NAME:

NUMBER:

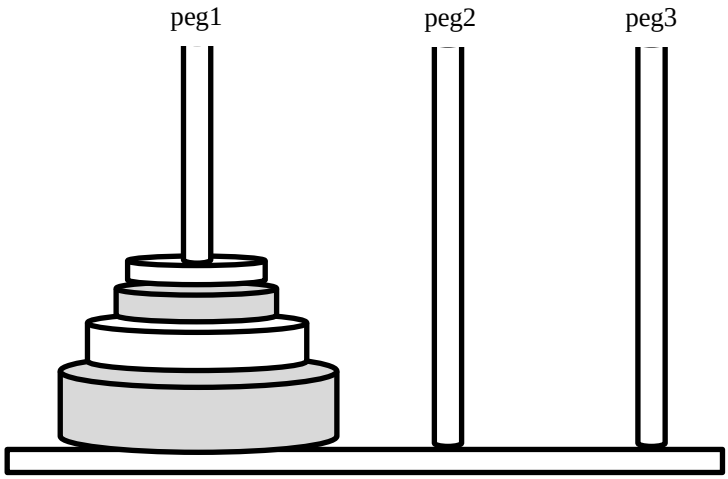
Date: 03.02.2021, Thursday
Time: 13:30 (90 minutes)
Place: D1
Lecturer: İrem Ülkü

ANKARA UNIVERSITY, COMPUTER ENGINEERING DEPARTMENT

COM1001: Computer Programming I (Fall 2021-22) - RESIT EXAM (110P)

(32P) QUESTION 1:

The Towers of Hanoi is one of the most famous classic problems in computer science. We want to move a stack of disks from one peg to another. The following diagram shows the pegs with four disks on peg 1.



The disks threaded onto one peg and arranged from bottom to top by decreasing size. We can move the stack from one peg to another under the constraints that exactly one disk is moved at a time and at no time may a larger disk be placed above a smaller disk. Three pegs are provided, one being used for temporarily holding disks.

Let's assume that we want to move the disks from peg 1 to peg 3. Moving n disks can be viewed in terms of moving only n – 1 disks (recursion), as follows:

a) Move n – 1 disks from peg 1 to peg 2, using peg 3 as a temporary holding area.

b) Move the last disk (the largest) from peg 1 to peg 3.

c) Move the n – 1 disks from peg 2 to peg 3, using peg 1 as a temporary holding area.

The process ends when the last task involves moving n = 1 disk (i.e., the **base case**). This base case is accomplished by simply moving the disk, without the need for a temporary holding area.

The program in the next page uses a recursive function with four parameters: 1- The number of disks to be moved, 2- The peg on which these disks are initially threaded, 3- The peg to which this stack of disks is to be moved and 4- The peg to be used as a temporary holding area. The program displays instructions for moving the disks from the starting peg to the destination peg as follows:

1 → 3 (This means move one disk from peg 1 to peg 3.)

1 → 2

3 → 2

1 → 3

2 → 1

2 → 3

1 → 3

Fill the blank spaces **1**, **2**, **3, 4, 5, 6, 7,** and **8** in this code.

```
def solve_towers(disks, source_peg, destination_peg, temp_peg):
    # base case -- only one disk to move
    if disks == 1:
        print("{} --> {}".format(source_peg, destination_peg) )
        return
    solve_towers(----------1,----------2,----------3,----------4)
    print("{} --> {}".format(source_peg, destination_peg) )
    solve_towers(----------5,----------6,----------7,----------8)

start_peg = 1  # value 1 used to indicate start_peg in output
end_peg = 3  # value 3 used to indicate end_peg in output
temp_peg = 2  # value 2 used to indicate temp_peg in output
total_disks = 3  # number of disks

solve_towers(total_disks, start_peg, end_peg, temp_peg)
```

(32P) QUESTION 2:

An instructor teaches a class in which each student takes three exams. The instructor would like to store this information in a file named grades.dat for later use. Below code enables an instructor to enter each student's first name and last name as strings and the student's three exam grades as integers. Each record is written into the grades.dat file and dictionary of student data is created in the following format:

gradebook_dict = {'students': [student1dictionary, student2dictionary, ...]}

Each dictionary in the list represents one student and contains the keys 'first_name', 'last_name', 'exam1', 'exam2' and 'exam3', which map to the values representing each student's first name (string), last name (string) and three exam scores (integers).

