Dear students, please be careful about the cheating. Your codes will be evaluated and compared by using cheating program. If the cheating is detected in your codes, it will be added -100 point to your total point of your homeworks. You will submit your homeworks on google classroom. Google classroom code will be announced this week. Please use your own algorithm.

Programming Assignment #1

QUESTION I (10 pts):

The objective of this question is to generate and read files that contain a list of random numbers. Write a function that generates a file with following parameters:

def fillFile(fileSize, fileName):

The function should be called to generate files in the following sizes:

fileSizes = [1000, 5000, 10000, 25000, 50000, 100000, 200000]

The generated files can have names file1000, file5000, file10000, file25000, file50000, file100000, file200000. Each file should have serial numbers from 0 to size of file. By using one tenth of file shuffle numbers among themselves. For example, if your file 100 numbers, choose 10 numbers randomly and shuffle these numbers.
You can use random.randint(0,fileSize+1000) to generate a random number.

Write another function that reads numbers inside of a file and returns a list.

def readFile(fileName):

The returned list contains the numbers stored in the file.

At each call of fillFile and readFile, record the times before and after the function call as in following example (import time) :

```
start  = time.time()
fillFile(fileSize, "file" + str(fileSize))
finish = time.time()
runTime = finish – start
```

Record the run times of functions into a file named "fileStats.txt". The "**fileStats.txt**" file should contain entries for run times as follows:

fillFile n1, n2, n3, n4, n5, n6, n7
readFile n1, n2, n3, n4, n5, n6, n7

where n1, n2, n3, n4, n5, n6 and n7 are execution times for file sizes 1000, 5000, 10000, 25000, 50000, 100000, 200000.

Submit your program code and filleStats.txt in ZIP file together with the solutions of other programs.

QUESTION II (30 pts):

Perform a benchmark analysis of the following sorting algorithms:

- Bubble Sort
- Selection Sort
- Insertion Sort
- Shell Sort
- Merge Sort
- Quick Sort
- Heap sort

You can get source codes of sorting programs from lectures notes. By using the files that you generated in Question I, read each file and sort file by using each of sorting algorithms. Record the execution time of each sorting algorithm into a file named "sortStats.txt". The "sortStats.txt" file should contain entries for execution times as follows:

Bubble_Sort n1, n2, n3, n4, n5, n6, n7
Selection_Sort n1, n2, n3, n4, n5, n6, n7
Insertion_Sort n1, n2, n3, n4, n5, n6, n7
Shell_Sort n1, n2, n3, n4, n5, n6, n7
Merge_Sort n1, n2, n3, n4, n5, n6, n7
Quick_Sort n1, n2, n3, n4, n5, n6, n7
Heap_Sort n1, n2, n3, n4, n5, n6, n7

where n1, n2, n3, n4, n5, n6 and n7 are execution times for file sizes 1000, 5000, 10000, 25000, 50000, 100000, 20000.

Submit your program code and sortStats.txt in ZIP file together with the solutions of other programs

QUESTION III (15 pts):

Perform a benchmark analysis of the following searching methods:

- Linear (Sequential) Search
- Binary Search
- Hashing

Read the file "file100000" into a list. Sort the file using the one of fast sorting algorithms. You can use orderedSequentialSearch, binarySearch, and HashTable code given in https://runestone.academy/runestone/books/published/pythonds/SortSearch/toctree.html.

Fill the hash table with key values stored in the file100000. The data values mapping to the key values can be assigned to a string like "Data" + str(key). Make the hash table size 10% bigger. For example you can set self.size = 110017 which is a prime number.

Generate a list of 1000 random numbers using **random.randint(0,100000).** By using the searching methods, make a search of these random numbers. Record their total execution times into a file named "searchStats.txt". The "searchStats.txt" file should contain entries for execution times as follows:

Linear_Search n

Binary_Search n
Hashing n

where n is the total execution time for searching these 500 numbers.

Submit your program code and searchStats.txt in ZIP file together with the solutions of other programs

QUESTION IV (15 pts):

1
1 2 1
1 2 4 2 1
1 2 4 8 4 2 1
1 2 4 8 16 8 4 2 1
1 2 4 8 16 32 16 8 4 2 1
1 2 4 8 16 32 64 32 16 8 4 2 1
1 2 4 8 16 32 64 128 64 32 16 8 4 2 1
1 2 4 8 16 32 64 128 256 128 64 32 16 8 4 2 1

By using nested loops, write the code of this triangularshape.

