JC1 Promotional Examination 2017

Candidate name: _____
Centre number: **3030**
Index number: _____
Programming language used: _____

QUESTION 1: WORDS

EVIDENCE 1

```
WordFile = open("WORDS1.txt", "r")
WordDict = dict()
HighestNo = 0                          #contains the current highest
no of occurrences
for EachWord in WordFile:
    if EachWord[-1] == "\n":           #not the last line
        WordLine = EachWord[:-1].split(",")
    else:
        WordLine = EachWord.split(",")
    WordDict[WordLine[0]] = int(WordLine[1]) #dictionary entry
([word] ==> number)
    if int(WordLine[1]) > HighestNo:
        HighestNo = int(WordLine[1])

WordFile.close()

WordArray = list(WordDict.keys())        #array containing all terms
NumberArray = list(WordDict.values())    #array containing all the
number of occurrences
for i in range(len(NumberArray)):
    if NumberArray[i] == HighestNo:
        HighestIndex = i
        break

print("The term containing the highest number of occurrences, with
{}, is {}.".format(HighestNo, WordArray[HighestIndex]))
```

EVIDENCE 2

```
== RESTART: C:\Users\USER\Google Drive\codes\promo 2017\task1_dictionary.py ==
The term containing the highest number of occurrences, with 75, is computing.
```

EVIDENCE 3

```
WordFile = open("WORDS2.txt", "r")
WordDict = dict()
HighestNo = 0                              #contains the current high-
est no of occurrences
for EachWord in WordFile:
    WordLine = EachWord[:-1]               #the word line is never the
last line
    NumberLine = WordFile.readline()       #readline again to get the
next line
    if NumberLine[-1] == "\n":            #not the last line
        NumberLine = NumberLine[:-1]
    WordDict[WordLine] = int(NumberLine) #dictionary entry ([word]
==> number)
    if int(NumberLine) > HighestNo:
        HighestNo = int(NumberLine)

WordFile.close()

WordArray = list(WordDict.keys())          #array containing all terms
NumberArray = list(WordDict.values())      #array containg all the
number of occurrences
HighestIndices = []                        #array containing the index
of all the highest occurrences
for i in range(len(NumberArray)):
    if NumberArray[i] == HighestNo:
        HighestIndices.append(i)

print("The terms containing the highest number of occurrences, with
{}, are:".format(HighestNo))
for i in HighestIndices:
    print(WordArray[i])
```

EVIDENCE 4

```
The terms containing the highest number of occurrences, with 86, are:
system
computer
>>>
```

QUESTION 2: PRIME NUMBER

EVIDENCE 5

```
def prime(N):
    flag = True
    for i in range(2, N):       # from 2 to N - 1, or always re-
turns False
        if N % i == 0:          # i is a factor of N
            flag = False
            exit                # exits the for loop as flag is
False
        else:                   # i is not a factor of N
            pass                # do nothing, flag remains the same
    if N == 1:
        flag = False            # 1 is not a prime number

    if flag:
        primeFlag = " "         # if prime, adds a space
    else:
        primeFlag = " not "     # if not prime, inserts word 'not'

    print("{} is{}a prime number.".format(N, primeFlag))
```

EVIDENCE 6

```
======= RESTART: C:\Users\USER\Google Drive\codes\promo 2017\task2.py =======
>>> prime(1)
1 is not a prime number.
>>> prime(2)
2 is a prime number.
>>> prime(13)
13 is a prime number.
>>> prime(77)
77 is not a prime number.
>>>
```

EVIDENCE 7

```
def IsPrime(N):
    flag = True
    for i in range(2, N):        # from 2 to N - 1, or always re-
turns False
        if N % i == 0:           # i is a factor of N
            flag = False
            exit                 # exits the for loop as flag is
False
        else:                    # i is not a factor of N
            pass                 # do nothing, flag remains the same
    if N == 1:
        flag = False             # 1 is not a prime number
    return flag

counter = 0                      # number of prime numbers between 1
and N
N = 0
while counter < 20:
    N += 1
    if IsPrime(N):               # N is prime
        print(N)
        counter += 1             # counter increments
```

EVIDENCE 8

```
======= RESTART: C:\Users\USER\Google Drive\codes\promo 2017\task2.py =======
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
>>>
```

QUESTION 3: COUNTRIES LINKED LIST

EVIDENCE 9

```
class ListNode:
    def __init__(self, Name = "", Pointer = -1):
        self.__Name = Name
        self.__Pointer = Pointer
    def GetName(self):
        return self.__Name
    def SetName(self, NewName):
        self.__Name = NewName
    def GetPointer(self):
        return self.__Pointer
    def SetPointer(self, NewPointer):
        self.__Pointer = NewPointer

class LinkedList:
    def __init__(self, Size = 20):
        self.__Node = [ListNode() for i in range(Size)]
        for i in range(Size - 1):
            self.__Node[i].SetPointer(i + 1)
        self.__Start = -1
        self.__NextFree = 0

    def Display(self):
        print("{:^10} | {:^20} | {:^10}".format("Node", "Name",
"Pointer"))
        print("-"*46)
        for i in range(len(self.__Node)):
            print("{:^10} | {:^20} | {:^10}".format(i,
self.__Node[i].GetName(), self.__Node[i].GetPointer()))
        print()
        print("Start =", str(self.__Start))
        print("NextFree =", str(self.__NextFree))

    def IsEmpty(self):
        return self.__Start == -1

    def IsFull(self):
        return self.__NextFree == -1
```

