```
import numpy as np
'''
Samuel Haberkorn
CS2300 - Computational Linear Algebra.3
Project 3
11/13/2020
'''

#Predefined problems from assignment sheet

sets = [
    # Basis 1,  Basis 2, Candidate
    [[1, 0, 1], [0,1,0], [15, -10, 15]],
    [[6.9, 0, 0], [0,-3.2, 0], [1.5, -6.2, 0.37]],
    [[-3,3,4], [0,1,0], [-6, 4, -8]],
    [[-3,3,4], [1,1,1], [1,7,8]]
]


def main():
    #Selector for which mode to run in. All, single, or custom
    for i, set in enumerate(sets):
        print(f"{i + 1}. Basis set = {set[0]}, {set[1]} and candidate vector {set[2]}")
    choice = int(input("Please select your problem, enter zero to run them all or -1 to enter your own: "))
    if choice == 0:
        for problem in sets:
            runProblem(problem)
    elif choice == -1:
        runProblem(gatherSet())
```

```
    else:
        runProblem(sets[choice-1])



 #Function to get a set of vectors from the user
def gatherSet():
    vectors = []
    for i in range(2):
        print(f"Enter Basis Vector {i}")
        vectors.append([float(input("x1: ")), float(input("x2: ")), float(input("x3: "))])
    print(f"Enter Candidate Vector")
    vectors.append([float(input("x1: ")), float(input("x2: ")), float(input("x3: "))])

    return vectors

# function to handle the solving of system of equations
def solve_system(a, b):
    try:
        x = np.linalg.solve(a, b)
    except np.linalg.LinAlgError:
        return None
    return x

# runs the problem
def runProblem(chosen):
    # checks for a solution with the 1st 2 equations
    a = np.array([[chosen[0][0], chosen[1][0]], [chosen[0][1], chosen[1][1]]])
    unused = [chosen[0][2], chosen[1][2], chosen[2][2]]
    b = np.array(chosen[2][:2])
```

```
    x = solve_system(a, b)
    if x is None:
        #checks for a solution with the last 2 equations
        a = np.array([[chosen[0][1], chosen[1][1]], [chosen[0][2], chosen[1][2]]])
        b = np.array(chosen[2][1:])
        unused = [chosen[0][0], chosen[1][0], chosen[2][0]]
        x = solve_system(a, b)
        if x is None:
            # checks for a solution with equation 1 and 2
            a = np.array([[chosen[0][0], chosen[1][0]], [chosen[0][2], chosen[1][2]]])
            b = np.array([chosen[2][0], chosen[2][2]])
            unused = [chosen[0][1], chosen[1][1], chosen[2][1]]
            x = solve_system(a, b)

    # if none of these equations have solutions, then it's singular
    if x is None:
        print(f"Singular! There are {'infinite' if chosen[0][0] != 0 else 'no'} solutions.")
        return

    # check if it's valid by plugging the solutions into the equation not used. If it gets and answer, we are
good!
    status = (x[0] * unused[0]) + (x[1] * unused[1]) == unused[2]

    # print confirmation message
    print(f"{'Yes,' if status else 'No'} the vector {chosen[2]} is {'' if status else 'not '}in the subspace
spanned by {chosen[0]}, {chosen[1]}")

if __name__ == '__main__':
```

```
print("Samuel Haberkorn\nProject 3\n\n")
main()
```