*EXERCICE 1*

|  |
| --- |
| **WHAT YOUR PROGRAM SHALL DO** |
| We want to check if the array of numbers is containing the negative number, we count it.  We print:   * Number of negative of negative number if we found, * Otherwise print -1  |  |  | | --- | --- | | **Function name** | countNegNum | | **Parameters** | Array of number | | **Return value** | **Count of negative number** (number) | | **Examples** | countNegNum ([12, -1, 20, -4, 10, -29]) -> 3 | |

|  |  |
| --- | --- |
| **EXAMPLES** | |
| CONSOLE | EXPLANATION |
| > [2, -12, -4, 10, -9]  > 3 | We count the number of negative numbers in array:  [2, **-12**, **-4**, 10, **-9**] = **3**  So, the count of negative number is **3** |
| > [0, 1, 2, 3, 4]  > -1 | We count the number of negative numbers in array:  [0, 1, 2, 3, 4] = 0  So, if there are no negative number, we return -**1** |
| > []  > -1 | We count the number of negative numbers in array:  [] = **0**  So, if there are no negative number, we return -**1** |

*EXERCICE 2*

|  |
| --- |
| **WHAT YOUR PROGRAM SHALL DO** |
| We want to find the index of the minimum number.  To perform this program, you MUST code and use this function:   |  |  | | --- | --- | | **Function name** | getIndexMinNumber | | **Parameters** | array | | **Return value** | **Index of minimum number** | | **Examples** | getIndexMinNumber ([ 1, 4, 6, 9, 3, 0, 4, 5])->5 | |

**Warning:** if the array is empty, we return **None**.

|  |  |
| --- | --- |
| **EXAMPLES** | |
| CONSOLE | EXPLANATION |
| > [4,1,3]  >1 | The answer is **1** because:  [ 4, 1, 3]  Index of array 0 **1** 2 -> **1**  We need to find the minimum number first, then we need to find the index of that minimum number.  So, we return **1** |
| > [1, 2, 3, 5]  >0 | The answer is **0** because:  [1, 2, 3, 5]  Index of array **0**  1 2 3 -> **0**  We need to find the minimum number first, then we need to find the index of that minimum number.  So, we return **0** |
| > []  >None | Now, we have an empty array.  Array = []  So, we return **None** |

*EXERCICE 3*

|  |
| --- |
| **WHAT YOUR PROGRAM SHALL DO** |
| First you will to implement the following function:   |  |  | | --- | --- | | **Function name** | getIndexOfNumberFive | | **Parameters** | An array | | **Return value** | First index of number 5 | | **Examples** | getIndexOfNumberFive([1,7,6,5,3,5,4,5]) -> **3** | |

|  |  |
| --- | --- |
| **EXAMPLES** | |
| CONSOLE | EXPLANATION |
| > [4,1,3]  > None | The answer is **None**  So, we return **None**, because no number 5 in array |
| > [1, 2,3,5]  >3 | The answer is **3** because:  [ 1, 2, 3, 5]  0 1 2 **3** -> **3**  So, we return **3** |
| >[9, 7, 4, 3, 5, 4, 5]  >4 | The answer is **4**:  [9, 7, 4, 3, 5, 4, 5]  0 1 2 3 **4** 5 **6** -> **4**  So, we return **4,** because we find only the **first** index. |
| > []  > None | The answer is **None**:  So, we return **None**, because the array is empty. |

*EXERCICE 4*

|  |
| --- |
| **WHAT YOUR PROGRAM SHALL DO** |
| First you need to implement the following function:   |  |  | | --- | --- | | **Function name** | sumOddEvenNumber | | **Parameters** | A list of number (array) | | **Return value** | array of (sum of Even and sum of Odd) ex: return [sumOfEven, sumOfOdd] | | **Examples** | example:  **INPUT**  listArray = [1,2,3,4]  **OUTPUT**  Sum of Even is **6**, and sum of Odd is **4**. | | **Warning** | If there is empty array. We print:  **Output:**  **There are no even and odd number** | |

