1. In the game of ScrabbleTM, each letter has points associated with it. The total score of a word is the sum of the scores of its letters. More common letters are worth fewer points while less common letters are worth more points. The points associated with each letter are shown below:

|  |  |
| --- | --- |
| Points | Letters |
| 1 | A, E, I, L, N, O, R, S, T, U |
| 2 | D, G |
| 3 | B, C, M, P |
| 4 | F, H, V, W, Y |
| 5 | K |
| 8 | J, X |
| 10 | Q, Z |

Write a program that computes the ScrabbleTM score for a word.

|  |  |
| --- | --- |
| Requirement | |
| Input | A string with only letters, case insensitive |
| Output | The ScrabbleTM score of the string |
| Other | Write your code in func1 |

Test Cases

|  |  |
| --- | --- |
| Input | Output |
| MIT | 5 |
| ﻿Princeton | 13 |
| Soochow | 15 |

1. Write a program that performs simple run-length encoding, which converts a given string, for example, BWWWWWBWWWW, to the shorter 1B5W1B4W.

|  |  |
| --- | --- |
| Requirement | |
| Input | A string with only upper-case letters |
| Output | A string after conversion |
| Other | Write your code in func2 |

Test Cases

|  |  |
| --- | --- |
| Input | Output |
| ABAB | 1A1B1A1B |
| AABB | 2A2B |
| AAAABBAAAAAAAAAAAABBB | 4A2B12A3B |

1. Given a list of size where each element is a subset of {1, 2, …, }, find the majority element. The majority element is the element that appears more than times.

|  |  |
| --- | --- |
| Requirement | |
| Input | A list |
| Output | The majority element if exist, and None otherwise |
| Other | Write your code in func3 |

Test Cases

|  |  |
| --- | --- |
| Input | Output |
| [{1}, {2}, {1}] | {1} |
| {{1}, {2}} | None |
| [{1,2}, {1,2}, {1,2}, {1}, {2}] | {1,2} |

1. Given a list, reverse the items in it. Note that if an item is a list, its items should also be reversed.

|  |  |
| --- | --- |
| Requirement | |
| Input | A list |
| Output | A reversed list |
| Other | Write your code in func4 |

Test Cases

|  |  |
| --- | --- |
| Input | Output |
| [1, 2, 3] | [3, 2, 1] |
| [1, [2, 3, 4], 5] | [5, [4, 3, 2], 1] |
| [1, [2, [3, 4, 5], 6], 7] | [7, [6, [5, 4, 3], 2], 1] |

1. Given a set of infinite bills including $1, $2, $5, and $10, you task is to find how many different ways you can pay $ using different denominations of bills. For example, when , you have 5 different ways as shown below:

* $1, $1, $1, $1, $1, $1
* $1, $1, $1, $1, $2
* $1, $1, $2, $2
* $1, $5
* $2, $2, $2

Note that although each bill has a different serial number, you view two bills as identical if they have the same denomination.

|  |  |
| --- | --- |
| Requirement | |
| Input | A positive integer |
| Output | The number of different ways paying $ |
| Other | Write your code in func5 |

Test Cases

|  |  |
| --- | --- |
| Input | Output |
| 2 | 2 |
| 6 | 5 |
| 10 | 11 |

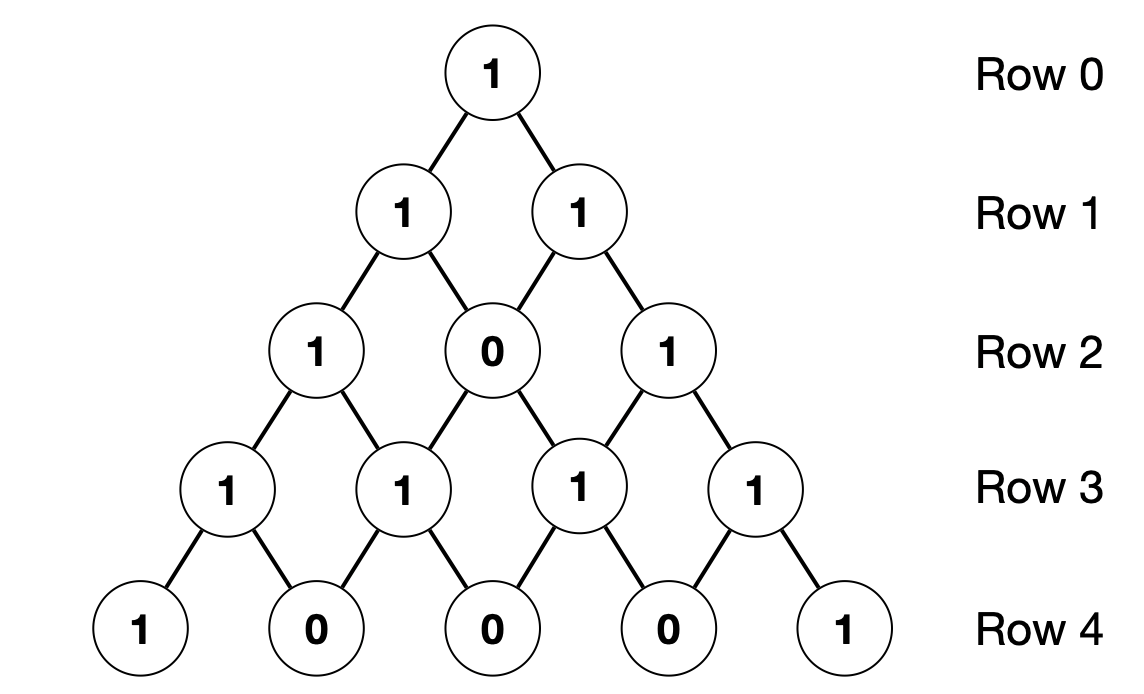
1. Given a string which will contain single digit numbers, letters, and question marks, write a program to check if there are **EXACTLY** 3 question marks between **EVERY** pair of two numbers that add up to 10. If so, your program should return True, otherwise it should return False. If there are not any two numbers that add up to 10 in the string, your program should return False as well.

|  |  |
| --- | --- |
| Requirement | |
| Input | A string |
| Output | The row where the -th zero is in |
| Other | Write your code in func6 |

Test Cases

|  |  |
| --- | --- |
| Input | Output |
| aa6?9 | False |
| acc?7??sss?3rr1??????5 | True |
| arrb6???4xxbl5???eee5 | True |

1. XOR Pascal's triangle can be constructed in the following manner: In row 0 (the topmost row), there is a nonzero entry 1. Each entry of each subsequent row is constructed by XORing the number above and to the left with the number above and to the right, treating blank entries as 0. An example is illustrated as below.



Given an integer , write a program that computes the row where the -th zero is in.

|  |  |
| --- | --- |
| Requirement | |
| Input | A positive integer |
| Output | The row where the -th zero is in |
| Other | Write your code in func7 |

Test Cases

|  |  |
| --- | --- |
| Input | Output |
| 1 | 2 |
| 2 | 4 |
| 3 | 4 |

1. Write a program that removes consecutive duplicate words in a given string.

|  |  |
| --- | --- |
| Requirement | |
| Input | A string |
| Output | A string that does not have consecutive duplicate words |
| Other | Write your code in func8 |

Test Cases

|  |  |
| --- | --- |
| Input | Output |
| The the the cow jumped over over the the moon. | The cow jumped over the moon. |