**Question 1 (Fibonacci prime)** **35 marks**

A Fibonacci prime is a Fibonacci number and it is prime number. Given a list of integer numbers you need to find if the given number is prime number as well as it occurs in the Fibonacci series.

**Input**
Input consists of a list of integer numbers N where 2 <= N <= 31999

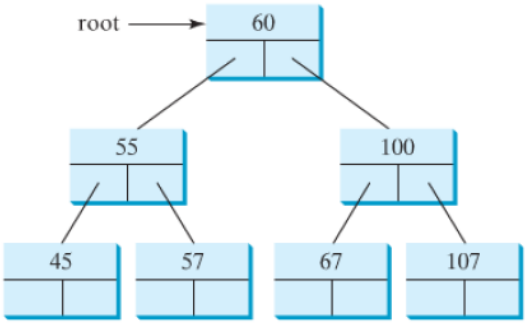
**Output**
The output contains the list of characters which contains entry as 'y' if the corresponding integer in the input array is a Fibonacci prime else 'n'.

| Test case | Input | Output | Marks |
|---|---|---|---|
| UTC01_01 | 10, 3, 5, 4, 89 | 'n', 'y', 'y', 'n', 'y' | 35 |
| UTC01_02 | 5, 5, 10, 89, 1597 | 'y', 'y' ,'n', 'y', 'y' | |
| UTC01_03 | 191, 233, 13, 144, 101 | 'n', 'y', 'y', 'n', 'n' | |

**Question 2 (Binary Tree)** **35 Marks**

A binary tree is a tree data structure in which each node has at most two child nodes, usually distinguished as "left" and "right". Nodes with children are parent nodes, and child nodes may contain references to their parents. Outside the tree, there is often a reference to the "root" node (the ancestor of all nodes), if it exists. Any node in the data structure can be reached by starting at root node and repeatedly following references to either the left or right child.

**Representing a binary tree:**
A binary tree can be represented using a set of linked nodes. Each node contains a value and two links named left and right that reference the left child and right child.



**Tree insertion**

If a binary tree is empty, create a root node with the new element. Otherwise, locate the parent node for the new element node.
If the new element is less than the parent element, the node for the new element becomes the left child of the parent. If the new element is greater than the parent element, the node for the new element becomes the right child of the parent. Duplicate element is not inserted.

**Tree Traversal**

Tree traversal is the process of visiting each node in the tree exactly once. There are several ways to traverse a tree:

- The inorder traversal follows the recursive visitation sequence: Left child, Root, Right child.
- The postorder traversal follows the recursive visitation sequence: Left Child, Right child, Root
- The preorder traversal follows the recursive visitation sequence: Root, Left child, Right child

For example, in the following tree:

- inorder is: {45, 55, 57, 60, 67, 100, 107 }
- postorder is: {45, 57, 55, 67, 107, 100, 60 }.
- preorder is: {60, 55, 45, 57, 100, 67, 107}

Implement the functionalities of BinaryTree class to store the elements in a binary tree and return the data by traversing them using inorder, postorder and preorder.

| Test Case | Input | Output | Marks |
|---|---|---|---|
| UTC02_01 | Add elements {60, 55, 100, 57, 107,67,45}<br>binaryTree.getData("inorder") | {45, 55, 57, 60, 67, 100, 107 } | 15 |
| UTC02_02 | Add elements {5,2,8,7,9,10}<br>binaryTree.getData("inorder") | {2,5,7,8,9,10 } | |
| UTC02_03 | Add elements: {60, 55, 100, 57, 107,67,45}<br>binaryTree.getData("postorder") | {45, 57, 55, 67, 107, 100, 60 } | 10 |
| UTC02_04 | Add elements: {60, 55, 100, 57, 107,67,45}<br>binaryTree.getData("preorder") | {60, 55, 45, 57, 100, 67, 107} | 10 |

**Question3 (Reverse the words in a sentence)** **30 Marks**

Given a sentence which consists of alphabets [a-zA-Z], digits [0-9], “,” and “.”.

Implement a method to return a string after reversing only the alphabets [a-zA-Z] in a sentence, which is passed as an argument to a method.

**Note:** Reversing has to be done word-by-word.

**Example:** If sentence contains “Hello World”, the reversed sentence will be “dlroW olleH”

| Test Case | Input | Output | Marks |
|---|---|---|---|
| UTC01_01 | 1 cup of hot coffee costs 8.00, whereas cold coffee costs 45.00. | 1 puc fo toh eeffoc stsoc 8.00, saerehw dloc eeffoc stsoc 45.00. | 35 |
| UTC01_02 | It Costs 25000rs for 1 LCD Projector. | tI stsoC 25000sr rof 1 DCL rotcejorP. | |
| UTC01_03 | 8990.33 | 8990.33 | |