202('0c0C2OMPIN')
True
>>> is_base202('<!doctype html><html itemtype="tp://schema.onet/BankPage">')
False
```

- `get_nth_base202_amount(text, n)`: takes a string that could be of any length and contain any characters, and a non-negative integer `n`. The function should return the `n`'th 10-character COMP202COIN substring contained within the string. We will start counting at zero, so if `n=0`, the first 10-character COMP202COIN substring contained within the string should be returned. If there is no `n`'th COMP202COIN substring, the function should return the empty string instead.

  Note: A 10-character COMP202COIN substring cannot be part of another 10-character COMP202COIN substring. For example, `` `0c0cOIN0COIN' `` contains only one 10-character COMP202COIN, starting at index 0 and ending at index 9. Although there is another 10-character COMP202COIN from index 2 to 11, it should not be recognized by the function because indices 2 through 9 are already part of the first COMP202COIN string, and so cannot be 're-used.'
  Examples:

```
>>> get_nth_base202_amount("BANKING TRANSACTIONS....PLANET ORION......FEBRUARY \
15, 3019.......0cCCMMPP22........FEBRUARY 16, 3019..........0cOCOCOCOC.........\
FEBRUARY 17, 3019..........0C24242412", 1)
'0cOCOCOCOC'
>>> get_nth_base202_amount("BANKING TRANSACTIONS....PLANET ORION......FEBRUARY \
15, 3019.......0cCCMMPP22........FEBRUARY 16, 3019..........0cOCOCOCOC.........\
FEBRUARY 17, 3019..........0C24242412", 2)
''
```

- `get_total_dollar_amount(text)`: takes a string that could be of any length and contain any characters, and returns the total dollar amount (in base 10) of all COMP202COIN present in the string, if any.
  Example:

```
>>> get_total_dollar_amount("BANKING TRANSACTIONS....PLANET ORION......FEBRUARY\
 15, 3019.......0cCCMMPP22........FEBRUARY 16, 3019..........0cOCOCOCOC........\
FEBRUARY 17, 3019..........0C24242412")
9167275
```

The communication problem has caused a problem for the banks, who have lost the amounts of how much COMP202COIN they have to give out to clients. They are therefore imposing a maximum limit on the amount of COMP202COIN in any client's holdings. If a client's total holdings of COMP202COIN is greater than the limit, then the accounts of that client will all be reduced by a percentage such that the total of their holdings sums to the limit. We will write this task in the following function:

- `reduce_amounts(text, limit)`: takes a string that could be of any length and contain any characters, and a non-negative integer `limit`. If the string contains any COMP202COIN amounts and their total dollar amount is less than or equal to the `limit`, then the function should return the same

string unchanged. Otherwise, it should calculate the difference between the total dollar amount and the limit, then find the percent decrease (by dividing the difference by the total dollar amount). It should then go through the string, and reduce each COMP202COIN amount by the given percentage (rounding down). Note that as the percentage relates to dollar amounts and not COMP202COIN amounts, each time a COMP202COIN amount is found in the string, it must first be converted to base 10, then reduced by the percentage and then converted back to base 202. The updated string should then be returned. Note that any characters not part of a COMP202COIN amount should remain unchanged in the returned string.

EXAMPLES:

