|  |  |
| --- | --- |
| Candidate name: | Justin Leow |
| Centre number: | Dunman High School |
| Index number: | 7 |
| Programming language used: | Python3.3 |

|  |
| --- |
| **Question 1**  **Evidence 1**  *TASK 1.1 program code* |
| def bubble\_sort(A):  swapped = True # assume not sorted  while swapped:  swapped = False  for i in range(1,len(A)):  if A[i-1][0] < A[i][0]:  A[i-1], A[i] = A[i], A[i-1]  swapped = True  return A  ##main  bonusArr = []  for i in range(5):  enumValidated = False  bonusValidated = False  while not enumValidated:  uinput = input("Enter Employee number:")  enumValidated = True #assume validated    if uinput == "": #presence check  if i == 0:#empty input, does not want to quit  print("You must enter an employee number!")  enumValidated = False  else:#quit  done = True  break    elif len(uinput) != 2:#length check  print("Employee number must be 2 characters in length!")  enumValidated = False    elif uinput[0] != "E":  print("First character of Employee number must be an 'E'!")  enumValidated = False    else:  try:  int(uinput[1])  enum = uinput  except:  print("second character of Employee number must be an integer!")  enumValidated = False    while not bonusValidated:  uinput = input("Enter salary of Employee:")  bonusValidated = True #assume validated    if uinput == "": #presence check  #empty input, does not want to quit  print("You must enter an employee number!")  bonusValidated = False    try:  uinput = int(uinput)    if not 10 <= uinput <= 5000:  print("Salary must be between 10 and 5000 inclusive")  bonusValidated = False  else:  salary = uinput    except:  print("The salary has to be an integer!")  bonusValidated = False    if salary < 1000:  bonus = 0.1 \* salary  elif salary < 3000:  bonus = 0.15 \* salary  else: #salary >= 3000  bonus = 0.25 \* salary    bonus = "{0:.2f}".format(bonus)    employee = [bonus,enum]      bonusArr.append(employee)    bonusArr = bubble\_sort(bonusArr)  with open("BONUS.txt","w") as outfile:  for line in bonusArr:  string = line[1] + " " + str(line[0]) + "\n"  outfile.write(string) |
| **Evidence 2**  *Screenshots + Annotation* |
| Entered empty string. Returned appropriate error  Entered inappriopriate length.  Entered wrong format which returned appriopriate error  Entered inappriopriate format for salary  Entered inappropriate ranges for salary  Entered all employees successfully  Bonus recorded appropriately |
| **Question 2**  **Evidence 3**  *Task 2.1 program code* |
| def Fibonacci(n):  if n == 0:  return 0  if n == 1:  return 1    return Fibonacci(n-1) + Fibonacci(n-2)  ##main  print(Fibonacci(10)) |
| **Evidence 4**  *Screenshot* |
|  |
| **Evidence 5**  *Task 2.2 program code* |
| fibArr = [0,1]  for i in range(100):  fibArr.append("")  def Fibonacci(n):  if n == 0:  return 0  if n == 1:  return 1    try:  newfib = int(fibArr[n-1] + fibArr[n-2])    fibArr[n] = newfib  return fibArr[n]  except:#fibArr[n-1] has not been calculated yet  return Fibonacci(n-1) + Fibonacci(n-2)  ##main  print(Fibonacci(50)) |
| **Evidence 6**  *Screenshot* |
|  |
| **Evidence 7**  *Task 2.3 program code* |
| def Fibonacci(n):  if n == 0:  return 0  if n == 1:  return 1    firstNum = 0  secondNum = 1  for i in range(1,n):  ans = firstNum + secondNum  firstNum = secondNum  secondNum = ans    return ans  ##main  print(Fibonacci(50)) |
