CSEN 102 - Midterm 2012 solutions

November 14, 2015

1 Conditional

Your task is to implement an algorithm that can calculate your maximum heart rate and your optimal training pulse for fat-burning, endurance increase or cardiovascular system improvement. The Formula for calculating the maximum heart rate (Pulse) depends on the age and the gender:

- For men: 200 - Age
- For women 226 - Age

The training can be classified into the following zones

Health zone: This amounts to 50-60& of the maximum heart rate. Within this pulse range particularly the cardiovascular system will be invigorated. This range is particularly suitable for beginners.

Fat burning zone: This amounts to 60-70% of the maximum heart rate. Within this pulse range, most calories from fat are burned. Furthermore the cardiovascular system will be trained.

Anaerobic zone: This amounts to 80-90% of the maximum heart rate. Within this pulse range, the body cannot cover the oxygen demand any longer. This range is for the development of power and muscle mass.

Red zone: This amounts to 90-100% of the maximum heart rate. This pulse range should be handled with caution. It is dangerous for beginners and can be harmful for the heart.

(a) Write an algorithm that given the age and the gender should display the different zones. For example, your algorithm should display the following for a 45 years old man:

Health zone:	Between 88 and 105
Fat-burning zone:	to 122
Aerobic zoe:	to 140
Anaerobic zone:	to 158
Maximum heart rate / Red zone:	to 175

Solution

```
age = eval(input())
gender = input()

if gender == 'male':
  maxHeartRate = 220 - age
else:
  maxHeartRate = 226 - age

healthZoneFrom = 50 * maxHeartRate / 100
healthZoneTo = 60 * maxHeartRate / 100
fatBurningZoneTo = 70 * maxHeartRate / 100
aerobicZoneTo = 80 * maxHeartRate / 100
anaerobicZoneTo = 90 * maxHeartRate / 100

print("Health Zone: Between ", healthZoneFrom, " and ", healthZoneTo)
print("Fat-burning zone: to " , fatBurningZoneTo)
print("Aerobic zone: to " , aerobicZoneTo)
print("Anaerobic zone: to " , anaerobicZoneTo)
print("Maximum heart rate / Red zone: to ", maxHeartRate)
```

(b) Write an algorithm that given the age, the gender and the heart rate will display the corresponding zone. For example, the algorithm should display for a 45 years old man and heart rate of 190, the following message

```
Red Zone
```

Your algorithm should consist of only *if* statements, i.e. you are not allowed to use any *else* statements.

Solution

```
age = eval(input())
gender = input()
heartRate = eval(input())
if  gender == 'male':
  maxHeartRate = 220 - age
if  gender != 'male':
  maxHeartRate = 226 - age

healthZoneFrom = 50 * maxHeartRate / 100
healthZoneTo = 60 * maxHeartRate / 100
fatBurningZoneTo = 70 * maxHeartRate / 100
aerobicZoneTo = 80 * maxHeartRate / 100
anaerobicZoneTo = 90 * maxHeartRate / 100

# Note that because the restriction  of  not being able  to  use  else
    statements
# our if  conditions  must be exclusive  in  order  to  prevent the  triggering
    of
# more than one
if (healthZoneFrom <= heartRate <= healthZoneTo):
  print ("Health Zone")

if (healthZoneTo <  heartRate <= fatBurningZoneTo):
  print ("Fat-burning zone"

if (fatBurningZoneTo < heartRate <= aerobicZoneTo):
  print ("Aerobic zone")

if (heartRate > aerobicZoneTo and heartRate <= anaerobicZoneTo):
  print ("Anaerobic zone")

if (heartRate > anaerobicZoneTo):
  print ("Red zone")
```

2 Sequential (a) + Iteration (b)

Many people begin running because they want to lose weight. As one of the most vigorous exercises out there, running is an extremely efficient way to burn calories and drop pounds. However, there are other ways to burn calories. In the following several activities are listed to burn around 250 calories:

(a) **Dancing:** 52 minutes

Doing yoga: 90 minutes

Riding a bike: 30 minutes

Rollerblading: 18 minutes

Running: 23 minutes

Standing while talking on the phone: 120 minutes

Horseback riding: 57 minutes

So as you can see, what will only take a few minutes to consume, could take a large amount of time to burn off in the gym. Assume that burning calories is proportional to the time needed in an activity (which is in general not the case), write an algorithm that given the calories will output the activities above with the time needed to burn those calories.

```
calories = eval(input())

# The '.0' is to force floating point division
# ie 10/3 is 3 10/3.0 or 10.0/3 is 3.3333~
# another method is to have had calories = float(input())
# instead of calories = eval(input())

dancingMins = calories / 250.0 * 52
yogaMins = calories / 250.0 * 90
ridingMins = calories / 250.0 * 30
rollerMins = calories / 250.0 * 18
phoneMins = calories / 250.0 * 120
horseMins = calories / 250.0 * 57

print("Dancing: ", dancingMins, " minutes")
print("Doing yoga: ", yogoMins, " minutes")
print("Riding a bike: ", ridingMins, " minutes")
print("Rollerblading: ", rollerMins, " minutes")
print("Standing while talking on the phone: " , phoneMins, " minutes")
print("Horseback riding: ", horseMins, " minutes")
```

(b) A lady decides to lose weight to be prepared for her wedding. Assume that she is able to loose 5% from her weight every month by doing a strict diet.

