Question 1 :

Arjun is in the street where his house is located. Help Arjun to decide the nearest park from his home. At most a street can be split into two. Each street is represented with a number. Street number will be its length. Arjun’s house street number is a starting number. Park’s street number is a multiple of 4. Arjun is afraid of dogs and avoids streets having dogs. Streets having dogs will be represented as odd numbers. Print all the park's street number and total travelling distance from nearest to farthest in a new line. If no park is found return -1.

Input Description :

First line consists of an integer (n) representing the number of elements
Second Line consists of a list of integers representing the street numbers.

Output Description :

Print park’s street number and total travelling distance starting from nearest to farthest one in a new line.

Solution :

```
class Node:
    def __init__( self ,  data ) :
        self.data = data
        self.left = None
        self.right = None

class BinaryTree:
    def __init__(self):
        self.root = None

def createBinaryTree(lst, n, node, i):
    if i < n:
        if lst[i] == -1:
            return node
        temp = Node(lst[i])
        node = temp

        node.left = createBinaryTree(lst, n, node.left, (2*i)+1)

        node.right = createBinaryTree(lst, n, node.right, (2*i)+2)

    return node
```

```
def findNearestPark(root, dist, dict_):
    if root == None:
        return
    if root.data % 2 == 1:
        return
    if root.data % 4 == 0:
        dict_[dist] = root.data

    findNearestPark(root.left, dist+root.data, dict_)

    findNearestPark(root.right, dist+root.data, dict_)

    return dict_

n = int(input())
lst = [int(x) for x in input().split()[:n]]

tree = BinaryTree()
tree.root = createBinaryTree(lst, n, tree.root, 0)
dict_ = { }
park_dist = findNearestPark(tree.root, 0, dict_)

if park_dist == None:
    print(-1)
elif len(park_dist) == 0:
    print(-1)
else:
    for i in sorted(park_dist.keys()):
        print(park_dist[i], i)
```

Test case 1 :

Input :
7
2 3 4 5 6 7 8
Output :
4 2
8 6

Test case 2 :

Input :
7
2 3 -1 5 6 7 8
Output :
-1

Test case 3 :

Input :
17
6 2 9 2 1 7 9 4 8 11 12 16 -1 12 -1 15 5
Output :
8 10

Test case 4 :

Input :
12
2 3 6 4 8 11 12 -1 3 6 16 2
Output :
12 8

Test case 5 :

Input :
5
6 5 4 3 2
Output :
4 6