```python
# QUESTIOON 1 -> Worked on by Sruthi Santhosh and Jade Hudson
def search_q1(Y, n, x):

    for i in range (0, n):
        if Y[i] == x:
            return i
    return -1


Y = ['apple', 'banana', 'mango', 'grapes', 'pineapple', 'durian']
x = "pineapple"

n = len(Y)
result = search_q1(Y, n, x)

if result == -1:
    print("Element is not present in the list")
else:
    print("Element", x, "is present at index", result)

# This is a linear function, the time complexity is f(n)= n - the for loop searches the entire list from start to finish until it comes across the target.


# QUESTION 2 -> Worked on by Sruthi Santhosh and Jade Hudson
def search_q2(X, item):
    first = 0
    last = len(X)-1
    found = False
    while first<=last and not found:
        mid = (first + last)//2
        if X[mid] == item:
            found = True
            print("The element item", item, "was found at index ", X.index(60))
        else:
            if item < X[mid]:
                last = mid - 1
            else:
                first = mid + 1
    return found


print(search_q2([10, 15, 35, 42, 60, 70, 82, 94], 60))

# The code has a time complexity of O(log n), this is because this code is a binary search algorithm
# Binary search algorithms work by taking half of the input and comparing the value there to the target.
# If the value is higher (or lower), it will take the higher (or lower) half and repeat.


# QUESTION 3 -> Worked on by Sruthi Santhosh and Jade Hudson
test = 0
n = 10
for i in range(n):
   test = test + 1

for j in range(n):
   test = test - 1

# This code has a time complexity of f(n)=n, This is a linear function because the function has to run n number of times in both of the for loops.


# QUESTION 4 -> Worked on by Sruthi Santhosh and Jade Hudson
i = n
while i > 0:
   k = 2 + 2
   i = i // 2

# This is a linear function with a time complexity of f(n)=n. The while loop has to run n number of times until it reaches the end.


# QUESTION 5 -> Worked on by Sruthi Santhosh and Jade Hudson
mat = [[1, 2, 3], [1, 1, 1], [5, 7, 8]]
add = 0
for i in range(len(mat)):
    for j in range(len(mat[0])):
        add += mat[i][j]
print(add)

# This is a quadratic equation with a time complexity of f(n) = n^2
# This is because there is a for loop in a for loop aka a nested loop


# QUESTION 6 -> Worked on by Sruthi Santhosh and Jade Hudson
def fibonacci(n):
    if n<2:
        return n
    return fibonacci(n-1) + fibonacci(n-2)


for n in range(2,12,2):
    print("Series sum for {} is {}".format(n, fibonacci(n)))

# This is a time complexity of O(2^n)
# This is because this is function, calling the same function inside of said function.
```