(a) Write an algorithm that given the weight of a lady and her target weight will calculate how many months the diet will last.

Solution

```
weight = eval(input())
targetWeight = eval(input())

months = 0

while weight > targetWeight:
  weight = weight - 5/100.0 * weight
  months = months + 1

print("The number of months the diet will last: ", months)
```

(c) Write an algorithm that given the weight, the current date and the date of the wedding will calculate the weight of the lady during her wedding day. The date is given by day, month and year. Your algorithm should display the number of days left to the wedding as well as the weight. For simplicity take the integer division of the days left by 30 to get the number of months.

Solution

```
weight = eval(input())
currentDay = eval(input())
currentMonth = eval(input())
currentYear = eval(input())
weddingDay = eval(input())
weddingMonth = eval(input())
weddingYear = eval(input())

yearsDifference = weddingYear - currentYear
monthDifference = weddingMonth - currentMonth
daysDifference = weddingDay - currentDay
totalDaysDifference = yearsDifference * 12 * 30 + monthDifference * 30 +
      daysDifference
months = totalDaysDifference / 30
while months > 0:
  weight = weight - 5.0 /100 * weight
  months = months - 1

print("The number of days left to the wedding: " , totalDaysDifference)
print("The weight of the lady during her wedding day: " , weight)
```

3 Tracing

Given the following Python code fragment:

```
a = 1
b = 2
c = 6

while c > 0:
  if c % 2 == 1:
    a = a * b
  b = b * b
  c = c / 2
  print(a, ",", b, ",", c)
```

(a) What is the output of the code above for a=1, b=2 and c=6. Use a tracing table to trace the while loop.

Solution

a	b	c	Output so far
1	2	6	
1	4	3	1,4,3
4	15	1	1,4,3,4,16,1
64	256	0	1,4,3,16,1,64,256,0

(b) What is the value of the b after the execution of the code for any values of a, b and c.

Solution

The value of the b is b^{2^i} where i is the number of iterations of the while loop. Since the c is divided by 2 in every iteration, the i will be equal to $ceil(\log_2 c)$. Therefore, the value of the b after the execution of the code will be $b^{2^{ceil(\log_2 c)}}$ (Thanks to Dr. Haythem O. Ismail for his help).

4 Iteration over Lists

(a) Given a list of integers, write an algorithm that checks whether a list is ordered in ascending order or not. For example for the list consisting of

```
5 4 12 16 1
```

the algorithm should display

```
The list is not sorted
```

For the list

```
5 10 12 16
```

the algorithm should display

```
The list is sorted
```

Note: For the case where the list is not sorted, your algorithm should stop right away. For the example above, your algorithm should stop after comparing the 5 with the 4.

Solution

```
n = eval(input())
i = 0
A = [ ]
while i < n:
  A[i] = eval(input())

# Reset counter
i = 0
isSorted = True
while i < n and isSorted:
  if A[i] > A[i + 1]:
    isSorted = False
  i = i + 1

if isSorted:
  print("The list is sorted")
else:
  print("The list is not sorted")
```

(b) Write an algorithm that uses the following approach to sort a list of positive integers excluding zero. Each integer x of the input list will be stored in the index that corresponds to the value of x in the output list. For example, for the following list

4	3	7	2

the algorithm should sort the elements in the following list:

-1	2	3	4	-1	-1	7

and displays the following message

```
2 3 4 7
```

Please note that -1 stored in an index i means that the element with the value i does not exist in the orginal list. Therefore, you are required to fill these cells with -1.

Solution

```
n = eval(input())
i = 0
A = [ ]
while i < n:
  A[i] = eval(input())

i = 0
maxValue = -1

while i < n:
  if maxValue < A[i]:
    maxValue = A[i]
  i = i + 1

i = 0
# A list of length maxValue where each element is a negative one
B = [-1] * maxValue
while i < n:
  B[A[i]] = A[i]
  i = i + 1

i = 0
# In order to print them on one line
# We must construct a new string
# and use print that at the end
# since each print statement
# forces a new line
output = ''
while i < maxValue:
  if B[i] != -1:
    output = output + str(B[i])
  i = i + 1
print(output)
```

(c) What is the drawback of the sorting algorithm (one English statement)?

Solution

The drawback of this algorithm is the inefficeincy in terms of memory storage. For example, if the list contains only one number: 10^6 the list will have 10^6 cells all filled with -1 except for the last cell.