```
>>> reduce_amounts('0c000000C2', 5)
'0c0000000P'
>>> reduce_amounts("0cCCMMPP22     0cOCOCOCOC", 9000000)
'0cCCOCMCI0     0cO0NOPNCN'
```

Question 3: Farkle (55 points)

Farkle is a dice game. To play, all that is required is six 6-sided dice. Normally Farkle is played with 2 to 8 players. Each player in turn rolls all six dice and checks to see if they have rolled any scoring dice or combinations. (See Scoring below.) Any dice that score may be set aside and then the player may choose to roll all the remaining dice. The player must set aside at least one scoring die of their choice if possible but is not required to set aside all scoring dice.

For example, if a player rolled 1-2-2-5-5-6 on their turn, they could set aside the 1 and the two 5's for scoring, or they could choose to set aside only the 1. Any scoring dice that are not set aside may be rerolled along with the non-scoring dice.

If all six dice have been set aside for scoring (known as having "hot dice"), the player can choose to roll all six dice again and continue adding to their accumulated score, or they can bank their points, end their turn, and pass the dice to the next player.

If a player scores no points on a roll, this is known as a Farkle. The player may continue to roll any dice that have not been previously set aside for scoring, but all of their points gained so far that turn are lost.

A player's turn continues until either they decide to stop (at which point they then score their accumulated points) or until they have no more dice to throw. By deciding to keep re-rolling, the player may obtain a higher score, but they are also taking the risk to lose all their points so far if they Farkle.

At the end of a player's turn, any points they have scored are recorded and the dice are passed to the next player.

The goal of this question is to write several modules to implement a multi-player game of Farkle.

Scoring

This is the commonly used scoring rules for Farkle. Different variations are also used.

1	100 points
5	50 points
Three 1's	1,000 points
Three 2's	200 points
Three 3's	300 points
Three 4's	400 points
Three 5's	500 points
Three 6's	600 points
1-2-3-4-5-6	3000 points
3 Pairs	1500 points

Note that scoring combinations only count when made with a single throw. (Example: If a player rolls a 1 and sets it aside and then rolls two 1's on their next throw, they only score 300 points, not 1000.)

Sometimes a single roll will provide multiple ways to score. For example, a player rolling 1-2-4-5-5-5 could score one of the following:

- 100 points for the 1
- 150 points for the 1 and a 5
- 500 points for the three 5's
- 600 points for the 1 and the three 5's

It is up to the player to decide which dice to set aside for scoring.

Farkle Utility Functions

Let's start by creating a module named `farkle_utils` which contains several helper functions needed to implements the full program. Inside this module import the module `random`.

For full marks, all the following functions must be part of this module. **Be careful!**, functions that take lists as input should **not** modify them. To prevent this from happening you can create additional lists inside the function which are copies of the inputs and work with those instead.

- `single_dice_roll`: simulates the roll of one 6-sided dice. The function takes no inputs, and returns an integer between 1 and 6 (both included). Please use `randint` to do so.

  For example:

  ```
  >>> random.seed(1000)
  >>> single_dice_roll()
  4

  >>> random.seed(3)
  >>> single_dice_roll()
  2
  >>> single_dice_roll()
  5
  ```

- `dice_rolls`: given a positive integer `n` as input, it returns a list containing the numbers representing `n` independent 6-sided dice rolls.

  For example:

  ```
  >>> random.seed(1)
  >>> dice_rolls(6)
  [2, 5, 1, 3, 1, 4]

  >>> random.seed(3)
  >>> dice_rolls(2)
  [2, 5]
  ```

- `contains_repetitions`: given a list of integers, an integer `n`, and a second positive integer `m`, the function returns `True` if `n` appears in the list at least `m` times, `False` otherwise.

  For example:

  ```
  >>> contains_repetitions([1, 2, 1], 1, 2)
  True
  >>> contains_repetitions([1, 2, 1, 1], 2, 5)
  False
  >>> contains_repetitions([1, 2, 1, 1, 2, 1], 1, 3)
  True
  ```

- `pick_random_element`: given a list of integers, it returns a random element from the list. If the list is empty, the function should return `None`.

For example:

```
>>> random.seed(5)
>>> pick_random_element([15, -5, 3])
3

>>> pick_random_element([1, 2, 3, 4, 5, 6, 7, 8, 9])
5
```

- `contains_all`: given a list of integers, it returns `True` if the list contains all unique consecutive positive integers starting from 1. The order in which the elements appear in the list is not important.

  For example:

  ```
  >>> contains_all([1, 2, 3, 4, 5])
  True
  >>> contains_all([3, 1, 2])
  True
  >>> contains_all([4, 5])
  False
  >>> contains_all([1, 1, 1])
  False
  ```

- `count_num_of_pairs`: given a list of integers, it returns the number of pairs in the list. Note that if a number appears 4 times in the list, this should count as two pairs, if it appears 6 times it should count as 3 pairs, and so on. The order in which the elements appear in the list is not important.

  For example:

  ```
  >>> count_num_of_pairs([1, 1, 2, 2])
  2
  >>> count_num_of_pairs([1, 2, 1, 2, 1])
  2
  >>> count_num_of_pairs([1, 1, 1, 2, 2, 1, 1])
  3
  ```

- `is_included`: given two lists of integers as input, it returns `True` if the second one is a subset of the first one, `False` otherwise. Note that once again, the order in which the elements appear in both list is not important.

  For example:

