Stevie Parris

Professor Popyack

CS 164

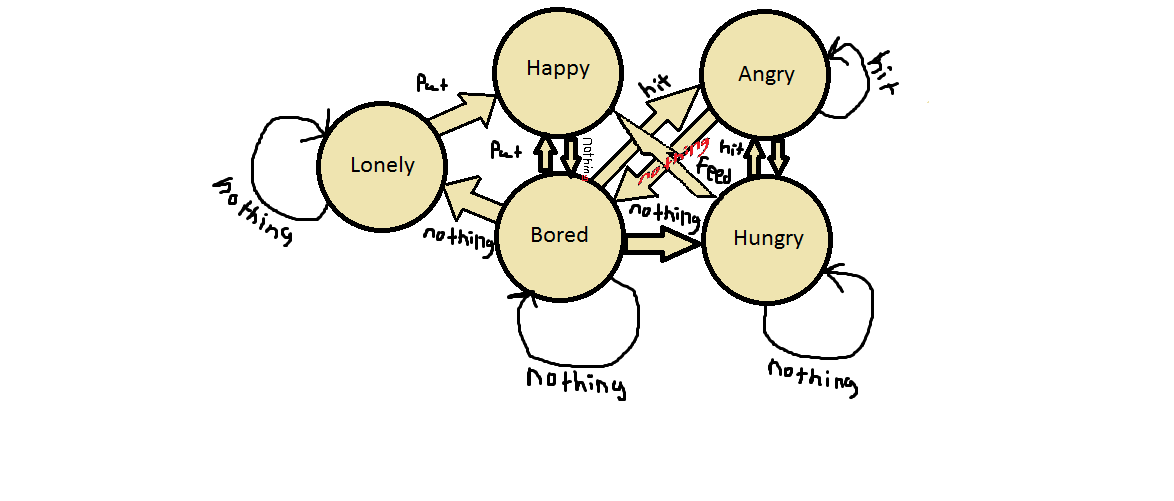
11 November 2013

CS Homework #6

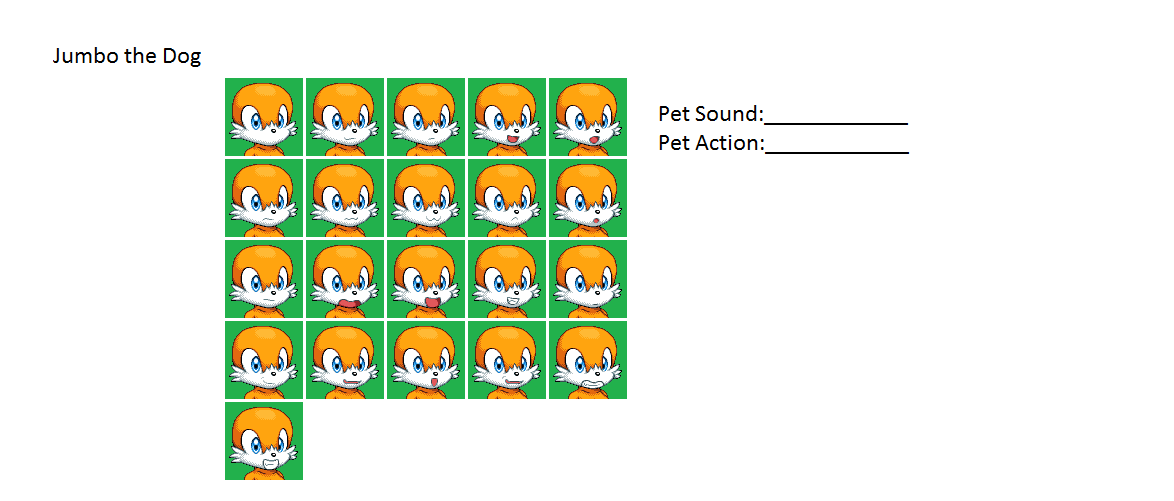
**Problems to be turned in**:

