American University of Beirut
Department of Electrical and Computer Engineering

EECE 230 Introduction to Computing and Programming
Sections: 11, and 12
Spring 2019-20, Final May 2, 2020
9:00 AM – 12:00 PM

Course Instructors: Prof. Hazem Hajj and Zaher Kanafani

Instructions – Please read carefully before you start:

Time Management and Submission Requirement:

- The duration of the quiz is 3 hours, including submission time.
- Plan your time wisely. Do not spend too much time on any one problem. Read through all of them first and attack them in the order that allows you to make the most progress.
- Please keep in mind that you need around 10 minutes, before the end of the time allowed, to submit your solution. It is your responsibility to make sure your file is correctly submitted.
- **You are asked to submit a single zip file containing your Python files (ending with .py extension). Include your name as a comment in your python file.**
- You can use all the material on moodle (lecture slides, source code, programming, previouses, assignments, and solutions).

Code of Conduct:

- Please make sure you type the code of honor statement in each of the problem solutions.
- You are NOT allowed to use the internet for Google search.
- You are NOT allowed to use external help.
- Cell phones and any other communication devices are absolutely NOT allowed during the final. They should be turned off and put away.
- If you get caught violating the above rules or if you communicate with a person other than the exam proctors during the exam, you will immediately get zero and you will be referred to the appropriate disciplinary committee.

Grade Distribution

- The exam consists of 6 problems for 135 points
- The problems are of varying difficulty. Below is a rough ordering estimate of the problems in order of increasing difficulty.
    - Level 0 (give away) (5 points): 1 (Honor statement)
    - Level 1 (simple) (85 points): problems 2, 3, 4a, 6
    - Level 2 (medium difficulty) (45 points): 4b and 5
- Detailed comments are worth partial credit.

Good luck! **and do not upload a wrong file**.

1. **(5 points) Code of Honor statement – To be included in every problem!**
   Please type the following statement as comments at the beginning of **each** of your python program file:
   "I, <Write your name>, promise to conduct the final and this problem on my own without external help."

2. **(20 points) (Simple) Find the four smallest perfect numbers.**
   a. Write a function ***perfect(n)*** to check if a number n is perfect or not. A perfect number has to be >1 and has the sum of its divisors equal to it. The sum has to include 1 and exclude itself. For example 28 is a perfect number because 28 divisors are: 1, 2, 4, 7, 14 and 1+2+4+7+14 = 28
   b. Use this function in a main program to test it and output the first four perfect numbers.

   A sample run is shown below

```
The first four perfect numbers are:
6
28
496
8128
```

3. **(20 points) (Simple) Drawing and Saving a triangle of height specified by user.**
   Write a program that inputs an integer n and outputs a triangle of height n as shown below. If n>0, the triangle is up, if n<0, the triangle is down. Test your program with n=5 and n= -5. Output on the console and also save the outputs to a file called out_triangle.txt

   Sample run (should look the same in console and text file):

```
Input shape length n: 5
    *
   * *
  * * *
 * * * *
* * * * *

Input shape length n: -5
* * * * *
 * * * *
  * * *
   * *
    *
```

4. **(30 points) (Medium) Find the largest sequence of positive numbers**
   Write a program that takes as input a sequence of integers on one line. The purpose is to find the largest sequence of positive numbers. If there are more than one, we are satisfied with one of them. Solve it using:
   a. Two loops
   b. One loop

   Sample runs:

```
Enter integers spearated by spaces: -2 -2 -3
Negative Numbers found
Max-Count =  0
```

```
Enter integers spearated by spaces: -3 4 -3 -3 2
Negative Numbers found
Max-Count =  1
A max-Count subsequence:
4
```

```
Enter integers spearated by spaces: 3 4 -4 4 4 4 4 1 2 -3 -4 4 5
Negative Numbers found
Max-Count =  6
A max-Count subsequence:
4 4 4 4 1 2
```

```
Enter integers spearated by spaces: 4 5 6 1
No Negative Numbers, the Max-Count is the whole List
Max-Count =  4
A max-Count subsequence:
4 5 6 1
```

5. **(30 points) (Medium) Recursion with Palindromes of Vectors and Matrices**

In this problem, we want to write recursive functions to test whether vectors and matrices have palindromes. A palindrome is a sequence of characters or numbers that looks the same forwards and backwards. We define a Palindrome matrix as a matrix whose rows and columns are both palindrome sequences. To get full grade for parts a and b, you need to solve the problem parts using recursive functions as specified. If part a or b is solved without recursion, or directly using slicing, that part will get partial credit.
Examples of palindrome matrices:

```
A = [1 2 1]

B = [1 2 1
     2 3 2
     1 2 1]

C = [1 2 3 2 1
     2 3 4 3 2
```

```
3  4  5  4  3
2  3  4  3  2
1  2  3  2  1]
```

a. (10 pts) Write a **recursive function *my_pal_Vector(V)*** that takes as **input a vector** V and **returns a (Boolean) number:** 1 (True) if the vector is a palindrome and 0 (False) if the vector is not a Palindrome. You are not allowed to use loops or slicing.
Hint: In general for a vector V of N elements, V is a palindrome if the first element and last element of V are equal, and the internal sequence V(2:N-1) is **also** a palindrome.