| **Evidence 8**  *Screenshot* |
|  |
| **Question 3**  **Evidence 9**  *Task 3.1 program code* |
| def GetCheckDigit(IMEI):  imeiSum = 0  for i in range(14):  if i % 2 == 0:#even  imeiSum += int(IMEI[i])  else: #odd  tempSum = 2 \* int(IMEI[i])  if tempSum >= 10:  tempSum = str(tempSum)  imeiSum += int(tempSum[0]) + int(tempSum[1])  else:  imeiSum += tempSum    checkDigit = imeiSum % 10  if checkDigit == 0:  return 0  else:  return 10 - checkDigit  def IsValidIMEI(IMEI):  if len(IMEI) != 15: #length check  return False  try:  int(IMEI)  except: #check if IMEI is an integer  return False  if not GetCheckDigit(IMEI) == int(IMEI[14]):  return False  else:  return True      ##main  done = False  while not done:  foo = input("Input IMEI:")  if foo == "":  done = True  else:  print(IsValidIMEI(foo)) |
| **Evidence 10**  *Screenshot* |
|  |
| **Evidence 11**  *Task 3.2 program code* |
| def GetCheckDigit(IMEI):  imeiSum = 0  for i in range(14):  if i % 2 == 0:#even  imeiSum += int(IMEI[i])  else: #odd  tempSum = 2 \* int(IMEI[i])  if tempSum >= 10:  tempSum = str(tempSum)  imeiSum += int(tempSum[0]) + int(tempSum[1])  else:  imeiSum += tempSum    checkDigit = imeiSum % 10  if checkDigit == 0:  return 0  else:  return 10 - checkDigit  def IsValidIMEI(IMEI):  if len(IMEI) != 15: #length check  return False  try:  int(IMEI)  except: #check if IMEI is an integer  return False  if not GetCheckDigit(IMEI) == int(IMEI[14]):  return False  else:  return True      ##main  Others = 0  US = 0  Europe = 0  UK = 0  China = 0  India = 0  with open("IMEIS.txt","r") as infile:  line = infile.readline()  while line != "":  if not IsValidIMEI:  print(line + " is not a valid IMEI")  else:  rbi = line[:2]  if rbi == "01" or rbi == "30":  US += 1  elif rbi == "33" or rbi == "45" or 49 <= int(rbi) <= 54:  Europe += 1  elif rbi == "35" or rbi == "44" or rbi == "98":  UK += 1  elif rbi == "86":  China += 1  elif rbi == "91":  India += 1  else:  Others += 1  #determine origin of mobile device, and increase the counter of the respective distribution    line = infile.readline()  print("Others: {0:>2}%".format(Others))  print("United States: {0:>2}%".format(US))  print("Europe: {0:>2}%".format(Europe))  print("United Kingdom: {0:>2}%".format(UK))  print("China: {0:>2}%".format(China))  print("India: {0:>2}%".format(India)) |
| **Evidence 12**  *Screenshot* |
|  |
| **Evidence 13**  *Task 3.3 program code* |
| def GetCheckDigit(IMEI):  imeiSum = 0  for i in range(14):  if i % 2 == 0:#even  imeiSum += int(IMEI[i])  else: #odd  tempSum = 2 \* int(IMEI[i])  if tempSum >= 10:  tempSum = str(tempSum)  imeiSum += int(tempSum[0]) + int(tempSum[1])  else:  imeiSum += tempSum    checkDigit = imeiSum % 10  if checkDigit == 0:  return 0  else:  return 10 - checkDigit  def GenerateIMEI():  with open("CHINA.txt","w") as outfile:  #generate IMEIs with model number 123456  currSerial = 1  for i in range(20000):  currSerialStr = str(currSerial)  while len(currSerialStr) < 6:  currSerialStr = "0" + currSerialStr  currIMEI = "86123456" + currSerialStr  currIMEI = currIMEI + str(GetCheckDigit(currIMEI)) + "\n"  outfile.write(currIMEI)    currSerial += 1    #generate IMEIS with model number 234567  currSerial = 1  for i in range(30000):  currSerialStr = str(currSerial)  while len(currSerialStr) < 6:  currSerialStr = "0" + currSerialStr  currIMEI = "86234567" + currSerialStr  currIMEI = currIMEI + str(GetCheckDigit(currIMEI)) + "\n"  outfile.write(currIMEI)    currSerial += 1  ##main  GenerateIMEI() |
