İbrahim Talha ASAN
COE-64170019

Assignment 3 Report

Problem 1:

Definition: The purpose of Problem 1 is to write the function that can convert the decimal number received as a string from the user to hexadecimal number. In addition, if the user makes a mistake (such as entering a rational number), to show a warning message.

Program Code:

```
def decToHex(n):
  hexaDecimalNumber = ['0'] * 50
  i = 0
  try:
    while (int(n) != 0):
      temp = 0
      temp = int(n) % 16
      if (temp < 10):
        hexaDecimalNumber[i] = chr(temp + 48)
        i = i + 1
      else:
        hexaDecimalNumber[i] = chr(temp + 55)
        i = i + 1
      n = int(int(n) / 16)
    j = i - 1
    while (j >= 0):
      print((hexaDecimalNumber[j]), end="")
      j = j - 1
  except ValueError:
    print('The variable is not a integer.')

number = input("Enter a integer value :")
decToHex(number)
```

Program Outputs:

- Enter a integer value :>? 1254
  4E6
- Enter a integer value :>? 55,4
  The number is not a integer value.

Discussions: There is no problem solving Problem 1.

İbrahim Talha ASAN
COE-64170019

Problem 2:

Definition: The purpose of Problem 2 is to write a messaging application consisting of one party. For example, an application where Ali can send randomly generated messages to his friend Veli and communicate. Also this message application must have a server as a control mechanism at the back and in the question, this system is requested to be installed and run.

Program Code:

```
class Ali:
    def __init__(self,messageSent):
        self.messageSent=messageSent

    def sendMessage(self):
        print("Sending message from Ali : " + str(self.messageSent))
class Veli:
    def __init__(self,isMessageConvenient,messageReceived):
        self.messageReceived = messageReceived
        self.isMessageConvenient = isMessageConvenient

    def showDisplay(self):
        if self.isMessageConvenient==True:
            print("Message received from Veli : " + str(self.messageReceived))
            print("Message has been read from Veli.")
            print("Veli's answer is :" + " I received your " + str(self.messageReceived))
            self.messageReceived = ""
        else:
            print("Message is inconvenient.")
class Server:
    def __init__(self,messageSent):
        if messageSent == "":
            self.isMessageConvenient = False
        else:
            self.isMessageConvenient = True
        self.messageSending = messageSent

    def isMessageConveniently(self):
        print("Message Conveniently Situation : " + str(self.isMessageConvenient))
import random
numberOfMessages = random.randint(0,10)
for i in range(numberOfMessages):
    inputMessage = random.choice(["Message " + str(i+1)])
    aliMessage = Ali(inputMessage)
    aliMessage.sendMessage()

    serverMessage = Server(aliMessage.messageSent)
    serverMessage.isMessageConveniently()

    veliMessage = Veli(serverMessage.isMessageConvenient,serverMessage.messageSending)
    veliMessage.showDisplay()
    print()
```

İbrahim Talha ASAN
COE-64170019

Program Outputs:

Sending message from Ali : Message 1

Message Conveniently Situation : True

Message received from Veli : Message 1

Message has been read from Veli.

Veli's answer is : I received your Message 1

Sending message from Ali : Message 2

Message Conveniently Situation : True

Message received from Veli : Message 2

Message has been read from Veli.

Veli's answer is : I received your Message 2

Sending message from Ali : Message 3

Message Conveniently Situation : True

Message received from Veli : Message 3

Message has been read from Veli.

Veli's answer is : I received your Message 3

Sending message from Ali : Message 4

Message Conveniently Situation : True

Message received from Veli : Message 4

Message has been read from Veli.

Veli's answer is : I received your Message 4

Sending message from Ali : Message 5

Message Conveniently Situation : True

Message received from Veli : Message 5

Message has been read from Veli.

Veli's answer is : I received your Message 5

Discussions: There is no problem solving Problem 2.