  ```
  >>> n = [1, 2, 4, 5, 5, 5]
  >>> m1 = [1, 5, 5, 5]
  >>> is_included(n, m1)
  True
  >>> n # it remains the same after the function execution
  [1, 2, 4, 5, 5, 5]
  >>> m1 # it remains the same after the function execution
  [1, 5, 5, 5]

  >>> m2 = [1, 1]
  >>> is_included(n, m2)
  False
  ```

```
>>> n # it remains the same after the function execution
[1, 2, 4, 5, 5, 5]
>>> m2 # it remains the same after the function execution
[1, 1]

>>> is_included([2, 5], [5])
True

>>> is_included([6, 4, 4, 2, 6, 3], [4, 4, 6, 6])
True
```

- `get_difference`: given two lists of integers as input, it returns a list which, if added to the second one, would result in a list containing the same elements as the first one. If the second list is not a subset of the first one, then this function should return an empty list. Note that once again, the order in which the elements appear in both list is not important.

  For example:

  ```
  >>> get_difference([1, 2, 3, 4, 5], [2, 4])
  [1, 3, 5]
  >>> get_difference([1, 2, 1, 3], [1, 3])
  [2, 1]
  >>> get_difference([1, 2, 1], [2, 2])
  []
  ```

A game of Farkle

Let's now create a module named `farkle`. This module will contain the functions that allow us to execute a game of Farkle between multiple players.

Inside this module, import `random` and `farkle_utils`, and add the following global variables:

```
SINGLE_ONE = 100
SINGLE_FIVE = 50
TRIPLET_MULTIPLIER = 100
STRAIGHT = 3000
THREE_PAIRS = 1500
```

Use the variables whenever appropriate in your code. For full marks, all the following functions must be part of this module:

- `compute_score`: given a list of integers between 1 and 6 (both included) as input representing the dice rolls selected by a player, it computes and returns the points scored. This function should use the global variables above and should refer to the scoring rules introduced at the beginning of this question. Please note that when scoring a triplet of 1's, the score should be multiplied by 10 on top of using the `TRIPLET_MULTIPLIER` defined above. Note that if not all the rolls selected can be used when computing the score, then the total score should be 0. **The function should output the highest possible score that can be made by using all rolls. For instance, assuming the global variables are initialized as indicated above `[2, 2, 2, 2, 2, 2]` would be considered a three pairs scoring 1500 points, while `[1, 1, 1, 1, 1, 1]` would be considered two triplets of 1 scoring 2000 points. Note that, you can always assume that triplets score more than counting a single value three times. That is, it will never be the case that, for example, `[1, 1, 1]` scores more as three ones compare to a triplet.**

For example:

```
>>> compute_score([1])
100
>>> compute_score([1, 5])
150
>>> compute_score([1, 1, 1])
1000
>>> compute_score([1, 1, 2, 2, 3, 3])
1500
>>> compute_score([1, 2, 3, 4, 5, 6]) == STRAIGHT
True
>>> compute_score([5, 1, 5, 5]) == SINGLE_ONE + 5 * TRIPLET_MULTIPLIER
True
>>> compute_score([1, 1, 2])
0
```

- `get_winners`: given a list of positive integers representing the scores of the players, and a positive integer representing the score to reach in order to win the game, it returns a list of players with the highest score which reached or surpassed the winning score. Note that we represent a player with an integer. This integer is the position of the player's score in the list when we start counting from 1. For instance, given the scores [500, 100, 5], we know that Player 1 scored 500 points, Player 2 100, and Player 3 only 5 points. If there are ties, the player numbers should appear in increasing order.

  For example:

