**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** |
|  |
| **EVIDENCE 3** |
| WordFile = open("WORDS2.txt", "r")  WordDict = dict()  HighestNo = 0 #contains the current highest 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** |
|  |
| **QUESTION 2: PRIME NUMBER** |
| **EVIDENCE 5** |
| def prime(N):  flag = True  for i in range(2, N): # from 2 to N - 1, or always returns 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** |
| task21.PNG |
| **EVIDENCE 7** |
| def IsPrime(N):  flag = True  for i in range(2, N): # from 2 to N - 1, or always returns 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** |
| task22.PNG |
| **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** |
|  |
| **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].GetPointer()  self.\_\_Node[self.\_\_NextFree].SetPointer(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].GetPointer()  self.\_\_Node[Current].SetPointer(self.\_\_NextFree)  self.\_\_Node[self.\_\_NextFree].SetPointer(-1)  self.\_\_NextFree = HoldFree  else: #insert in between nodes  HoldFree = self.\_\_Node[self.\_\_NextFree].GetPointer()  self.\_\_Node[Previous].SetPointer(self.\_\_NextFree)  self.\_\_Node[self.\_\_NextFree].SetPointer(Current)  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** |
|  |
| **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(): #country is found  print("{} is found in the linked list, at position {}.".format(CountryInput, Current))  else:  print("{} is not found in the linked list.".format(CountryInput)) |
| **EVIDENCE 15** |
|  |

**At the end of the examination, save your EVIDENCE.docx** **in pdf with filename EVIDENCE\_yourname.pdf in your removable storage device.**