| **Evidence 14**  *Partial screenshots* |
| First ten IMEIs  Last ten IMEIs |
| **Evidence 15**  *Task 3.4 program code* |
| class Stack():  '''stack class'''    def \_\_init\_\_(self):  self.\_\_list = []    def push(self,element):  self.\_\_list.insert(0,element)    def display(self):  print("".join(self.\_\_list))  def IMEI2HEX(quotient):  quotient = int(quotient)  convert = 0,1,2,3,4,5,6,7,8,9,'A','B','C','D','E','F'  stack = Stack()    while quotient > 15:  rem = convert[quotient % 16]  quotient = quotient // 16  stack.push(str(rem))  stack.push(str(convert[quotient]))  stack.display()  def GetCheckDigit(IMEI):  imeiSum = 0  for i in range(14):  if i % 2 == 0:#even  imeiSum += int(IMEI[i])  else: #odd  tempSum = 2 \* int(IMEI[i])  if tempSum >= 10:  tempSum = str(tempSum)  imeiSum += int(tempSum[0]) + int(tempSum[1])  else:  imeiSum += tempSum    checkDigit = imeiSum % 10  if checkDigit == 0:  return 0  else:  return 10 - checkDigit  def IsValidIMEI(IMEI):  if len(IMEI) != 15: #length check  return False  try:  int(IMEI)  except: #check if IMEI is an integer  return False  if not GetCheckDigit(IMEI) == int(IMEI[14]):  return False  else:  return True  ##main  done = False  while not done:  uinput = input("Input IMEI:")  if uinput == "":  done = True  else:  if not IsValidIMEI(uinput):  print("invalid IMEI")  else:  IMEI2HEX(uinput) |
| **Evidence 16**  *Screenshot* |
|  |
| **Question 4**  **Evidence 17**  *TASK 4.1 program code* |
| import random  import datetime  ##main  username = input("Enter username:")  timestamp = datetime.datetime.now()  timestamp = timestamp.strftime("%Y-%m-%d %H:%M:%S")  print("Hello " + username + "!")  print("The time now is " + timestamp)  with open("LEADERBOARD.DAT","r") as infile:  leaderArr = []    line = infile.readline()  while line != "":  leader = line.split(",")  leaderArr.append(leader)  line = infile.readline()      for leader in leaderArr:  print("The current game leader is " + leader[1] + " with a score of " + leader[2])    with open("QUESTIONS.DAT","r") as infile:  line = infile.readline()  qArr = []  while line != "":  if line[0] == "Q":  question = []  question.append(line)    line = infile.readline()  while line[0] == "A":  question.append(line)  line = infile.readline()  if line == "":  break    qArr.append(question)  numQ = len(qArr)  score = 0  done = False  while not done:  proceed = input("Proceed? [y/n]")  if proceed == "n":  done = True  break  elif proceed == "y":  for i in range(numQ):  print(qArr[i][0])  numA = len(qArr[i]) - 1  for j in range(1,numA):  print(qArr[i][j])  uinput = input("Enter Answer:")  correct = False  while not correct:  if uinput == "1":  print("correct!")  correct = True  score += 1  else:  print("wrong!")    else:  print("Invalid input.") |
| **Evidence 18**  *Annotated screenshots* |
| Prints timestamp, current highscore correctly  Checks for correct answer and tells user |
| **Evidence 19**  *TASK 4.2 program code* |
| [paste evidence here] |
| **Evidence 20**  *Annotated screenshots* |
| [paste evidence here] |