  ```
  >>> get_winners([500, 100, 5], 10000)
  []
  >>> get_winners([500, 10000, 50], 10000)
  [2]
  >>> get_winners([11, 3, 9, 11], 5)
  [1, 4]
  ```

- `play_one_turn`: given a positive integer representing a player, it allows that player to play their turn. The function then returns the score of this player after they end their turn. For this function you do not need to write the examples in the docstring. You should still write the type contract and the description.

  Here is how a turn of Farkle for one player works:

  1. The player starts with 6 dice to roll.
  2. The function asks the user whether they would like to roll or pass.
  3. If the player chooses to roll, then the dice are rolled and the result is displayed on one line in the shell. Otherwise (if the player chose anything else but 'roll'), the function returns the score they have accumulated up to now. **Note that the function should not be case sensitive**.
  4. The player is then asked to select the scoring dice to set aside. They do so by entering as input a sequence of integers separated by a space.
  5. If the player selects dice that have not been rolled, the function will keep asking for a new selection until the player enters a valid one.
  6. The function then computes the current score of the player in this turn and the remaining dice available to roll.

- Note that if, with the selection made, the player scores a 0, the player has "farkled" and all points for this turn are lost. The player may continue to roll any dice that have not been previously set aside for scoring, including those that scored a 0.
- If all remaining dice were set aside for scoring, then the player have "hot dice" and may continue their turn with a new throw of all six dice, adding to the score they have already accumulated.

7. The function displays the current accumulated score as well as the remaining number of dice the player has. Assuming that the player has at least one dice to roll, the function asks them whether they'd like to roll or pass and the same process from step 3 is repeated.

Please note that the messages displayed by the function do not matter, and you are encouraged to personalize them. What matters is that all the information listed above appears, and is requested in the order specified. For example, the player should be asked whether they'd like to 'roll', a sequence of integers representing the rolls should be displayed, etc.

You can find below some examples of executing the function. Note that the inputs provided from the user appear in blue. The last number (the one that appears at the end of the execution) represents the value returned (not displayed!!) by the function.

Example 1:

```
>>> random.seed(0)
>>> play_one_turn(1)
Player 1 it's your turn!

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  4, 4, 1, 3, 5, 4
Please select the dice you'd like to set aside for scoring: 1 4 4 4 5
Your current score in this turn is: 550
You have 1 dice to keep playing.

What would you like to do? (roll/pass): pass
550
```

Example 2:

```
>>> random.seed(5)
>>> play_one_turn(3)
Player 3 it's your turn!

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  5, 3, 6, 3, 6, 6
Please select the dice you'd like to set aside for scoring: 3 3 3 5
You do not have these dice. Select again: 6 6 6
Your current score in this turn is: 600
You have 3 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 3 dice:  6, 5, 1
Please select the dice you'd like to set aside for scoring: 1 5
Your current score in this turn is: 750
You have 1 dice to keep playing.
```

```
What would you like to do? (roll/pass): pass
750
```

Example 3:

```
>>> random.seed(100)
>>> play_one_turn(2)
Player 2 it's your turn!

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  2, 4, 4, 2, 6, 4
Please select the dice you'd like to set aside for scoring: 4 4 4
Your current score in this turn is: 400
You have 3 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 3 dice:  6, 3, 4
Please select the dice you'd like to set aside for scoring: 6
FARKLE! All the points accumulated up to now are lost.
Your current score in this turn is: 0
You have 3 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 3 dice:  5, 1, 5
Please select the dice you'd like to set aside for scoring: 1 5 5
HOT DICE! You are on a roll. You get all six dice back.
Your current score in this turn is: 200
You have 6 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  1, 1, 6, 4, 3, 1
Please select the dice you'd like to set aside for scoring: 1 1 1
Your current score in this turn is: 1200
You have 3 dice to keep playing.

What would you like to do? (roll/pass): pass
1200
```

Example 4:

```
>>> play_one_turn(45)
Player 45 it's your turn!

What would you like to do? (roll/pass): not today!
0
```

Example 5:

```
>>> random.seed(17)
>>> play_one_turn(7)
Player 7 it's your turn!

What would you like to do? (roll/pass): ROLL
```

```
Here's the result of rolling your 6 dice:  5, 4, 3, 3, 3, 2
Please select the dice you'd like to set aside for scoring: 1
You do not have these dice. Select again: 1 2 3 4 5 6
You do not have these dice. Select again: 3 3 3
Your current score in this turn is: 300
You have 3 dice to keep playing.

What would you like to do? (roll/pass): ROLL
Here's the result of rolling your 3 dice:  6, 6, 5
Please select the dice you'd like to set aside for scoring: 5
Your current score in this turn is: 350
You have 2 dice to keep playing.

