Jeff Hildebrandt

Programming Assignment 1

COT 5405

1) The following is pseudo code for the algorithm. The actual algorithm is at the end of this document

```
barbers = array of barbers in order of arrival
d = difference allowed before order switch

while there are still services for the day
      highest = largest earnings so far
      for each barber in barbers (goes in order)
            barber's earnings += service cost
            if  barber's earnings > barber highest earning
                  highest = current barber
            else if barber's earnings + d <= barber with the highest
            earning so far
                  reorder so current barber is in front of the barber with
                  the highest earning so far
```

2) Unfortunately, the way this algorithm was set up. There's no guarantee that one barber won't get the lowest service every time and another barber won't get the maximum service every time. Also, there's no guarantee that the barber with the lowest range will get an extra round depending on if the total number of customers that come in that day are divisible by the number of barbers.

   So, based on the previous paragraph, the difference that my algorithm can guarantee is the following:
   S = guaranteed income difference of the highest earning barber and the lowest earning
   H = cost of the most expensive service
   L = cost of the least expensive service
   R = number of rounds

$$S = R(H - L)$$

3) The algorithm runs once through the number of barbers per round. Potentially, a reordering of the barbers array can happen number of barbers – 1 times since it's impossible for a reorder to happen on the first barber. Which because an insert in an array is an O(n) time complexity, the algorithm runs at O(n(n-1)).
4) Yes, S would be smaller if barbers could have more than one assignment in a round. This would allow a barber that is much farther behind in earnings to catch up to the higher earners. If the highest earner has earnings higher than the most expensive service to the lowest earner, then the lowest earner could be assigned twice in one round which would equalize all earnings.
5) As covered by question number four, having barbers able to be assigned multiple times per round would even out earnings. Another thing to consider would be to assign a barber after knowing the

price of the service. This would allow a low earner to wait for an expensive service to come across in a round, allowing the wages to be equalized based off assigning expensive services to low earners.

