1. (10 points) For each of the following claims, determine whether they are true or false. Justify your determination (show your work). If the claim is false, state the correct asymptotic relationship as O, Θ, or Ω. Unless otherwise specified, lg is log_2.

(a)	$n+1$	$=$	$O(n^4)$
(b)	2^{2n}	$=$	$O(2^n)$
(c)	2^n	$=$	$\Theta(2^{n+7})$
(d)	1	$=$	$O(1/n)$
(e)	$\ln^2 n$	$=$	$\Theta(\lg^2 n)$
(f)	n^2+2n-4	$=$	$\Omega(n^2)$
(g)	3^{3n}	$=$	$\Theta(9^n)$
(h)	2^{n+1}	$=$	$\Theta(2^{n \lg n})$
(i)	$\sqrt{n}$	$=$	$O(\lg n)$
(j)	10^{100}	$=$	$\Theta(1)$

2. (25 points) Asymptotic relations like O, Ω, and Θ represent relationships between *functions*, and these relationships are transitive. That is, if some $f(n) = \Omega(g(n))$, and $g(n) = \Omega(h(n))$, then it is also true that $f(n) = \Omega(h(n))$. This means that we can sort *functions* by their asymptotic growth.[1]

   Sort the following *functions* by order of asymptotic growth such that the final arrangement of functions $g_1, g_2, \ldots, g_{12}$ satisfies the ordering constraint $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, $\ldots$, $g_{11} = \Omega(g_{12})$.

n	n^2	$(\sqrt{2})^{\lg n}$	$2^{\lg n}$	$n!$	$(\lg n)!$	$\left(\frac{3}{2}\right)^n$	$n^{1/\lg n}$	$n \lg n$	$\lg(n!)$	e^n	42

   Give the final sorted list and identify which pair(s) functions $f(n), g(n)$, if any, are in the same equivalence class, i.e., $f(n) = \Theta(g(n))$.

[1] The notion of sorting is entirely general: so long as you can define a pairwise comparison operator for a set of objects $\mathcal{S}$ that is transitive, then you can sort the things in $\mathcal{S}$. For instance, for strings, we use a comparison based on lexical ordering to sort them. Furthermore, we can use any sorting algorithm to sort $\mathcal{S}$, by simply changing the comparison operators $>$, $<$, etc. to have a meaning appropriate for $\mathcal{S}$. For instance, using Ω, O, and Θ, you could apply QuickSort or MergeSort to the functions here to obtain the sorted list.

3. (30 points) Professor Dumbledore needs your help optimizing the Hogwarts budget. You'll be given an array A of exchange rates for muggle money and wizard coins, expressed as integers. Your task is to help Dumbledore maximize the payoff by buying at some time i and selling at a future time $j > i$, such that both $A[j] > A[i]$ and the corresponding difference of $A[j] - A[i]$ is as large as possible.

   For example, let $A = [8, 9, 3, 4, 14, 12, 15, 19, 7, 8, 12, 11]$. If we buy one stock at time $i = 2$ (assuming 0 indexing) with $A[i] = 3$ and $j = 7$ with $A[j] = 19$, Hogwarts gets an income of $19 - 3 = 16$ coins.

   (a) Consider the pseudocode below that takes as input an array A of size n:

   ```
   makeWizardMoney(A) :
       maxProfitSoFar = 0
       for i = 0 to length(A)-1 {
           for j = i+1 to length(A) {
               coins = A[j] - A[i]
               if (coins > maxProfitSoFar) { maxProfitSoFar = coins }
       }}
       return maxProfitSoFar
   ```

   What is the running time complexity of the procedure above? Write your answer as a Θ bound in terms of n.

   (b) Explain (1–2 sentences) under what conditions on the contents of A the `makeWizardMoney` algorithm will return 0 gold.

   (c) Professor Dumbledore knows you know that `makeWizardMoney` is wildly inefficient. He suggests you write a function to make a new array M of size n such that

   $$M[i] = \min_{0 \le j \le i} A[j] \ .$$

   That is, $M[i]$ gives the minimum value in the subarray of $A[0..i]$.

   What is the running time complexity of the pseudocode to create the array M? Write your answer as a Θ bound in terms of n.

   (d) Use the template code provided and implement the function described in (3c) to compute the maximum coin difference in time $\Theta(n)$.

   (e) Use the template code provided to determine and compare the runtimes for the functions in 2a and 2d. Explain your findings.

4. (15 points) Consider the problem of finding the maximum element in a given array. The input is a sequence of n numbers $A = \langle a_1, a_2, \ldots, a_n \rangle$. The output is an index i such that $a_i \geq a_1 \geq a_2 \geq ... \geq a_n$.

   (a) Write pseudocode for a simple maximum element search algorithm, which will scan through the input sequence A, and return the index of the last occurrence of the maximum element.

   (b) Using a loop invariant, prove that your algorithm is correct. Be sure that your loop invariant and proof covers the initialization, maintenance, and termination conditions.

5. (20 points) Crabbe and Goyle are arguing about binary search. Goyle writes the following pseudocode on the board, which he claims implements a binary search for a target value `v` within an input array `A` containing n elements sorted in ascending order.

```
bSearch(A, v) {
   return binarySearch(A, 1, n-1, v)
}

binarySearch(A, l, r, v) {
   if l <= r then return -1
   m = floor( (l + r)/2 )
   if A[m] == v then return m
   if A[m] > v then
     return binarySearch(A, m+1, r, v)
     else return binarySearch(A, l, m-1, v)
```

   (a) Help Crabbe determine whether this code performs a correct binary search. If it does, prove to Goyle that the algorithm is correct. If it does not, state the bug(s), fix the line(s) of code that are incorrect, and then prove to Goyle that your fixed algorithm is correct.

   (b) Goyle tells Crabbe that binary search is efficient because, at worst, it divides the remaining problem size in half at each step. In response Crabbe claims that fournary search, which would divide the remaining array A into fourths at each step, would be *way more* efficient asymptotically. Explain who is correct and why.