What would you like to do? (roll/pass): DONE
350
```

- `play_farkle`: given no inputs, this functions executes a game of Farkle. For this function you do not need to write the examples in the docstring. You should still write the type contract and the description.

  Here is how a game of Farkle should work:

  – The function asks how many players would like to play? If the input is not an integer between 2 and 8 (both included), the function asks for a new integer until the input provided is valid.

  – The function then asks what should be winning score of this game. If the input is not a positive integer, the function asks for a new integer until the input is valid.

  – Then rounds of Farkle are played until a player can be awarded the winner! A round of Farkle consists in all the players taking their turn. Note that once a player has achieved a winning point total, the round keeps being executed so that all the other players can take their turn and have a chance to score enough points to surpass that high-score.

  – After each round the scores the players have accumulated until then should be displayed. When doing that, the round number should also appear.

  – If at the end, more than one player has achieved the winning high-score, then the winner is selected at random.

  Please note that the messages displayed by the function do not matter, and you are encouraged to personalize them. What matters is that all the information listed above appear/are requested in the order specified. For example, the number of players should be asked (and settled on) before the winning score, etc.

  You can find below a some examples of executing the function. Note that the inputs provided from the user appear in light blue.

  Example 1:

```
>>> random.seed(0)
>>> play_farkle()
Welcome to COMP202_Farkle!

Please select the number of players (2-8): 2
Select the winning score for this game: 500

Player 1 is your turn!
```

```
What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  4, 4, 1, 3, 5, 4
Please select the dice you'd like to set aside for scoring: 1 5 4 4 4
Your current score in this turn is: 550
You have 1 dice to keep playing.

What would you like to do? (roll/pass): pass

Player 2 is your turn!

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  4, 3, 4, 3, 5, 2
Please select the dice you'd like to set aside for scoring: 5
Your current score in this turn is: 50
You have 5 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 5 dice:  5, 2, 3, 2, 1
Please select the dice you'd like to set aside for scoring: 1
Your current score in this turn is: 150
You have 4 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 4 dice:  5, 3, 5, 6
Please select the dice you'd like to set aside for scoring: 5 5
Your current score in this turn is: 250
You have 2 dice to keep playing.

What would you like to do? (roll/pass): pass

After round 1 the scores are as follows:
Player 1 : 550
Player 2 : 250

Thank you for playing! The winner of this game is: Player 1
```

Example 2:

```
>>> random.seed(100)
>>> play_farkle()
Welcome to COMP202_Farkle!

Please select the number of players (2-8): 5
Select the winning score for this game: 500

Player 1 is your turn!

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  2, 4, 4, 2, 6, 4
Please select the dice you'd like to set aside for scoring: 4 4 4
Your current score in this turn is: 400
You have 3 dice to keep playing.
```

```
What would you like to do? (roll/pass): roll
Here's the result of rolling your 3 dice:  6, 3, 4
Please select the dice you'd like to set aside for scoring: 6
FARKLE! All the points accumulated up to now are lost.
Your current score in this turn is: 0
You have 3 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 3 dice:  5, 1, 5
Please select the dice you'd like to set aside for scoring: 5 5 1
HOT DICE! You are on a roll. You get all six dice back.
Your current score in this turn is: 200
You have 6 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  1, 1, 6, 4, 3, 1
Please select the dice you'd like to set aside for scoring: 1 1 1
Your current score in this turn is: 1200
You have 3 dice to keep playing.

What would you like to do? (roll/pass): pass

Player 2 is your turn!

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  6, 6, 2, 3, 2, 3
Please select the dice you'd like to set aside for scoring: 2 3 2 3 6 6
HOT DICE! You are on a roll. You get all six dice back.
Your current score in this turn is: 1500
You have 6 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  2, 2, 2, 2, 3, 3
Please select the dice you'd like to set aside for scoring:  2 2 2
Your current score in this turn is: 1700
You have 3 dice to keep playing.

What would you like to do? (roll/pass): pass

Player 3 is your turn!

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  6, 4, 2, 4, 4, 5
Please select the dice you'd like to set aside for scoring: 4 4 4 5
Your current score in this turn is: 450
You have 2 dice to keep playing.

What would you like to do? (roll/pass): pass

Player 4 is your turn!

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  3, 4, 2, 6, 6, 1
Please select the dice you'd like to set aside for scoring: 1
```

```
Your current score in this turn is: 100
You have 5 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 5 dice:  2, 1, 5, 4, 2
Please select the dice you'd like to set aside for scoring: 5
Your current score in this turn is: 150
You have 4 dice to keep playing.

What would you like to do? (roll/pass): pass

Player 5 is your turn!