Finally the grades.dat file is used to display the data in tabular format, including an additional column showing each student's average to the right of that student's three exam grades and an additional row showing the class average on each exam below that exam's column. Fill the blank spaces **1**, **2**, **3, 4, 5, 6, 7,** and **8** in this code.

```
import pickle as p

gradebook_dict = {'students' : []}

with open(----------1) as grades:
    print("Enter a student's first name, last name and  three grades, separated by spaces. Enter -1 to end input ")
    line = input('?')

    while line != ----------2:
        line = line.----------3
        student_dict = {'first_name': line[0], 'last_name': line[1], 'exam1': int(line[2]), 'exam2':
        int(line[3]), 'exam3': int(line[4])}
        ----------4.append(student_dict)
        line = input('?')

    p.----------5

with open(----------6) as grades:
    gradebook_dict = p. ----------7

print(f'{"First":<10}{"Last":<10}{"Test1":>5}  {"Test2":>5}  {"Test3":>5}  {"Average"}')
test1_total = 0
test2_total = 0
test3_total = 0

for student in ----------8:
    test1_total += student['exam1']
    test2_total += student['exam2']
    test3_total += student['exam3']
    average = (student['exam1'] + student['exam2'] + student['exam3']) / 3
    print(f"{student['first_name']:<10}{student['last_name']:<10}" +
        f"{student['exam1']:>5}  {student['exam2']:>5}  " +
        f"{student['exam3']:>5}  {average:>7.2f}")

print(f'{"Test Averages":<20}{test1_total/3:>5.2f}  {test2_total/3:>5.2f}  {test2_total/3:>5.2f}')
```

(24P) QUESTION 3:

Below program creates a simple trivia game for two players. The program works like this:

• Starting with player 1, each player gets a turn at answering 2 trivia questions. (There should be a total of 4 questions.) When a question is displayed, 4 possible answers are also displayed. Only one of the answers is correct, and if the player selects the correct answer, he or she earns a point.

• After answers have been selected for all the questions, the program displays the number of points earned by each player and declares the player with the highest number of points the winner.

First a Question class is written to hold the data for a trivia question. The Question class has attributes for the following data:

• A trivia question
• Possible answer 1
• Possible answer 2
• Possible answer 3
• Possible answer 4
• The number of the correct answer (1, 2, 3, or 4)

The Question class also has an appropriate _ _init_ _ method, accessors, and mutators.

This program has a list containing 4 Question objects, one for each trivia question. Fill the blank spaces **1**, **2**, **3, 4, 5, 6, 7, 8, 9, 10, 11** and **12** in this code.

```
class Question:
    def __init__(self, question, answer1, answer2, answer3, answer4, solution):
        self.__question = question
        self.__answer1 = answer1
        self.__answer2 = answer2
        self.__answer3 = answer3
        self.__answer4 = answer4
        self.__solution = solution

    def set_question(self, question):
        self.__question = question

    def set_answer1(self, answer1):
        self.__answer1 = answer1

    def set_answer2(self, answer2):
        self.__answer2 = answer2

    def set_answer3(self, answer3):
        self.__answer3 = answer3

    def set_answer4(self, answer4):
        self.__answer4 = answer4

    def set_solution(self, solution):
        self.__solution = solution

    def get_question(self):
        return self.__question

    def get_answer1(self):
        return self.__answer1

    def get_answer2(self):
        return self.__answer2

    def get_answer3(self):
        return self.__answer3

    def get_answer4(self):
        return self.__answer4
```

```
    def get_solution(self):
        return self.__solution

    def __str__(self):
        result = self.get_question() + '\n' + '1. ' + self.get_answer1() + '\n' + '2. ' + self.get_answer2()
        + '\n' + '3. ' + self.get_answer3() + '\n' + '4. ' + self.get_answer4()
        return result

    def isCorrect(self, answer):
        return answer == ----------------------------------------------------1

import question
def main():
    first_points = 0
    second_points = 0
    player = ''

    # Create question list.
    questions = ----------------------------------------------------------------2

    for i in range(10):
        if i % 2 == 0:
            player = 'first'
        else:
            player = 'second'
        print('Question for the', player, 'player:')

        current = questions[i]
        print(current)

        user_answer = int(input('Enter your solution (a number' + ' between 1 and 4): '))
        if current.--------------------------------------------------------------3:
            if player == 'first':
                first_points += 1
            else:
                second_points += 1
            print('That is the correct answer.')
            print()
        else:
            print('That is incorrect. The correct answer is', ------------------------------------4)
            print()

    print('The first player earned', first_points, 'points.')
    print('The second player earned', second_points, 'points.')
    if first_points == second_points:
        print('It is a tie.')
    elif first_points > second_points:
        print('The first player wins the game.')
    else:
        print('The second player wins the game.')

def get_questions():
    questions = []

    # Create questions and add to list.
    question1 = question.Question('What is the largest planet?', 'Mars', 'Jupiter', 'Earth', 'Pluto', 2)
    ------------------------------5 .append(-----------------------------------6)
    question2 = question.Question('What is the hottest planet?', 'Mars', 'Pluto', 'Earth', 'Venus', 4)
    ------------------------------7 .append(-----------------------------------8)
    question3 = question.Question('What is the largest type of penguins?', 'Chinstrap',  'Macaroni',
    'Emperor', 'White-flippered', 3)
    ------------------------------9 .append(-----------------------------------10)
    question4 = question.Question('How long is a year on Mars?', '550 Earth days', '498 Earth days',
    '126 Earth days', '687 Earth days', 4)
    ------------------------------11 .append(-----------------------------------12)

    return questions
main()
```

