1. Insertion sorting is one by one remove the element from the original array and insert it into the proper location which is larger than the previous value and smaller than next value in the array.  
   For sorting the array 52314.  
   2 is inserted before 5, resulting in 25314  
   3 is inserted between 2 and 5, resulting in 23514  
   1 is inserted at the start, resulting in 12354  
   4 is inserted between 3 and 5, 12345.  
   End of sorting.  
   Generate 1024 0~10000 random values and put into a 1D array. Visualize the insertion sorting process with waveform graph and slightly delay.
2. Selection sorting is making only one exchange for every list pass through. On every pass through, the selection sort searches for the largest number and places it at first location by swapping.  
   On the second pass through, the second largest is found and placed in its location after the largest by swapping… and so on…  
   The final item would be placed in (n-1) steps.  
   Generate 1024 0~10000 random values and put them into a 1D array. Visualize the sorting process on a waveform graph with slightly delay.  
   First Pass  
   (68,2,93,45) -> (68,2,45,93), 93<->45 swap  
   Second Pass  
   (68,2,45,93) -> (45,2,68,93), 68<->45 swap  
   Final Pass  
   (45,2,68,93) -> (2,45,68,93), 2<->45 swap
3. Write a **BenchmarkSorting.vi** to benchmark the performance of bubble (you have done this in HW2), selection, and insertion sorting.  
   The vi will generate 1024 0~10000 random values first and feed the same dataset to the three sorting algorithm.  
   **Package the three sorting algorithm into subVIs (named bubble sorting, selection sorting and insertion sorting) and run them one by one.** Show the time required to finish sorting with the 3 different sorting methods individually.  
   **(Note1: please remove all wait function inside the loops when sorting)**  
   **(Note2: you need to hand in four VIs in question 3 )**
4. Write a count down timer VI with 3 numerical display controls in hh:mm:ss format to let user enter the time to counting down from. The VI has a Boolean button start/stop controlling start or stop (pause) count down timer.  
   Use local variables (do not use shift registers) to read the 3 numerical controls and update the count down time per second. Beeps for 3 seconds by using beep.vi when counting down time reach 00:00:00. The time value then count up and showing a negative sign with a rectangular LED in front of hh:mm:ss.
5. Write a VI to test the reaction time from ‘you see the LED is on’ to ‘you press the “Stop timer” button’. The VI repeats the following two steps until you press the “Stop program” button.  
   (1) Random delay for 5 ~ 8 seconds and LED is off.  
   (2) Turn LED on. User press the “stop timer” button when he see the LED on. Show the reaction time with a numerical indicator and also record all test results in a waveform chart before the program stop.  
   (Please use sequence for timing and local variable for the LED control.)