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  5, 2, 2, 2, 1, 6
Please select the dice you'd like to set aside for scoring: 5 1 2 2 2
Your current score in this turn is: 350
You have 1 dice to keep playing.

What would you like to do? (roll/pass): pass

After round 1 the scores are as follows:
Player 1 : 1200
Player 2 : 1700
Player 3 : 450
Player 4 : 150
Player 5 : 350

Thank you for playing! The winner of this game is: Player 2
```

Example 3:

```
>>> random.seed(123)
>>> play_farkle()
Welcome to COMP202_Farkle!

Please select the number of players (2-8): 2
Select the winning score for this game: 2000

Player 1 is your turn!

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  1, 3, 1, 4, 3, 1
Please select the dice you'd like to set aside for scoring: 1 1 1
Your current score in this turn is: 1000
You have 3 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 3 dice:  1, 4, 5
Please select the dice you'd like to set aside for scoring: 1 5
Your current score in this turn is: 1150
You have 1 dice to keep playing.

What would you like to do? (roll/pass): done
```

```
Player 2 is your turn!

What would you like to do? (roll/pass): ROLL
Here's the result of rolling your 6 dice:  5, 3, 3, 1, 2, 2
Please select the dice you'd like to set aside for scoring: 1
Your current score in this turn is: 100
You have 5 dice to keep playing.

What would you like to do? (roll/pass): ROLL
Here's the result of rolling your 5 dice:  3, 5, 3, 6, 2
Please select the dice you'd like to set aside for scoring: 5
Your current score in this turn is: 150
You have 4 dice to keep playing.

What would you like to do? (roll/pass): ROLL
Here's the result of rolling your 4 dice:  2, 1, 4, 1
Please select the dice you'd like to set aside for scoring: 1 1
Your current score in this turn is: 350
You have 2 dice to keep playing.

What would you like to do? (roll/pass): OK

After round 1 the scores are as follows:
Player 1 : 1150
Player 2 : 350


Player 1 is your turn!

What would you like to do? (roll/pass): roll
Here's the result of rolling your 6 dice:  5, 4, 1, 1, 3, 6
Please select the dice you'd like to set aside for scoring: 1 1
Your current score in this turn is: 200
You have 4 dice to keep playing.

What would you like to do? (roll/pass): roll
Here's the result of rolling your 4 dice:  4, 1, 1, 1
Please select the dice you'd like to set aside for scoring: 1 1 1
Your current score in this turn is: 1200
You have 1 dice to keep playing.

What would you like to do? (roll/pass): done for now

Player 2 is your turn!

What would you like to do? (roll/pass): ROLL
Here's the result of rolling your 6 dice:  6, 2, 2, 1, 3, 4
Please select the dice you'd like to set aside for scoring: 1
Your current score in this turn is: 100
You have 5 dice to keep playing.

What would you like to do? (roll/pass): ROLL
Here's the result of rolling your 5 dice:  5, 4, 3, 4, 1
```

```
Please select the dice you'd like to set aside for scoring: 1
Your current score in this turn is: 200
You have 4 dice to keep playing.

What would you like to do? (roll/pass): ROLL
Here's the result of rolling your 4 dice:  3, 3, 5, 4
Please select the dice you'd like to set aside for scoring: 3 3 3
You do not have these dice. Select again: 3 3 5
FARKLE! All the points accumulated up to now are lost.
Your current score in this turn is: 0
You have 4 dice to keep playing.

What would you like to do? (roll/pass): PASS

After round 2 the scores are as follows:
Player 1 : 2350
Player 2 : 350

Thank you for playing! The winner of this game is: Player 1
```

What To Submit

You must submit all your files on codePost (https://codepost.io/). The file you should submit are listed below. Any deviation from these requirements may lead to lost marks.

`artwork.py`
`coins.py`
`farkle_utils.py`
`farkle.py`
`README.txt` In this file, you can tell the TA about any issues you ran into doing this assignment. If you point out an error that you know occurs in your program, it may lead the TA to give you more partial credit.
Remember that this assignment like all others is an `individual` assignment and must represent the entirety of your own work. You are permitted to verbally discuss it with your peers, as long as no written notes are taken. If you do discuss it with anyone, please make note of those people in this `README.txt` file. If you didn't talk to anybody nor have anything you want to tell the TA, just say "nothing to report" in the file.