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

```
>>>
1 A123
2 A134
3 A575
>>>
```

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

```
1   A123 1:52:19
2   A134 1:57:19
3   A575 1:58:18
4   A795 2:8:4
5   A154 2:8:11
6   A131 2:11:39
7   A563 2:13:46
8   A165 2:16:32
9   A345 2:16:32
10   A677 2:16:33
11   A543 2:16:34
12   A539 2:16:53
13   A676 2:16:56
14   A546 2:16:57
15   A541 2:21:18
```

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

```
#normal test case
print(reversal('yooooo'))

#symatrical case
print(reversal('1234321'))
```

```
>>>
oooooy
1234321
>>>
```

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

```
#lower case and upper case test
print(is_palindrome('Wow'))

#normal case
print(is_palindrome('wew'))

#fail case
print(is_palindrome('eeew'))
```

```
>>>
True
True
False
>>>
```

EVIDENCE 9

EVIDENCE 10

```
#more vowels
print(has_more_vowels('Google IO'))

#less vowels
print(has_more_vowels('Apple WWDC'))

#equal
print(has_more_vowels('iv'))
```

```
>>>
True
False
False
>>>
```

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

```
#normal
print(swap('hiyo'))



>>>
ihoy
>>>
```

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

```
>>>
[['20150227', 'C11243', 227.7], ['20150227', 'C31134', 119.6], ['20150227', 'C8
8691', 55.3], ['20150227', 'C37254', 339.2], ['20150227', 'C45353', 49.9], ['20
150228', 'C45354', 99.8], ['20150228', 'C26285', 39.5], ['20150228', 'C25621',
238.8], ['20150228', 'C92852', 71.1], ['20150228', 'C72959', 213.3], ['20150301
', 'C22523', 50.7], ['20150301', 'C79257', 207.9], ['20150301', 'C79252', 396.0
], ['20150301', 'C88691', 84.5], ['20150301', 'C26285', 99.8], ['20150301', 'C4
5353', 55.3]]
>>>
```

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)