b. (15 pts) Write a **recursive function *my_pal_Matrix_R(A)*** that takes as **input a matrix A** and **returns a (Boolean) number:** 1 if **all the rows** of A are palindromes, and 0 if any of the rows is not a Palindrome.
Hint: In general for a Matrix A of M rows, A is a palindrome in rows if the first row is a Palindrome and the remaining Matrix A(2:M,:) is **also** a palindrome.

c. (5 pts) Use the above functions in a main program to test them for the following cases: Stick to the output to get full grade on this part.

```
Vectors:
[1, 2, 3, 2, 1] True
[1, 2, 3, 3, 2, 1] True
[1, 2, 3, 5, 1] False
[1, 2, 3, 4, 2, 1] False

Matrices:
[[1, 2, 1], [2, 3, 2], [1, 2, 1]] True
[[1, 2, 3, 2, 1], [2, 3, 4, 3, 2], [3, 4, 5, 4, 3], [2, 3, 4, 3, 2], [1, 2, 3, 2, 1]] True
[[1, 2, 1], [2, 3, 4], [1, 1, 1]] False
[[1, 2, 3, 2, 1], [2, 3, 44, 3, 2], [3, 4, 5, 4, 3], [2, 3, 4, 3, 2], [1, 2, 3, 2, 1]] False
```

6. (30 points) (Simple) Abstract Data Types (ADTs) for Points and Circles
In this problem, we want to design a data type ***Point*** for representing points in cartesian (x,y) coordinates. We then want to use the Point ADT to design a data type ***Circle*** to represents a circle represented by the point coordinates of its center and its radius.

a. (8 points) Design the ADT class ***Point*** which is represented by two floats or ints x and y. Include the needed methods to support the following:

- **Initializing or instantiating an object of type Point** as follows: P = Point(a,b). The method takes the arguments a and b, then makes sure they are of type floats or integers. Include an assertion with message “Bad Point Data” to make sure the inputs are of types either integers or floats. The method then sets the caretsian x and y values of a and b respectively.

- **Converting or casting a Point object to a string** when the str() function is applied to the object of type Point. The method should cast the point object P with coordinates a and b into into the string 'Point (a,b)'. See example runs below.
- Implementing the **method distance()** which is used to compute the distance from a Point object P = Point(a,b) to a point object Q = Point(c,d) using the following syntax: P.distance(Q). Include an assertion with message "Bad Distance Input" to make sure that Q is of type Point. The method should return the distance between P and Q rounded to 2 decimal digits. See example run below.

b. (15 points) Design the ADT class ***Circle, which has a center P*** **represented as as Point** ADT class object. The Circle also includes a radius rad, which can be of type float or integer. Include the needed methods to support the following:
   - **Initializing or instantiating an object of type Circle** as follows: C = Circle(p, r), where p is an object of type Point and r is a float or integer representing the radius of the circle. The method takes the arguments **p and r as input arguments.** Include assertion with message "Bad Circle Input!" to make sure the following holds: type of p is Point, type of r is int or float.
   - **Converting or casting a Circle object to a string** when the str() function is applied to the object of type circle. The method should cast the circle object C with center p = Point (a,b) and radius r into the string 'Circle (Center = Point (a,b), Radius = r)'. See example runs below.
   - **Overload the method __add__() to support the addition '+' of two Circle objects.** The method will return a Circle object, whose radius is the sum of the two Circle objects' radii, and the new center coordinates are produced by taking the average of the coordinates of the two circles' centers.
   - **Implementing a method contains(),** which checks if a given point Point object q = Point(c,d) is inside the circle object C using the following syntax: C.contain(q). Include assertion with message "Bad Contain Input!" to make sure the following holds: type of q is of type Point.This method should return a Boolean True or False indicating whether the point q lies within the circle C.

     import pi from math if needed.

c. (7 points) Using the above classes, write the code in the main program to run the following tests:
   - Declare two points with Coordinates (3.5,6) and (2,4) print them.
   - Print the distance between the two created points
   - Declare a circle C1 with center (2,4) and radius 2.3, and print it.
   - Check if the points (1.5,3) and (10.5, 3) are contained in the created Circle C1.
   - Declare another circle C2 with center (3.5,6) and radius 6.2, and print it
   - Print the sum of the two circles C1+C2

   Here is a sample run and make sure you follow the format shown below.

```
Point (3.5,6) Point (2,4)
Distance between them is: 2.5
C1: Circle ( Center = Point (2,4), Radius = 2.3)
Checking Contains for Points Point(1.5,3) and Point(10.5,3)
True
False
C2: Circle ( Center = Point (3.5,6), Radius = 6.2)
Adding the two Circles, C1+C2, we get:
Circle ( Center = Point (2.75,5.0), Radius = 8.5)
```