ASSIGNMENT 2

Question 1: Given the names and grades for each student in a class of N students, store them in a nested list and print the name(s) of any student(s) having the second lowest grade.

Note: If there, are multiple students with the second lowest grade, order their names alphabetically and print each name on a new line.

Example records=[["chi", 20.0"],["beta", 50.0],["alpha", 50.0]]
The ordered list of scores is [20.0, 50,0], so the second lowest score is 50.0. There are wo sudents with that score: [“beta”,”alpha”]. Ordered alphabetically, the names are printed as

```
alpha
beta
```

```
def second_lowest_grade_students(records):
    records = [["chi", 20.0], ["beta", 50.0], ["alpha", 50.0]]
    # Sort the records based on grades
    records.sort(key=lambda x: (x[1], x[0]))  # Sort by grade first, then by name

    # Find the second lowest grade
    second_lowest_grade = None
    for i in range(1, len(records)):
        if records[i][1] > records[0][1]:
            second_lowest_grade = records[i][1]
            break

    # Collect names of students with the second lowest grade
    second_lowest_students = []
    for student in records:
        if student[1] == second_lowest_grade:
            second_lowest_students.append(student[0])

    # Sort the names alphabetically
    second_lowest_students.sort()

    return second_lowest_students
# Find and print the names of students with the second lowest grade
result = second_lowest_grade_students(records)
for student in result:
    print(student)
```

```
alpha
beta
```

Question 2: Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution. And you may not use the same element twice. You can return the answer in any order.

```
def two_sum(nums, target):
    nums = [2, 7, 11, 15]
    target = 9
    # Create a dictionary to store the complement of each number
    num_indices = {}

    # Iterate through the array
    for i in range(len(nums)):
        complement = target - nums[i]
        # If the complement exists in the dictionary, return its index along with the current index
        if complement in num_indices:
            return [num_indices[complement], i]
        # Otherwise, add the current number and its index to the dictionary
        else:
            num_indices[nums[i]] = i
print(two_sum(nums, target))
```

```
[0, 1]
```