1. Read the articles [Ivars Peterson's MathLand: Pentium Bug Revisited (May 12, 1997)](https://www.cs.drexel.edu/~introcs/Fa13/assignments/HW6/IvarsPeterson/index.html) (https://www.cs.drexel.edu/~introcs/Fa13/assignments/HW6/IvarsPeterson/index.html) and [Cleve's Corner: A Tale of Two Numbers](http://www.mathworks.com/company/newsletters/news_notes/pdf/win95cleve.pdf) (http://www.mathworks.com/company/newsletters/news\_notes/pdf/win95cleve.pdf) , and answer the following questions: ***(15 points)***
   1. Moler's article describes a "Pentium-Safe Division" algorithm, in which certain quotients were identified as "at risk", and gave an alternative way of computing those quotients that was guaranteed to bypass the problem areas on faulty Pentium chips. What was the alternative computation?
      * function z = (x,y)
      * if at\_risk(y)
      * x = (15/16)\*x;
      * y = (15/16)\*y;
      * end
      * z = x/y
      * function a = at\_risk(y)
      * f = and(hex(y), ‘000FF000000000000’)
      * a = any(f == [‘1F’; ‘4F’; ‘7F’; ‘AF’; ‘DF’ ])
   2. The faulty areas on the chips were identified by hex constants. Give the constants in both their hexadecimal and decimal forms.
      * 1F, 4F, 7F, AF, DF
      * 31, 79, 127, 175, 223
   3. Peterson's article gives five binary number prefixes that needed to be followed by six 1's to create the risk of a problem. Give the numbers in both their binary and decimal forms.
      * 1.0001, 1.0100, 1.0111, 1.1010, and 1.1101
      * 1.0625, 1.25, 1.4375, 1.625, and 1.8125
   4. What numeric base does the SRT algorithm use? (*Hint: not base 10*)
      * SRT uses base 4.
   5. Peterson's article also describes a disastrous rocket launch that occurred because of an overflow error. Describe the error.
      * The reason that the rocket crashed was due to a change in number type. A number that was originally a 64 bit floating point was changed to an integer 16 bit and since the computer didn’t know how to interpret overflow, it interpreted the memory dump as instructions to the rocket nozzles.
2. Convert the following numbers from binary to decimal: **0.0101101011...** , **0.1111101110...** Express each as both a fraction of two integers and a decimal number. ***(8 points)***
   * + 11/31, : 0.35483870967
     + 59/60 : 0.983 (the 3 is repeating)
3. We have seen that multiplying a binary number by 2 is accomplished by simply appending a 0 to the end. How does the binary representation of a floating-point number change when it is multiplied by 2? ***(4 points)***
   * + It increases the exponent by 1, shifting it to the left.
4. We have seen that the decimal number 7/10 is represented as **.1011001100110...** in binary. Suppose only 10 bits are available for the mantissa in a floating point representation, and that it is*truncated*, i.e., all bits beyond the 10-th bit are lost. ***(15 points)***
   1. What is the *actual* value of the 10-bit binary number **.1011001100** ?
      * 179/256 = 0.69921875
   2. Suppose 12 bits are available. What is the actual value of the 12-bit binary number **.101100110011**? How much closer an approximation is this number than the 10-bit number in part a?
      * 2867/4096 = 0.69995117187
   3. Floating-point representation of the value **x** uses the value **f** as the mantissa, where **x=(1+f)\*2k**, for some **0<=f<1** . What are the values of **f** and **k** when **x** is 7/10 ?
      * 7/10 = (1+f) \* 2kkkkkkkkk
      * 2-0.6 < 7/10 < 2-0.5
      * k = -0.6
      * 7 = 10\*(1+f)\*2-0.6
      * f=0.06100159655727865764308186091451166707689848008469
   4. What is the 10-bit binary approximation of **f** ? What is the actual value of **x=(1+f)\*2k**, using this value for **f** ?
      * f = 0.00001111100
      * f = 0.060546875
   5. What is the 12-bit binary approximation of **f** ? What is the actual value of **x=(1+f)\*2k**, using this value for **f** ?
      * f = 0.0000111110
      * f = 0.060546875
5. Suppose you are using a computer that represents floating point numbers using 2 bytes: 1 sign bit, 5 bits for the exponent, and 10 bits for the mantissa. Calculate the (binary and decimal) values for each of the quantities below. ***(12 points)***  
   *Express your answer in the following form*:  
   -(1+7/8)\*2-4 = -(15/8)\*2-4 = -(15/128) = -0.1171875
   1. **0 10110 1010100000**
      * (1 + 0.65625)\*27= 6784/32) = 212
   2. **1 00110 1001001000**
      * -(1 + 0.5703125)\*2-9 = 201/65536 = -0.0030670166
   3. **0 11001 1100110101**
      * (1 + 0.8012695)\*210= 1889280/1024 = 1845
   4. **1 01001 1010000111**
      * -(1 + 0.62841796875)\*2-6 = -0.02544403076

A)



B)



C)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Type** | **Possible values** | **Meaning** | **Action** |
| PetFeed | button |  | Feed the dog | Changes state to “happy” if hungry |
| PetHit | button |  | hit the dog | changes state to "angry” if bored or hungry. |
| PetPat | button |  | pat the dog | changes state to "happy” if bored, and if lonely. |
| PetAction | Text box | Wag tail  Chase Tail  Beg | State = Happy  State = Bored  State = Hungry |  |
| PetSound | Text box | Whimper  Growl | State = Lonely  State = Angry |  |