İbrahim Talha ASAN
COE-64170019

Problem 3:

Definition: The goal of Problem 3 is to test and evaluate the execution time of the bucket order by generating the n and k values randomly 10 times. In addition, it is to show that the average complexity value of the bucket order is O (n + k).

Program Code:

```
import random
import time
import matplotlib.pyplot as plt

def insertion_sort(bucket):
    for i in range(1, len(bucket)):
        var = bucket[i]
        j = i - 1
        while (j >= 0 and var < bucket[j]):
            bucket[j + 1] = bucket[j]
            j = j - 1
        bucket[j + 1] = var


def bucket_sort(input_list): # Find maximum value in the list and use length of the list to determine
                                    #which value in the list goes into which bucket
    max_value = max(input_list)
    size = max_value / len(input_list)

    # Create n empty buckets where n is equal to the length of the input list
    buckets_list = []
    for x in range(len(input_list)):
        buckets_list.append([])

        # Put list elements into different buckets based on the size
    for i in range(len(input_list)):
        j = int(input_list[i] / size)
        if j != len(input_list):
            buckets_list[j].append(input_list[i])
        else:
            buckets_list[len(input_list) - 1].append(input_list[i])

    # Sort elements within the buckets using Insertion Sort
    for z in range(len(input_list)):
        insertion_sort(buckets_list[z])

    # Concatenate buckets with sorted elements into a single list
    final_output = []
    for x in range(len(input_list)):
        final_output = final_output + buckets_list[x]
    return final_output
```

```
runTimes=[]
totalnk=[]
for i in range(10):
   # array's number of elements, describe as n and random value 2 and 10.
   n = random.randint(2, 10)
   array = []
   for a in range(n):
      # array's values range as k and array's value range is randomly 0 and 1000.
      k = random.randint(0, 100)
      array.append(k - 1)
   begin = time.time()
   print('Original List:')
   print(array)
   sorted_list = bucket_sort(array)
   print('Sorted List:')
   print(sorted_list)
   time.sleep(1)
   end = time.time()
   print(f"Total runtime of the program is {end - begin}")
   print()
   runTimes.append(n+k)
   totalnk.append(end-begin)

plt.figure()
plt.plot(runTimes, totalnk,'o', color='black')
 # naming the x axis
plt.xlabel('Time (s)')
 # naming the y axis
plt.ylabel('n+k')
 # giving a title to a graph
plt.title('Comparing Run Time')
plt.show()
```

Program Outputs:

```
Original List:
[11, 84, 15]
Sorted List:
[11, 15, 84]
Total runtime of the program is 1.001359224319458
n+k value is : 19
time value is : 1.001359224319458

Original List:
[3, 81, 65, 75]
Sorted List:
[3, 65, 75, 81]
Total runtime of the program is 1.0023832321166992
n+k value is : 80
time value is : 1.0023832321166992
```

İbrahim Talha ASAN
COE-64170019

Original List:
[10, 77, 72, 55, 40]
Sorted List:
[10, 40, 55, 72, 77]
Total runtime of the program is 1.0011954307556152
n+k value is : 46
time value is : 1.0011954307556152

Original List:
[13, 24, 99, 65, 36, 37, 74, 0, 67, 60]
Sorted List:
[0, 13, 24, 36, 37, 60, 65, 67, 74, 99]
Total runtime of the program is 1.001387357711792
n+k value is : 71
time value is : 1.001387357711792

Original List:
[99, 50]
Sorted List:
[50, 99]
Total runtime of the program is 1.0024502277374268
n+k value is : 53
time value is : 1.0024502277374268

Original List:
[16, 3, 51, 90, 0, 82, 54, 33, 12, 30]
Sorted List:
[0, 3, 12, 16, 30, 33, 51, 54, 82, 90]
Total runtime of the program is 1.0007133483886719
n+k value is : 41
time value is : 1.0007133483886719