|  |  |
| --- | --- |
| **EXAMPLES** | |
| CONSOLE | EXPLANATION |
| > [5,6,7]  > Sum of Even is **6**, and sum of Odd is **12**. | The sum of even is 6 and the sum of odd is 12.  > [**5**,**6**,**7**]  The green number is **odd** and the red number is **even**.  sum of odd number is **12**, and sum of even number is **6.**  So, we return [6,12] |
| > []  > There are no even and odd number | There is no sum of even and the sum of odd number.  Because we enter the empty array.  So, we return [0,0] |
| > [2,4,6]  > Sum of Even is **12**, and sum of Odd is 0. | The sum of even is 6 and the sum of odd is 12.  > [**2**,**4**,**6**]  The red number is **odd** and the green number is **even**.  sum of even number is **12**, and we didn’t find the odd number**.**  So, we return [12,0] |

*EXERCICE 5*

|  |
| --- |
| **WHAT YOUR PROGRAM SHALL DO** |
| In this exercise we need to replace every number **9** to **168** in array.  First you need to implement the following function:   |  |  | | --- | --- | | **Function name** | replaceNumber | | **Parameters** | An array of number | | **Return value** | Array of number that already replaced. | | **Examples** | replaceNumber ([1,9,3,4,6,5,9,12]) 🡪 [1,168,3,4,6,5,168,12]  Explanation: we need to replace number 9 to 168**:**  [1,**9**,3,4,6,5,**9**,12] 🡪 [1,**168**,3,4,6,5,**168**,12] | |

|  |  |
| --- | --- |
| **EXAMPLES** | |
| CONSOLE | EXPLANATION |
| > [12, 6, 9, 45,0]  > [12, 6, 168, 45,0] | We have an array, and we need to find the number 9 in array.  When we found we need to replace number **9** to **168**.  [12, 6, **9**, 45,0] -> [12, 6, **168**, 45,0]  So, we return the new array that we already replaced.  We return [12, 6, **168**, 45,0] |
| > [12,89,9,20]  > [12,89,9,20] | We have an array, and we need to find the number 9 in array.  For this case we didn’t find number **9**.  So, we return the **same array** that we entered in the console. |
| >[]  >[] | We need to check **elements** in an **array** if we entered an empty array no need to check anymore just return that empty array is done.  So, we return **[]** |

*EXERCICE 6*

|  |
| --- |
| **WHAT YOUR PROGRAM SHALL DO** |
| In this exercise we need to replace every letter A to **$**  First you need to implement the following function:   |  |  | | --- | --- | | **Function name** | replaceLetter | | **Parameters** | An array of string | | **Return value** | Array of string that already replaced. | | **Examples** | replaceLetter (["ronan","rady","him","ratha","mengheang"]) 🡪 ['ron$n', 'r$dy', 'him', 'r$th$', 'menghe$ng']  Explanation: we need to replace letter **A** to **$:**  replaceLetter(["ron**a**n","r**a**dy","him","r**a**th**a**","menghe**a**ng"])  so, we return ['ron**$**n', 'r**$**dy', 'him', 'r**$**th**$**', 'menghe**$**ng'] | |

|  |  |  |
| --- | --- | --- |
| **EXAMPLES** | | |
| CONSOLE | EXPLANATION | |
| > ["ron**a**n","r**a**dy","him","r**a**th**a**","menghe**a**ng"]  > ['ron**$**n', 'r**$**dy', 'him', 'r**$**th**$**', 'menghe**$**ng'] | we need to replace letter **A** to **$:**  we find latter a:["ron**a**n","r**a**dy","him","r**a**th**a**","menghe**a**ng"]  we replace it to $: ['ron**$**n', 'r**$**dy', 'him', 'r**$**th**$**', 'menghe**$**ng']  so, we return ['ron**$**n', 'r**$**dy', 'him', 'r**$**th**$**', 'menghe**$**ng'] |
| > ["sophana","somphors","pros"]  > ['soph$n$', 'somphors', 'pros'] | we need to replace letter **A** to **$:**  we find latter a: ["soph**a**n**a**","somphors","pros"]  we replace it to $: ['soph**$**n**$**', 'somphors', 'pros']  so, we return ['soph**$**n**$**', 'somphors', 'pros'] | |
| >[]  >[] | We need to check every letter **A** in an **array** if we entered an empty array no need to check anymore just return that empty array is done.  So, we return **[]** | |