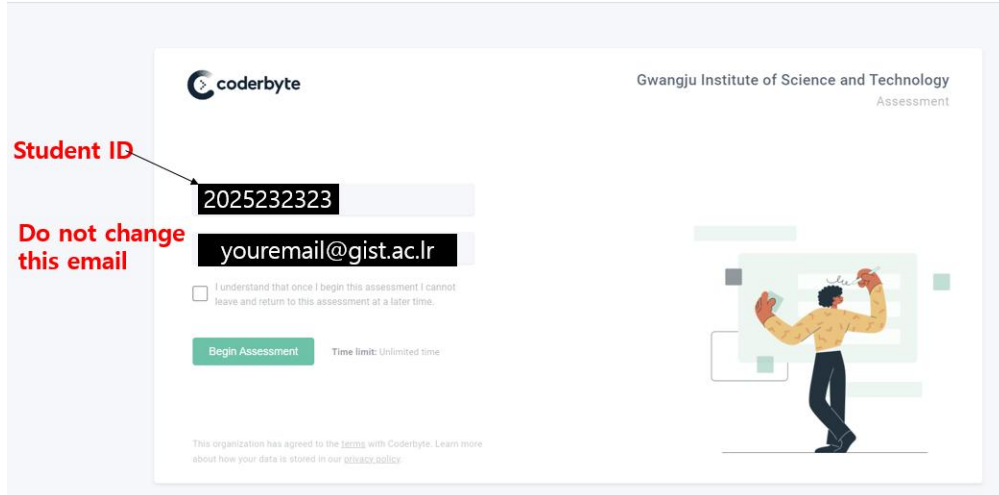


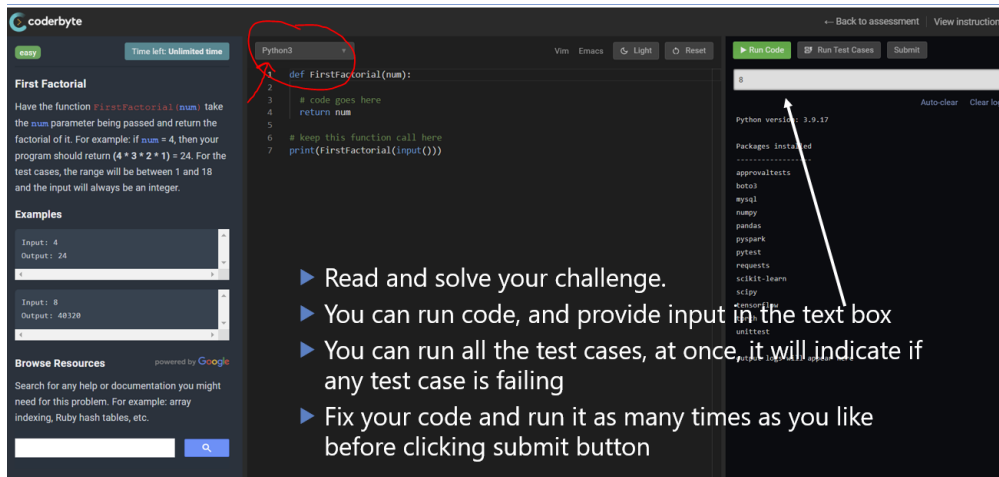
Algo Assignment 2: Recursion

General Instruction to submit the assignment

1. Open the link sent in email and enter your **Student ID** and **ZEUS email address**



2. You must change the programming language to **Python3** before starting to code. There is a drop down button above the coding window.



- ▶ Read and solve your challenge.
- ▶ You can run code, and provide input in the text box
- ▶ You can run all the test cases, at once, it will indicate if any test case is failing
- ▶ Fix your code and run it as many times as you like before clicking submit button

3. If you submit and try to resubmit etc., the system will not let you do that. (**once submitted you cannot update your code**)
4. Hence **solve this question beforehand and then enter your code in Coderbyte system**. This way you will save a lot of hassle. The questions will also be released in the pdf format. Make sure your code gives the right results for all the given test cases before you submit your code. You may ignore warning messages about cheating when you paste your code to Coderbyte.
5. You should **not use system libraries** while solving questions for at least first 3 assignments. Starting from DP assignments you can use **min max** and **sort** functions of python3.
6. Remember total points from the assignments are only **10%** of your grades. Hence these are mainly for practicing for mid-term and final-term coding tests.
7. The time complexity of your submissions will be tested only after 4th Assignment.

Q1. Remove Brackets

Have the function `RemoveBrackets(str)` take the `str` parameter being passed, which will contain only the characters "(" and ")", and determine the minimum number of brackets that need to be removed to create a string of correctly matched brackets. For example: if `str` is "(()))" then your program should return the number **1**. The answer could potentially be 0, and there will always be at least one set of matching brackets in the string.

Code frame

```
def RemoveBrackets(strParam):  
  
    # code goes here  
    return strParam  
  
# keep this function call here  
print(RemoveBrackets(input()))
```

Test cases

1. For input " (()) ", the correct output is **1**
 2. For input " (()) (((", the correct output is **3**
 3. For input " (((", the correct output is **2**
 4. For input " () ", the correct output is **0**
 5. For input ") (() ", the correct output is **2**
 6. For input " ((())) ", the correct output is **0**
 7. For input " () () () () () (((((", the correct output is **6**
 8. For input ") (() ", the correct output is **2**
 9. For input " ((())) () (()) ", the correct output is **1**
 10. For input ") (()) ", the correct output is **1**
-

Q2. Sort an Array using recursion

Sorting an array can be done using an iterative method. We have also learned divide and conquer methods of sorting an array using merge sort and quick sort. But in this question, you must sort an array using simple recursion. You should not care about the time complexity of this question. The goal is to teach you how you can think in terms of recursion to solve a given problem.