Original List:
[73, 55, 83, 65, 72, 98]
Sorted List:
[55, 65, 72, 73, 83, 98]
Total runtime of the program is 1.0029630661010742
n+k value is : 105
time value is : 1.0029630661010742

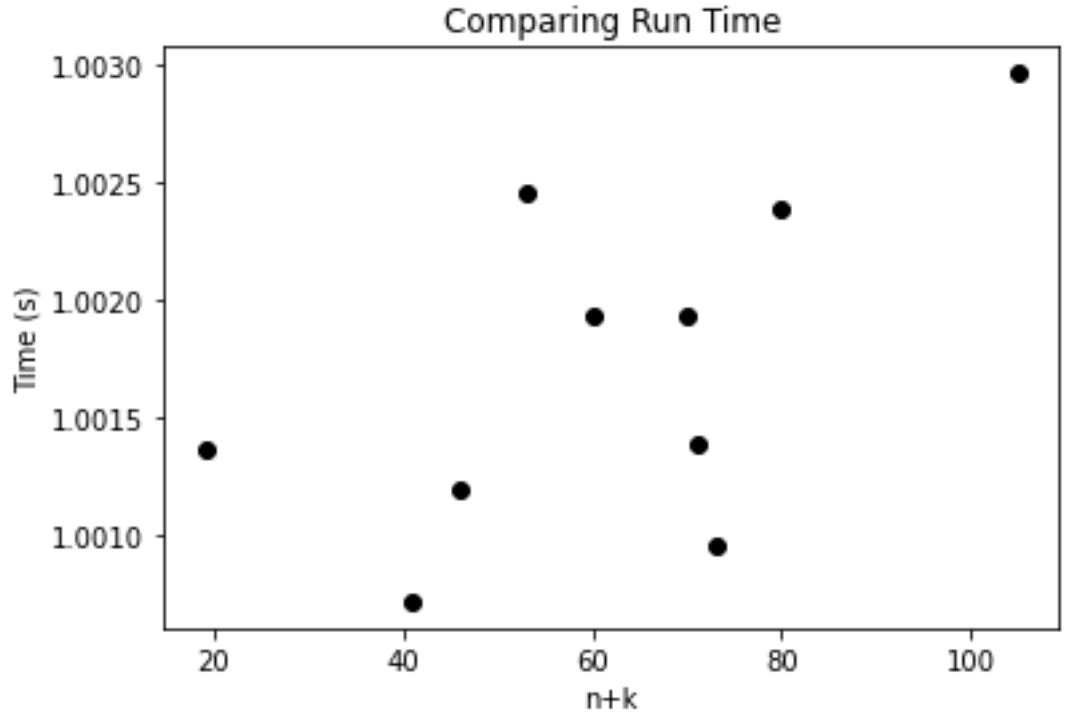
Original List:
[70, 97, 94, 68, 71, 39, 46, 58, 60]
Sorted List:
[39, 46, 58, 60, 68, 70, 71, 94, 97]
Total runtime of the program is 1.0019266605377197
n+k value is : 70
time value is : 1.0019266605377197

İbrahim Talha ASAN
COE-64170019

Original List:
[37, 48, 84, 78, 63, 66]
Sorted List:
[37, 48, 63, 66, 78, 84]
Total runtime of the program is 1.0009498596191406
n+k value is : 73
time value is : 1.0009498596191406

Original List:
[91, 46, 81, 96, 93, 32, 72, 51]
Sorted List:
[32, 46, 51, 72, 81, 91, 93, 96]
Total runtime of the program is 1.0019264221191406
n+k value is : 60
time value is : 1.0019264221191406

Discussions:



- Create an empty array.(bucket)
- Loop through the original array and put each object in a bucket.
- Sort each of the non-empty buckets
- Check the buckets in order and then put all objects back into the original array.

The average time complexity is O(n+k) where n is the number of your buckets Therefore, complexity becomes better if inside the buckets elements are already sorted.If insertion sort is used to sort elements then overall complexity in average case will be linear O(n+k).