(22P) QUESTION 4:

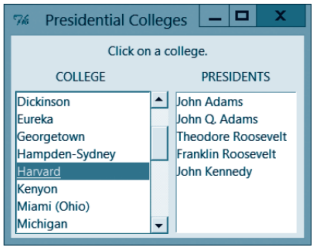
Below program uses the file **PresColl.txt** that contains the names of U.S. presidents and the undergraduate college attended by each of them. The presidents are listed in the order they served. The first three lines of the file are;

George Washington,No college

John Adams,Harvard

Thomas Jefferson,William and Mary

Program fills a list box with the colleges (in alphabetical order) attended by U.S. presidents and then displays the presidents who attended that college when the user clicks on a college in the list box. The output of this program is given below. Fill the blank spaces **1, 2, 3, 4, 5, 6, 7, 8, 9, 10** and **11** in this code.




```
from tkinter import *
class PresColleges:
        def __init__ (self):
        window = Tk()
        window.title("Presidential Colleges")
        instruction = "Click on a college."
        Label(window, text=instruction).grid(row=0, column=0, columnspan=3, pady=5)
        Label(window, text="COLLEGE", width=14).grid(row=1, column=0)
        Label(window, text="PRESIDENTS").grid(row=1, column=2)
        yscroll = Scrollbar(window, orient=VERTICAL)
        yscroll.grid(row=2, column=1, pady=5, sticky=NS)
        infile = open("PresColl.txt", 'r')
        collegeSet = {line.split(',')[1].rstrip() for line in infile}
        infile.close()
        collegeList = list(collegeSet)
        collegeList.sort()
        conOFlstColleges = StringVar()
        ----------1 = Listbox(window, width=20, height=8, listvariable=----------2,
        yscrollcommand=----------3)
        ----------4.grid(row=2, column=0, padx=(5,0), pady=5, sticky=E)
        ----------5.bind("<<ListboxSelect>>", self.presidents)
        ----------6.set(tuple(collegeList))
        self._conOFlstPresidents = StringVar()
        ----------7 = Listbox(window, width=18, height=8, listvariable=----------8)
        self._lstPresidents.grid(row=2, column=2, padx=8, pady=5, sticky=N)
        ----------9 = self._lstColleges.yview
        window.mainloop()

    def presidents(self, e):
        self.L = []
        college = ----------10.get(self._lstColleges.curselection())
        for line in open("PresColl.txt", 'r'):
            temp = line.split(',')
        if temp[1].rstrip() == college:
            self.L.append(temp[0])
        ----------11.set(tuple(self.L))

PresColleges()
```