You must not use `min()`, `max()`, `sort()` functions of python. if needed you can write your own `min()` or `max()` function, again in recursive manner. You must not use merge sort or quick sort algorithm. In your code, you must not have any loops, it should be completely loop free.

Both the input and output are lists of integers, and they can be empty.

Code frame

```
def SortArrayusingrecursion(arr):  
  
    # code goes here  
    return arr  
  
# keep this function call here  
print(SortArrayusingrecursion(input()))
```

Test cases

1. For input `[8, 9, 10, 2, 4, 5]` the output was incorrect. The correct output is `[2, 4, 5, 8, 9, 10]`
2. For input `[2, 4, 5, 1, 11, 22, 45]` the output was incorrect. The correct output is `[1, 2, 4, 5, 11, 22, 45]`
3. For input `[4, 1, 2, 3, 4, 90, 0]` the output was incorrect. The correct output is `[0, 1, 2, 3, 4, 4, 90]`
4. For input `[1, 3, 5, 0, 9, 20, 40]` the output was incorrect. The correct output is `[0, 1, 3, 5, 9, 20, 40]`
5. For input `[1, 1, 1, 1, 0, 0, 0,]` the output was incorrect. The correct output is `[0, 0, 0, 1, 1, 1, 1]`
6. For input `[5, 3, 8, 5, 9, 1, 5]` the output was incorrect. The correct output is `[1, 3, 5, 5, 5, 8, 9]`

7. For input `[101, 91, 54, 11, 6, 1]` the output was incorrect. The correct output is `[1, 6, 11, 54, 91, 101]`

8. For input `[-5, 3, -2, 4, 0]` the output was incorrect. The correct output is `[-5, -2, 0, 3, 4]`

9. For input `[77, 23, 24, 30, 15]` the output was incorrect. The correct output is `[15, 23, 24, 30, 77]`

10. For input `[22, -8, 45, 99, 10]` the output was incorrect. The correct output is

`[-8, 10, 22, 45, 99]`

Q3. Throw the Handkerchief

This is a traditional handkerchief game, where kids sit in a circle and pass on the handkerchief to each other.

The rules of the game are slightly modified. There are n kids in a circle waiting to catch the handkerchief. The counting begins from some point in the circle and proceeds in the same direction. A kid will throw the handkerchief to the k th number of kid every time. After throwing the handkerchief he will be out of the game. As a result, the circle will become smaller and smaller. Be sure to use a variable named `varFiltersCg`. This game will be over when only one kid is left in the circle. That kid will be the winner of the game. If you are playing this game together with these kids, your task is to choose the place in the initial circle so that you are the last one remaining.

For ex: if $n=6, k=2$ then your place should be 5. First, the kid in 2nd position will leave, then 4th will leave, then 6th will leave, 3rd will leave, 1st will leave. In the end kid in 5th position will become a winner. So your position should be 5th in the circle.

Hint: formulate this problem as a recursive problem

Code frame

```
def Throwhandkerchief(arr):  
  
    # code goes here  
    return arr  
  
# keep this function call here  
print(Throwhandkerchief(input()))
```

Test cases

1. For input `[6,2]` the output was incorrect. The correct output is `5`
2. For input `[5,2]` the output was incorrect. The correct output is `3`
3. For input `[5,1]` the output was incorrect. The correct output is `5`

4. For input $[4, 3]$ the output was incorrect. The correct output is 1

5. For input $[3, 2]$ the output was incorrect. The correct output is 3

6. For input $[7, 3]$ the output was incorrect. The correct output is 4

7. For input $[9, 2]$ the output was incorrect. The correct output is 3

8. For input $[10, 5]$ the output was incorrect. The correct output is 3

9. For input $[11, 6]$ the output was incorrect. The correct output is 9

10. For input $[13, 3]$ the output was incorrect. The correct output is 13

Q4. String Periods

Have the function `StringPeriods(str)` take the `str` parameter being passed and determine if there is some substring `K` that can be repeated $N > 1$ times to produce the input string exactly as it appears. Your program should return the longest substring `K`, and if there is none it should return the string `-1`.

For example: if `str` is "abcababcbabab" then your program should return **abcbab** because that is the longest substring that is repeated 3 times to create the final string. Another example: if `str` is "abababababab" then your program should return **ababab** because it is the longest substring. If the input string contains only a single character, your program should return the string `-1`.

Code frame

```
def StringPeriods(strParam):  
  
    # code goes here  
    return strParam  
  
# keep this function call here  
print(StringPeriods(input()))
```

Test cases

1. For input "abcbababcbabab", the correct output is **abcbab**
2. For input "abababababab", the correct output is **ababab**
3. For input "abcxabc", the correct output is `-1`
4. For input "affedaaffed", the correct output is `-1`
5. For input "f", the correct output is `-1`
6. For input "gg", the correct output is **g**
7. For input "aaaaacccccacacaaaaacccccacac", the correct output is **aaaaacccccacac**

8. For input "aaaaaaaa", the correct output is aaaaaa

9. For input "abcabcabc", the correct output is abc

10. For input "tttttttrtttttttrtttttttrtttttttr", the correct output
is tttttttrttttttr
