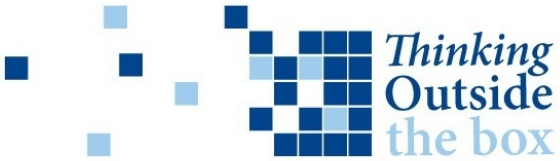
CIS 470 – iOS Programming
HW 17 – Lottery Quick Picks v2



Refer to H/W assignment #12, Lottery Quick Picks (attached in the appendix for your convenience).

Redo the assignment as an App instead of a console program. Your model should be the exact QuickPick class you developed for H/W 12 (minor modifications and corrections are acceptable, but should not be necessary).
Create a new project and add you swift class file to the project. This is your model. Create a view that permits the user to enter numPicks, and range. Add a button to generate the picks and calculate the odds of winning. Add a label to display the winning odds. Because the generated values may need more space than what can fit in a label, I suggest using a non-editable, multi-line TextView for this purpose. This is simply a text view with the "editable" property disabled. Your input fields must use a numeric keyboard, not a full text keyboard. You may optionally replace one of the input fields with a slider control. Additionally, you must do data validation in your app: no negative input values are permitted and numPicks must be less than range. I'll leave it up to you how you handle these user input cases. The keyboard must disappear when appropriate. Again, I'll leave the definition of "appropriate" up to you, but usability will be part of your graded. Include an icon.

You can earn up to 20 points of extra credit for sprucing up your *working* application with a background image, a button image or other embellishments (fonts, color, etc..).

Appendix

CIS 470 – iOS Programming
HW 12 – Lottery Quick Picks

"Quick Picks" refers to the use of a computer to randomly generate playing numbers when purchasing a lottery ticket. Because different lotteries use different range of numbers and different number of picks, the quick pick program has to be configurable. For example, one lottery may call for users to pick 6 numbers from a range of 1 to 36, while another lottery may require its users to pick 5 numbers from a range of 1 to 58. Assume that no lottery allows players to pick repeated values. Your task is to develop a class to generate quick pick numbers given a range and a number of required picks. Your class must have 2 public members and 2 public methods. The public members will be range, an Int representing the range of value from which numbers will be drawn (36 and 58 in the two examples above) and numPicks, an Int representing the number of picks from the range (6 and 5 in the two examples above). One of the two public functions will be generatePicks(), which accepts no parameters and returns an array of Ints containing the quick pick values. To use your class, a user will first set the values of range and numPicks, then call generatePicks().

Your class must also calculate the odds of winning for given values of range and numPicks. This will be returned by a call to the second public function getWinningOdds(), which accepts no parameters and returns a Double. The odds of winning is calculated as 1 in

$$\frac{range!}{numPicks! \cdot (range - numPicks)!}$$

All other class members or methods you create should be private. Note that the '!'s in the above expression denote factorials, not forced unwrapping.

Name your class *QuickPick* and save it all by itself in a swift file with the same name. Instantiate your class and use it in a program that prompts the user for a range and number of picks required. The program should then generate an appropriate set of values for the user as well as inform the user of her chances of winning (more like her chances of not losing!).

Hints:
1) create a private method in your class to calculate factorials. Using double precision numbers in this method will allow your code to easily handle a range of 100.

2) Recall that mathematically, *n! = n * (n-1) * (n-2) *... all the way down to 1*. So, for example, 5! = 5*4*3*2*1 = 120. By observation, *n! = n * (n-1)!*. Your factorial function will be a good opportunity to practice your recursion (though using recursion is not a requirement).

3) Take the time to get this code right. We will be re-using it to make a graphical quick –pick app.

4) This assignment can be completed using a struct rather than a class. This is acceptable.