1. Our web app has 1 fancy requirement: buttons with rounded corners at top-left and bottom-right. Provide a code sample with HTML5/CSS.
Included in WebInterviewQuestionsPaypros.html

2. Later on, a new requirement is to support gradient background on the buttons. How do you do it? Please provide a code sample with HTML5/CSS.
Included in WebInterviewQuestionsPaypros.html

3. Part 1:

```
function foo(a) {
        var tmp = 5;
        function bar(b) {alert(a + b + (++tmp));}
        return bar;
}
foo(3)(10); // A
foo(3)(10); // B
```

Part 2:

```
function foo(a) {
        var tmp = 5;
        function bar(b) {alert(a + b + (++tmp));}
        return bar;
}
var ref = foo(3);
ref(10); // C
ref(10); // D
```

a) What are the values that A and B alert? Are they same? Why?
The values of A and B are 19. They are the same because foo(3)(10) is copying by value.

b) What are the values that C and D alert? Are they same? Why?
The value of C is 19 and the value of D is 20. They are not the same because var ref is a static variable and ref(10) is copying by reference; meaning that the return value of the method foo is being saved to the ref variable.

c) What is the difference between part 1 and part2?
Part 1 is copying by value and part 2 is copying by reference. Or in other words, part2 is saving the return value of the method foo and increments by 1 each time it is called.

4. What's Ajax? When do you need to use it? Please give us an example you used Ajax to solve a problem.
I do not have any experience with Ajax.

5. (Optional) Implement a 2-level menu using JavaScript and CSS.

Algorithm
6. Given an unsorted integer list, find the closest number to a given target, where closest means smaller than target with smallest difference. What the best running time and why.
I have provided a C# solution in the folder "UnsortedClosestToGivenTarget." It uses linear search with the best, average, and worst running time of O(n). This is because the algorithm searches through the unsorted integer list once and as it is iterating through, it compares and updates the current closest value to the target value. I have also made an assumption that "closest" means smaller than <u>or equal to</u> the target value with the smallest difference being 0. If the question was really asking for the smaller than target value with the smallest difference as it says on the question sheet I can simply change line 43 of the program from >= to >.

If I had to use the algorithm repetitively then there would be an advantage to sort the list first (eg. Quicksort or Mergesort) and then repetitively search for the value closest to the target values using binary search. Quicksort and Mergesort have running times of O(n log n) which would be considered as a constant running time since sorting algorithm would only have to be ran once. Binary search has a running time of O(log n).

7. Given a list of chars and corresponding probability, we want a random method so that the probability of producing a specific char is exactly same as in the probability list.
I have provided a C# solution in the folder "CharsCorrespondingProbability." The program uses a Dictionary to represent the chars with corresponding probabilities. I designed the program so that the probabilities do not have to add up to 1 or 100. The program will calculate the probability by dividing the corresponding "probabilities" by the total (the value of the probabilities added up).

The program uses an array of size chars+1 and fills it up with the totals of the consecutive probabilities and uses Random class to get a number in between 0 and the total (all of totals of the consecutive probabilities added up). At the same time, the program creates another Dictionary to match the chars with the corresponding totals. Then it uses binary search to find the closest value, rounding up to the next element in the array. And finally, the program looks up the value in the Dictionary we just created to find and returns the corresponding char.

The running time of this method is O(log n); however, if the specification did not care about the memory usage or if the probabilities did not have too many significant digits, another approach can be used to improve the running time.

For example, let say that a=0.2, b=0.3, c=0.1, d=0.4.
```
-Create an array: char[] prob = new char[10];
-Populate in the array:  prob[0]~prob[1]=0.2
                         prob[2]~prob[4]=0.3
                                 prob[5]=0.1
                         prob[6]~prob[9]=0.4
-then use prob[rand() % 10] to get a random char corresponding to the probabilities.
```