QUESTION V (30 pts):

You will have an orthogonal triangle input from a file and you need to find the maximum sum of the numbers according to given rules below; 1. You will start from the top and move downwards to an adjacent number as in below. 2. You are only allowed to walk downwards and diagonally. 3. You can only walk over NON PRIME NUMBERS. 4. You have to reach at the end of the pyramid as much as possible. 5. You have to treat the input as pyramid.

215
193 124
117 237 442
218 935 347 235
320 804 522 417 345
229 601 723 835 133 124
248 202 277 433 207 263 257
359 464 504 528 516 716 871 182
461 441 426 656 863 560 380 171 923
381 348 573 533 447 632 387 176 975 449
223 711 445 645 245 543 931 532 937 541 444
330 131 333 928 377 733 017 778 839 168 197 197
131 171 522 137 217 224 291 413 528 520 227 229 928
223 626 034 683 839 053 627 310 713 999 629 817 410 121
924 622 911 233 325 139 721 218 253 223 107 233 230 124 233

1. You will start from the top and move downwards to an adjacent number as in below.
2. You are only allowed to walk downwards and diagonally.
3. You can only walk over NON PRIME NUMBERS.
4. You have to reach at the end of the pyramid as much as possible.
5. You have to treat the input as pyramid.

By using these numbers and rules, find the maximum sum and numbers which you use.

SUBMISSION INSRUCTIONS

File Name: COE-64160099-KAYA-A1.py or in zip file COE-64160099-KAYA-A1.zip

```
################################################################################################
#  Name:         Ali Cokcalışır
#  Student ID:   6321211
#  Department:   Computer Engineering
#
#  Assignment ID: A1
################################################################################################

################################################################################################
#  QUESTION I
#  Description:
#  Body mass index (BMI) is a measure of health #  based on weight. It can be calculated by
#  taking your weight in kilograms and # dividing it by the square of your height in meters.
#  Write a program that prompts # the user to enter a weight in pounds and height in inches
#  and displays the BMI. Note that one pound is 0.45359237 kilograms and one inch is
#  0.0254 meters.
#
#  Sources:
#  Give references for the sources that you used in your program if there are any
#
################################################################################################

print("\n")
print("SOLUTION OF QUESTION I:")
print("*********************************************************************************")
# Prompt the user to enter weight in pounds
weight = eval(input("Enter weight in pounds: "))

# Prompt the user to enter height in inches
height = eval(input("Enter height in inches: "))

KILOGRAMS_PER_POUND = 0.45359237  # Constant
METERS_PER_INCH = 0.0254  # Constant

# Compute BMI
weightInKilograms = weight * KILOGRAMS_PER_POUND
heightInMeters = height * METERS_PER_INCH
bmi = weightInKilograms / (heightInMeters * heightInMeters)

# Display result
print("BMI is", format(bmi, ".2f"))
if bmi < 18.5:
    print("Underweight")
elif bmi < 25:
    print("Normal")
elif bmi < 30:
    print("Overweight")
else:
    print("Obese")


################################################################################################
#  QUESTION II
#  Description:
#  You can use the math functions to solve many computational problems. Given the three
#  vertices of a triangle, for example, you can compute the angles by using
#  math formulas. The following program asks user to enter the coordinates of a triangle,
# then it computes its angles.
#
#  Sources:
#  Give references for the sources that you used in your program if there are any
```

```
#
#################################################################################################

print("\n")
print("SOLUTION OF QUESTION II:")
print("*********************************************************************************")
import math

x1, y1, x2, y2, x3, y3 = eval(input("Enter six coordinates of three points \
separated by commas like x1, y1, x2, y2, x3, y3: "))

a = math.sqrt((x2 - x3) * (x2 - x3) + (y2 - y3) * (y2 - y3))
b = math.sqrt((x1 - x3) * (x1 - x3) + (y1 - y3) * (y1 - y3))
c = math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2))

A = math.degrees(math.acos((a * a - b * b - c * c) / (-2 * b * c)))
B = math.degrees(math.acos((b * b - a * a - c * c) / (-2 * a * c)))
C = math.degrees(math.acos((c * c - b * b - a * a) / (-2 * a * b)))

print("The three angles are ", round(A * 100) / 100.0,
      round(B * 100) / 100.0, round(C * 100) / 100.0)
```

Note:

Submit all of your solutions within a ZIP or RAR file.