German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher
Dr. Fatma Meawad

December 06, 2012

Introduction to Computer Science
Winter term 2012-2013
Midterm Exam

Bar Code

Instructions: Read carefully before proceeding.

1) Duration of the exam: 2 hours (120 minutes).
2) (Non-programmable) Calculators are allowed.
3) No books or other aids are permitted for this test.
4) This exam booklet contains 11 pages, including this one. Three extra sheets of scratch paper are attached and have to be kept attached. **Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete**.
5) Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem or on the four extra sheets and make an arrow indicating that. **Scratch sheets will not be graded unless an arrow on the problem page indicates that the solution extends to the scratch sheets**.
6) When you are told that time is up, stop working on the test.

Good Luck!

Don't write anything below `;-)`

Exercise	1	2	3	4	$\sum$
Marks	12	18	10	16	56
Final Marks					

Exercise 1 (6+6=12 Marks)

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: 220 - 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.

- Aerobic zone: This amounts to 70-80% of the maximum heart rate.

  Within this pulse range, carbohydrates and fats are burned for power production in the muscle cells. This range requires the cardiovascular system as well as the lung and the metabolism.

- 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 zone:                 to 140
Anaerobic zone:               to 158
Maximum heart rate / Red zone: to 175
```

Solution:

```
get age, gender

if(gender = "male") then
   set maxHeartRate to 220 - age
else
   set maxHeartRate to 226 - age
endif

set healthZoneFrom to 50 * maxHeartRate / 100
set healthZoneTo to 60 * maxHeartRate / 100
print "Health Zone: Between ", healthZoneFrom, " and ", healthZoneTo
```

```
set fatBurningZoneTo to 70 * maxHeartRate / 100
print "Fat-burning zone: to "  , fatBurningZoneTo

set aerobicZoneTo to 80 * maxHeartRate / 100
print "Aerobic zone: to "  , aerobicZoneTo

set anaerobicZoneTo to 90 * maxHeartRate / 100
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 then statements, i.e. you are not allowed to use any `else` statements.

Solution:

```
get age, gender, heartRate

if(gender = "male") then
   set maxHeartRate to 220 - age
else
   set maxHeartRate to 226 - age
endif

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

if(heartRate >= healthZoneFrom AND heartRate <= healthZoneTo) then
   print "Health Zone"
endif

if(heartRate > healthZoneTo AND heartRate <= fatBurningZoneTo) then
   print "Fat-burning zone"
endif

if(heartRate > fatBurningZoneTo AND heartRate <= aerobicZoneTo) then
   print "Aerobic zone"
endif

if(heartRate > aerobicZoneTo AND heartRate <= anaerobicZoneTo) then
   print "Anaerobic zone"
endif

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

```
endif
```

Exercise 2 (4+6+8=18 Marks)

Sequential (a) + Iteration (b)

a) 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:

- 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.

Solution:

```
get calories

set dancingMins  to (calories/250) * 52
set yogoMins     to (calories/250) * 90
set ridingMins   to (calories/250) * 30
set rollerMins   to (calories/250) * 18
set phoneMins    to (calories/250) * 120
set horseMins    to (calories/250) * 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.

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

   Solution:

   ```
   get weight
   get targetWeight

   set months to 0

   while(weight > targetWeight)
   ```

```
{
   set weight to weight - (5.0/100.0) * weight
   set months to months + 1
}

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

2. 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:

```
get weight
get currentDay, currentMonth, currentYear
get weddingDay, weddingMonth, weddingYear

set yearsDifference to weddingYear  - currentYear
set monthDifference to weddingMonth - currentMonth
set daysDiffernce   to weddingDay   - currentDay

set totalDaysDifference to yearsDifference*12*30 + monthDifference*30 + daysDifferr
set months to totalDaysDifference/30

while(months > 0)
{
   set weight to weight - (5.0/100) * weight
   set months to months - 1
}

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

Exercise 3 (6+4=10 Marks)

Given the following Java code fragment:

```
int a, b, c;
a = 1;
b = 2;
c = 6;
while (c > 0) {
     if ( c % 2 == 1 ) {
        a = a * b;
     }
     b = b * b;
     c = c / 2;
     System.out.println( a + "," + b + "," + c);
}
```

and the corresponding pseudocode algorithm

```
get a, b, c
while (c > 0) {
     if ( c % 2 == 1 )
     then
        set a to a * b
     endif
     set b to b * b
     set c to  INT(c / 2)
     print a
     print b
     print 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	16	1	1, 4, 3, 4, 16, 1
64	256	0	1, 4, 3, 4, 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).

Exercise 4 (6+8+2=16 Marks)

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:

```
get n
get A1...An
set i to 1
set isSorted to true

while(i < n and isSorted = true)
{
   if(A[i] > A[i+1]) then
      set isSorted to false
   endif

   set i to i + 1
}

if(isSorted = true) then
   print "The list is sorted"
else
   print "The list is not sorted"
endif
```

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:

```
get n
get A1...An
set i to 1
set max to -1

while(i <= n)
{
   if(max < A[i]) then
      set max to A[i]
   endif
   set i to i + 1
}

set i to 1
while(i <= max)
{
   set B[i] to -1
   set i to i+1
}

set i to  1
while(i <= n)
{
   set B[A[i]] to A[i]
   set i to i+1
}

set k to 1
while (k <= max)
{
   if(B[k] <> -1) then
      print B[k], " "
   endif

   set k to k+1
}
```

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.

Extra Sheet

Extra Sheet

Extra Sheet