6) The program is run with random services, random number of services, random d number, and random barber order. Here are a few sample outputs:

```
d = 20
Total Services: 81 Services:
40  10  10  30  30  40  30  10  10  20  40  40
20  20  30  30  30  20  40  30  30  10  10  20
10  10  10  40  20  20  30  30  40  30  40  40
30  20  30  10  30  40  30  10  40  30  40  20
30  10  40  20  20  20  30  30  40  40  20  20
40  20  40  20  30  20  20  40  20  20  20  10
20  10  20  10  10  20  20  40  40
Barber order:
B , 0  E , 0  A , 0  D , 0  C , 0
E , 10  B , 40  A , 10  D , 30  C , 30
E , 50  A , 20  B , 70  D , 40  C , 50
A , 60  E , 90  D , 60  B , 90  C , 80
A , 90  D , 80  E , 120  C , 110  B , 130
D , 90  A , 120  E , 130  C , 130  B , 140
D , 100  A , 130  C , 150  E , 170  B , 160
D , 130  A , 160  C , 190  E , 200  B , 200
D , 170  A , 190  C , 210  B , 210  E , 230
D , 200  A , 230  B , 220  C , 240  E , 270
D , 230  B , 240  A , 270  C , 270  E , 280
D , 270  B , 260  A , 290  C , 290  E , 310
D , 300  B , 300  C , 310  A , 330  E , 330
B , 320  D , 340  C , 350  A , 350  E , 360
B , 340  D , 360  A , 370  C , 390  E , 380
B , 360  D , 370  A , 390  C , 400  E , 400
B , 370  D , 380  A , 410  C , 420  E , 440
B , 410  D , 380  A , 410  C , 420  E , 440
```

```
d = 30
Total Services: 74 Services:
20  30  20  30  20  10  40  10  30  40  40  30
30  20  10  20  20  10  30  10  30  40  40  30
30  10  20  40  20  20  10  30  10  20  20  20
30  10  40  10  40  30  40  30  10  20  40  30
20  20  10  40  20  20  40  30  40  20  10  20
20  30  40  40  20  40  40  30  10  40  10  10
40  10
Barber order:
D , 0  E , 0  C , 0  A , 0  B , 0
D , 20  E , 30  C , 20  A , 30  B , 20
D , 30  C , 30  E , 70  A , 60  B , 60
D , 70  C , 60  B , 70  E , 100  A , 80
D , 90  C , 80  B , 80  A , 90  E , 130
D , 120  C , 120  B , 120  A , 120  E , 160
D , 130  C , 140  B , 160  A , 140  E , 180
D , 140  C , 170  B , 170  A , 160  E , 200
D , 160  C , 200  B , 180  A , 200  E , 210
D , 200  C , 230  B , 220  A , 230  E , 220
D , 220  E , 240  C , 270  B , 250  A , 250
D , 230  E , 280  C , 290  B , 270  A , 290
D , 260  B , 280  E , 320  C , 310  A , 310
D , 280  B , 310  A , 330  E , 360  C , 350
D , 320  B , 350  A , 360  E , 370  C , 390
D , 330  B , 360  A , 400  E , 380  C , 390
```

```
d = 10
Total Services: 70 Services:
30  10  30  30  40  30  10  10  20  20  30  30
10  30  30  30  30  30  10  20  20  40  30  30
20  20  20  10  20  10  20  10  30  10  30  40
30  40  40  20  10  40  10  20  40  20  30  40
10  30  10  10  40  20  30  20  10  10  20  30
10  20  40  30  30  10  40  20  40  10
Barber order:
D , 0  B , 0  C , 0  A , 0  E , 0
B , 10  D , 30  C , 30  A , 30  E , 40
B , 40  D , 40  C , 40  A , 50  E , 60
C , 50  B , 70  D , 70  A , 80  E , 90
C , 80  A , 90  B , 100  D , 100  E , 110
C , 100  A , 130  B , 130  D , 130  E , 130
C , 120  B , 140  A , 150  E , 140  D , 150
C , 140  B , 150  E , 150  A , 180  D , 180
C , 180  B , 180  E , 190  D , 200  A , 220
C , 190  E , 200  B , 220  D , 220  A , 260
C , 210  E , 230  D , 230  B , 260  A , 290
C , 220  E , 240  D , 270  B , 280  A , 320
C , 240  E , 250  D , 280  B , 300  A , 350
C , 250  E , 270  D , 320  B , 330  A , 380
C , 260  E , 310  D , 340  B , 370  A , 390
```

```
d = 40
Total Services: 64 Services:
20  20  10  40  30  30  10  20  40  30  20  40
30  20  10  10  30  10  30  40  40  20  40  30
40  30  40  40  40  20  20  10  40  40  20  40
20  40  10  10  10  10  30  20  20  20  10  20
20  40  20  40  10  30  40  40  10  10  20  40
10  10  20  10
Barber order:
A , 0  C , 0  E , 0  D , 0  B , 0
A , 20  C , 20  E , 10  D , 40  B , 30
A , 50  C , 30  E , 30  D , 80  B , 60
A , 70  C , 70  E , 60  D , 100  B , 70
A , 80  C , 100  E , 70  D , 130  B , 110
A , 120  C , 120  E , 110  D , 160  B , 150
A , 150  C , 160  E , 150  D , 200  B , 170
A , 170  C , 170  E , 190  B , 190  D , 240
A , 210  C , 190  E , 230  B , 200  D , 250
A , 220  C , 200  B , 220  E , 260  D , 270
A , 240  C , 210  B , 240  E , 280  D , 310
A , 260  C , 250  B , 250  E , 310  D , 350
C , 260  A , 300  B , 260  E , 330  D , 390
C , 270  A , 310  B , 280  E , 340  D , 390
```

```
import random
from collections import deque

class BarberSchedule:
    def __init__(self, barbers, services, dollarDifference):
        self.barbers = barbers
        self.services = services
        self.dollarDifference = dollarDifference

    def start(self):
        print("d =",self.dollarDifference)
        print("Services:")
        for service in self.services:
            print(service," ",end="")
        print()
        print("Barber order:")
        self.printBarbers()
        while not self.nextRound():
            self.printBarbers()
        self.printBarbers()

    def nextRound(self):
        highest = 0
        for i in range(len(self.barbers)):
            self.barbers[i].earnings += self.services.popleft()
            if self.barbers[i].earnings > self.barbers[highest].earnings:
                highest = i
            elif self.barbers[i].earnings + self.dollarDifference <=
self.barbers[highest].earnings:
                self.barbers.insert(highest, self.barbers.pop(i))
            if len(self.services) <= 0:
                return True
    def printBarbers(self):
        for barber in self.barbers:
            print(barber.name,",",barber.earnings," ", end="")
        print()

class Barber:
    def __init__(self, name):
        self.earnings = 0
        self.name = name

def main():
    barbers = [Barber("A"), Barber("B"), Barber("C"), Barber("D"), Barber("E")]
    services = deque([])
    servicePrices = [10, 20, 30, 40]
    for x in range(random.randint(10, 50)):
        services.append(servicePrices[random.randint(0, 3)])
    random.shuffle(barbers)
    dollarDifference = servicePrices[random.randint(0, 3)]
    barberSchedule = BarberSchedule(barbers, services, dollarDifference)
    barberSchedule.start()

main()
```