The University of Hong Kong
COMP1117B Computer Programming
Assignment 3

Deadline date: Apr 8 11:59 pm

Reminder: This assignment involves some console input/output. You are reminded that the VPL system on HKU Moodle evaluates your program with a full score under the condition that your program output is the EXACT MATCH of the expected output. In other words, any additional or missing space character, tab character, newline character, etc. will be treated as errors during the evaluation of your program. Also, you are advised to make more test cases on your own for testing your program.

Question 1 (35%): Exist or not?

Task Description:

In this task, we will develop a program to detect whether a **target** number is the sum of **k** elements in the given list **numbers** (note that the elements in **numbers** are positive integers and we input the integers one by one, one integer at a time, and terminated by -1). Please carefully read the below cases for a better understanding.

```
Case 1
INPUT:
1
2
8
9
39
92
161
-1
209
3
OUTPUT:
Yes
Explanation:
It print “Yes” because there are 3 numbers in the given list that sum up to 209 (9 +
39 + 161)
```

```
Case 2
INPUT:
32
392
93022
349895
-1
1280298402
2
OUTPUT:
No
Explanation:
It print “No” because we cannot find 2 numbers in the given list [32, 293, 93022,
349895] to sum up to 1280298402
```

Given program

```
1. number = int(input())
2. numbers = []
3. while number != -1:
4.     numbers.append(number)
5.     number = int(input())
6. target = int(input())
7. k = int(input())
8.
9.
10. def isExist(# define parameters):
11.     # your implementation
12.
13.
14. if isExist(# input the appropriate variables):
15.     print("Yes")
16. else:
17.     print("No")
```

1. You are expected to use recursion algorithm to solve this problem. It will have a 20% deduction if your submission is not a recursion solution.
2. Hint: If the current number **x** should be included in finding the **target** number, then summing up **k** elements to find the target will be equivalent to summing up **[k-1]** elements to find **[target -x]**. Similarly, try to figure out the recurrence relationship for the situation where **x** should not be included in the process.
3. Please read the given code carefully for better understanding.
4. You can start the recursion on is_exist(), or you can define another recursive function and use is_exist() to call it. You may define as many functions as you need but you are not required to do so.
5. You must show the output of the function using the print() provided above.
6. You are not allowed to change any line of code in the given program.
7. Make sure your program can handle input range 1 < **k** < 6, where **k** is a positive integer. For example, in the above case ([1, 2, 8, 9, 39, 92, 161], 109, 3), the **k** is 3.
8. You can use any built-in functions in Python.

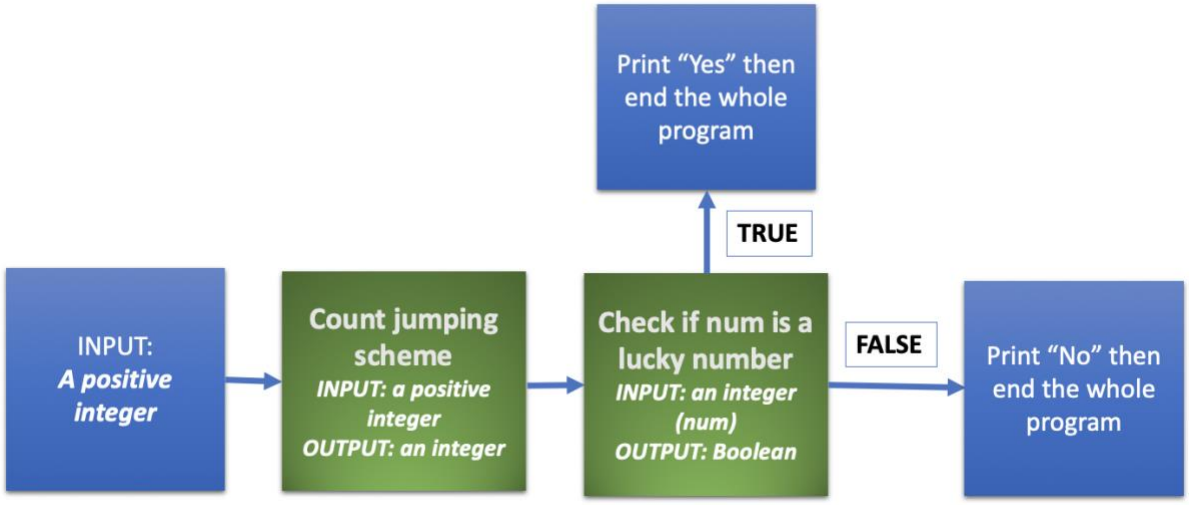
Question 2 (65%): Result-guessing game

Task Description:

Gary and Tom are playing a result-guessing game together. Overall, this game is simple, the judge will give a positive integer to the participants, and then the participants need to judge whether the given number list is fulfilling the requirements (below will have a detailed requirement description about the requirement). The winner will be the one who can give the fastest and correct answer. However, Tom is not good at logic, which makes his winning chance low. Therefore, in this task, please write a program to help Tom to win this game.

