Candidate name: Melvin Foo

Centre number: I DK

Index number: 08

Programming language used: Python 3.3

|  |
| --- |
| **EVIDENCE 1**  def open\_files(name):  try:  file = open(name, 'r')  lines = file.readlines()  file.close()  new\_lines = []  for line in lines:  line = line.rstrip()  runner\_id = line[:4]  hour = int(line[4:5]) \* 3600  minute = int(line[6:8]) \* 60  sec = int(line[9:])  new\_lines.append([runner\_id , hour+minute + sec])    return new\_lines  except IOError:  print('Unable to open file')  def insertion\_sort(a):#insertion sort  a\_to\_return = [a[0]]  for runner in a[1:]: #skip first  j = 0 #index to track  while j < (len(a\_to\_return)):  if a\_to\_return[j][1] > runner[1]: #timing will be at the second index  break  j += 1  a\_to\_return.insert(j, runner)  return a\_to\_return      swim = open\_files('SWIM.dat')  cycle = open\_files('CYCLE.dat')  run = open\_files('RUN.dat')  all\_competitors = []  for competitors in swim:  all\_competitors.append(competitors[0])  for competitors in cycle:  all\_competitors.append(competitors[0])  for competitors in run:  all\_competitors.append(competitors[0])  valid = []  for competitor in all\_competitors:  if all\_competitors.count(competitor) == 3: #finish all 3 stages  if competitor not in valid: #not already added  valid.append(competitor)  timing = {}  for competitor in swim:  if competitor[0] in valid:  timing[competitor[0]] = competitor[1]  for competitor in cycle:  if competitor[0] in valid:  timing[competitor[0]] += competitor[1]  for competitor in run:  if competitor[0] in valid:  timing[competitor[0]] += competitor[1]  a\_to\_sort = []  for key in timing.keys():  a\_to\_sort.append([key, timing[key]])  sorted\_ = insertion\_sort(a\_to\_sort)  print('1', sorted\_[0][0])  print('2', sorted\_[1][0])  print('3', sorted\_[2][0]) |
| **EVIDENCE 2** |
| **EVIDENCE 3**  def open\_files(name):  try:  file = open(name, 'r')  lines = file.readlines()  file.close()  new\_lines = []  for line in lines:  line = line.rstrip()  runner\_id = line[:4]  hour = int(line[4:5]) \* 3600  minute = int(line[6:8]) \* 60  sec = int(line[9:])  new\_lines.append([runner\_id , hour+minute + sec])    return new\_lines  except IOError:  print('Unable to open file')  def insertion\_sort(a):#insertion sort  a\_to\_return = [a[0]]  for runner in a[1:]: #skip first  j = 0 #index to track  while j < (len(a\_to\_return)):  if a\_to\_return[j][1] > runner[1]: #timing will be at the second index  break  j += 1  a\_to\_return.insert(j, runner)  return a\_to\_return      swim = open\_files('SWIM.dat')  cycle = open\_files('CYCLE.dat')  run = open\_files('RUN.dat')  all\_competitors = []  for competitors in swim:  all\_competitors.append(competitors[0])  for competitors in cycle:  all\_competitors.append(competitors[0])  for competitors in run:  all\_competitors.append(competitors[0])  valid = []  for competitor in all\_competitors:  if all\_competitors.count(competitor) == 3: #finish all 3 stages  if competitor not in valid: #not already added  valid.append(competitor)  timing = {}  for competitor in swim:  if competitor[0] in valid:  timing[competitor[0]] = competitor[1]  for competitor in cycle:  if competitor[0] in valid:  timing[competitor[0]] += competitor[1]  for competitor in run:  if competitor[0] in valid:  timing[competitor[0]] += competitor[1]  a\_to\_sort = []  for key in timing.keys():  a\_to\_sort.append([key, timing[key]])  sorted\_ = insertion\_sort(a\_to\_sort)  