EVIDENCE 10

```
======= RESTART: /Users/Angsuan/Google Drive/cu
>>> LinkedList1 = LinkedList()
>>> LinkedList1.Display()
   Node    |          Name          |  Pointer
-------------------------------------------------
    0      |                        |     1
    1      |                        |     2
    2      |                        |     3
    3      |                        |     4
    4      |                        |     5
    5      |                        |     6
    6      |                        |     7
    7      |                        |     8
    8      |                        |     9
    9      |                        |     10
    10     |                        |     11
    11     |                        |     12
    12     |                        |     13
    13     |                        |     14
    14     |                        |     15
    15     |                        |     16
    16     |                        |     17
    17     |                        |     18
    18     |                        |     19
    19     |                        |     -1

Start = -1
NextFree = 0
>>>
```

EVIDENCE 11

```
    def Insert(self, NewName):
        if self.__NextFree == -1:                                   #no
free nodes
            print("No space to insert.")
            return
        self.__Node[self.__NextFree].SetName(NewName)       #store
in next free node
        if self.__Start == -1:                                      #insert
into empty list
            HoldFree = self.__Node[self.__NextFree].GetPointer()
            self.__Node[self.__NextFree].SetPointer(-1)
            self.__Start = self.__NextFree
            self.__NextFree = HoldFree
        else:
            if NewName < self.__Node[self.__Start].GetName():   #as
first node of list
                HoldFree = self.__Node[self.__NextFree].Get-
Pointer()
                self.__Node[self.__NextFree].Set-
Pointer(self.__Start)
                self.__Start = self.__NextFree
                self.__NextFree = HoldFree
            else:
                Previous = self.__Start
                Current = self.__Start
                while NewName > self.__Node[Current].GetName() and
self.__Node[Current].GetPointer() != -1:
                    #search position to insert node
                    Previous = Current
                    Current = self.__Node[Current].GetPointer()
                if NewName > self.__Node[Current].GetName() and
self.__Node[Current].GetPointer() == -1:
                    #insert at last node of list
                    HoldFree = self.__Node[self.__NextFree].Get-
Pointer()
                    self.__Node[Current].Set-
Pointer(self.__NextFree)
                    self.__Node[self.__NextFree].SetPointer(-1)
                    self.__NextFree = HoldFree
                else:   #insert in between nodes
                    HoldFree = self.__Node[self.__NextFree].Get-
Pointer()
                    self.__Node[Previous].Set-
Pointer(self.__NextFree)
                    self.__Node[self.__NextFree].SetPointer(Cur-
rent)
                    self.__NextFree = HoldFree
```

EVIDENCE 12

```
CountryFile = open("COUNTRIES.txt", "r")
CountryList = LinkedList()          #new linked list
for Country in CountryFile:
    if Country[-1] == "\n":         #not the last line
        CountryList.Insert(Country[:-1])
    else:                           #the last line
        CountryList.Insert(Country)
CountryFile.close()
CountryList.Display()
```

EVIDENCE 13

```
------- RESTART: /Users/Angsuan/Google Drive/CO
   Node     |          Name          |  Pointer
-------------------------------------------------
     0      |          Qatar         |     7
     1      |          Brazil        |     6
     2      |       New Zealand      |     0
     3      |          Kenya         |     16
     4      |       Timor Leste      |     11
     5      |          Libya         |     17
     6      |          Egypt         |     8
     7      |        Singapore       |     12
     8      |        Finland         |     15
     9      |        Tanzania        |     13
     10     |        Algeria         |     1
     11     |        Uruguay         |     14
     12     |        Spain           |     9
     13     |        Thailand        |     4
     14     |       Uzbekistan       |     -1
     15     |       Kazakhstan       |     3
     16     |          Laos          |     5
     17     |          Mexico        |     2
     18     |                        |     19
     19     |                        |     -1

Start = 10
NextFree = 18
>>>
```

EVIDENCE 14

```
    def Query(self):
        CountryInput = input("Enter a country name: ")
        Previous = self.__Start
        Current = self.__Start
        while CountryInput > self.__Node[Current].GetName():
            #traverse linked list to find node
            Previous = Current
            Current = self.__Node[Current].GetPointer()
        if CountryInput == self.__Node[Current].GetName():  #coun-
try is found
            print("{} is found in the linked list, at position
{}.".format(CountryInput, Current))
        else:
            print("{} is not found in the linked list.".for-
mat(CountryInput))
```

EVIDENCE 15

```
Enter a country name: Laos
Laos is found in the linked list, at position 16.
Enter a country name: China
China is not found in the linked list.
>>>
```

At the end of the examination, save your `EVIDENCE.docx` in pdf with filename `EVIDENCE_yourname.pdf` in your removable storage device.