Overview of the expected program:

Overview of the requirements:



Count jumping scheme

The expected input of this module is a positive integer. Assuming we want to go to a destination that needs to jump **number** step(s). Here is the question: if we can only jump 1 or 2 step(s) for each time, how many possible ways are available to jump to the destination?

Example:
number = 3
The output will be 3 because there is total 3 schemes for jumping 3 steps.
1st scheme: jump 1 step + jump 1 step + jump 1 step
2ed scheme: jump 1 step + 2 steps
3rd scheme: jump 2 steps + 1 step

Check whether a number is a lucky number

The input of this module will be a positive **Integer** and the output will be a **Boolean**. Note that the lucky number is determined by the below rule:

abs((The most frequent digit ***** its frequency) – (the least frequent digit * its frequency))

If the number resulting from the above formula ends with 6, then it is considered as a lucky number. **Note** 1) If there are equal frequencies among the digits in the test cases, then the larger digit will be used for calculation. 2) The least frequency does not include zero.

Example:
number = 7398827232
There are three 2's, which is the most frequent digit in the number; there is only one 9 in the number. By applying the formula above, we'll have abs((2*3)-(9 *1)) = abs(-3) = 3. This number does not end with 6, so it is not a lucky number.

number = 330
There are two 3's, which is the most frequent digit in the number; there is only 1 0' in the number. By applying the formula above, we'll have abs((3*2)-(0*1)) = abs(6) = 6. This number ends with 6, so it is a lucky number.

For the whole process

For examples:

INPUT: **number** = 500
OUTPUT: "No"
The **number** will be first proceeded by the count jumping scheme module, and the result will be 225591516161936330872512695036072072046011324913758190588638866418474627738 6

868834050159870527969684986 26. It's not a lucky number, therefore the program will print "No".
INPUT: **number** = 18
OUTPUT: "Yes"
The **number** will be first proceeded by the count jumping scheme module, and the result will be 4181. It's a lucky number, therefore, the program will print "Yes".

Given program

```
1. def number_to_digitLis_reversed_order(num):
2.     digits = []
3.     while num != 0:
4.         digits.append(num % 10)
5.         num //= 10
6.     return digits
7.
8.
9. def numbers_to_digitLis_original_order(num):
10.    reversed_digits = number_to_digitLis_reversed_order(num)
11.    n = len(reversed_digits)
12.    original_digits = []
13.    for i in range(0, n):
14.       original_digits.append(reversed_digits[n-i-1])
15.    return original_digits
16.
17.
18.def is_lucky_number (num):
19.     digits = numbers_to_digitLis_original_order(num)
20.     # your implementation
21.
22.
23.def count_jumping_scheme(num):
24.     # your implementation
25.
26.
27.number = int(input())
28.num_of_schemes = count_jumping_scheme(number)
29.if is_lucky_number(num_of_schemes):
30.     print("Yes")
31.else:
32.     print("No")
```

N.B.

1. Make sure your program can handle input 1 < **num** < 1000.
2. You can use any built-in functions in Python.
3. Try to define your own functions for each of the modules to make the flow clear.
4. Please read the given code carefully for better understanding.
5. You must show the output of the function using the print() provided above.
6. You are not allowed to change any line of code in the given program.
7. You can use any built-in functions in Python.

Appendix

Important Notes:

- Your program must follow the formats of the sample inputs/outputs strictly.
- We will grade your programs with another set of test cases (i.e., not limited to the sample test cases presented in this document/VPL).

Policy on “Plagiarism” according to the General Office:

- Plagiarism is a very serious academic offence. Students should understand what constitutes plagiarism, the consequences of committing an offence of plagiarism, and how to avoid it.
- Definition of Plagiarism:
  - As defined in the University's Regulations Governing Conduct at Examinations, plagiarism is "the unacknowledged use, as one's own, of work of another person, whether or not such work has been published.", or put it simply, plagiarism is copying (including paraphrasing) the work of another person (including an idea or argument) without proper acknowledgement.
  - In case of queries on plagiarism, students are strongly advised to refer to "What is Plagiarism”.
- If a student commits plagiarism, with evidence after investigation, no matter whether the student concerned admits or not, a penalty will be imposed:
  - First Attempt: if the student commits plagiarism (in an assignment/test of a CS course) for the first time in his/her entire course of study, the student shall be warned in writing and receive zero mark for the whole assignment or the whole test; if the student does not agree, s/he can appeal to the BEng(CompSc) Programme Director within a week;
  - Subsequent Attempt: if the student commits plagiarism more than once in higher course of study, the case shall be referred to the Programme Director for consideration. The Programme Director shall investigate the case and consider referring it to the University Disciplinary Committee, which may impose any of the following penalties: a published reprimand, suspension of study for a period of time, fine, or expulsion from the University.
- Both the student who copies other's work and the student who offers his/her work for copying shall be penalized.
- Teachers should report plagiarism cases to the General Office for records and the issuing of warning letters.