print(sorted\_)  timings = []  for competitor in sorted\_:  timing = competitor[1]  timings.append(timing)  f= open('RESULTS.dat', 'w+')  for i in range(len(sorted\_)-1):  name = sorted\_[i][0]  hour = sorted\_[i][1] // 3600  minute = (sorted\_[i][1] % 3600) // 60  sec = (sorted\_[i][1]) % 60  for competitor in swim:  if competitor[0] == name:  swim\_timing = competitor[1]  for competitor in cycle:  if competitor[0] == name:  cycle\_timing = competitor[1]  for competitor in run:  if competitor[0] == name:  run\_timing = competitor[1]  string  print('{0:2>} {1} {2}:{3}:{4}'.format(i+1, sorted\_[i][0], hour, minute, sec), file = f)  f.close() |
| **EVIDENCE 4** |
| **EVIDENCE 5**  #Q2  def reversal(string):  if len(string) == 1:  return string  return string[-1] + reversal(string[:(len(string)-1)])  #normal test case  print(reversal('yooooo'))  #symatrical case  print(reversal('1234321')) |
| **EVIDENCE 6** |
| **EVIDENCE 7**  #Q2.2  def reversal(string):  if len(string) == 1:  return string  return string[-1] + reversal(string[:(len(string)-1)])  def is\_palindrome(string):  if string.isalpha():  string = string.lower()  return string == reversal(string)  #lower case and upper case test  print(is\_palindrome('Wow'))  #normal case  print(is\_palindrome('wew'))  #fail case  print(is\_palindrome('eeew')) |
| **EVIDENCE 8** |
| **EVIDENCE 9** |
| **EVIDENCE 10** |
| **EVIDENCE 11**  #Q2.4  def even\_first(num, index=0):  string = str(num)  if int(string[index]) % 2 == 0: #even  return string[index] + even\_first(int(string[1:], index + 1))  else: #odd  return even\_first(int(string[1:], index + 1)) + string[index]  even\_first(12) |
| **EVIDENCE 12** |
| **EVIDENCE 13**  def swap(string, index=0):  if string == '':  return ''  return string[index+1] + string[index] + swap(string[2:])  #normal  print(swap('hiyo')) |
| **EVIDENCE 14** |
| **EVIDENCE 15**  #Q3  def import\_records (name):  try:  file = open(name, 'r')  lines = file.readlines()  file.close()  new\_lines = []  for line in lines:  line = line.rstrip()  date = line[:8]  cust\_id = line[8:14]  price = float(line[15:].lstrip())  new\_lines.append([date, cust\_id, price])    return new\_lines  except IOError:  print('Unable to open file')  l = open\_files('TRANSACTIONS.dat')  q = queue()  for line in l:  q.enqueue(line)  print(open\_files('TRANSACTIONS.dat')) |
| **EVIDENCE 16** |
| **EVIDENCE 17**  #Q3.2  def import\_records(name):  try:  file = open(name, 'r')  lines = file.readlines()  file.close()  new\_lines = []  for line in lines:  line = line.rstrip()  date = line[:8]  cust\_id = line[8:14]  price = float(line[15:].lstrip())  new\_lines.append([date, cust\_id, price])    return new\_lines  except IOError:  print('Unable to open file')  def display\_sales(ql): #ql is a queue linked list  not\_empty = True  purchases = []  while not\_empty:  purchase = ql.pop()  if purchases = None:#reached the end  not\_empty = False  else:  purchases.append(purchase)  dates = {}  for purchase in purchases:  date = purchase[0] #date is in the 0 index  cost = purchase[2]  if date not in dates.keys():  dates[date] = cost  else:  dates[date] += cost  print('Date Sales')  for date in dates.keys():  print('{0}{1:>10.2f}'.format(date, dates[date])        l = open\_files('TRANSACTIONS.dat')  q = queue()  for line in l:  q.enqueue(line)  display\_sales(q) |
| **EVIDENCE 18** |
| **EVIDENCE 19** |
| **EVIDENCE 20 (Bonus)** |
| **EVIDENCE 21 (Bonus)** |