1. (30 points) For each of the following two functions f(n) and g(n), indicate whether f = O(g), or f = Ω(g) or both (in which case f = Θ(g)).

a.	f(n) = n − 100 and g(n) = n − 200	f = Θ(g)
b.	f(n) = n^1/2 and g(n) = n^2/3	f = O(g)
c.	f(n) = 100n + lg n and g(n) = n + (lg n) ^2	f = O(g)
d.	f(n) = n lg n and g(n) = 10n lg (10n)	f = O(g)
e.	f(n) = 10 lg n and g(n) = lg (n^2)	f = Ω(g)
f.	f(n) = n^2/ lg n and g(n) = n (lg n) ^2	f = Ω(g)
g.	f(n) = n^0.1 and g(n) = (lg n) ^10	f = Ω(g)
h.	f(n) = √ n and g(n) = (lg n)^3	f = O(g)
i.	f(n) = n2^n and g(n) = 3^n	f = O(g)
j.	f(n) = 2^n and g(n) = 2^n+1	f = O(g)

2. Given a collection of n nuts and a collection of n bolts, arranged in an increasing order of size, give an O(n) time algorithm to check if there is a nut and a bolt that have the same size. The sizes of the nuts and bolts are stored in the sorted arrays NUTS [1...n] and BOLTS [1...n], respectively. Your algorithm can stop as soon as it finds a single match (i.e., you do not need to report all matches).

```
i =0, j =0;
While (i <n and j <n)
Do
        if (NUTS[i] == BOLTS[j]) then
                //Match found
                return (True);
        else if (NUTS[i] < BOLTS[j]) then
                i=i+1;
        else j=j+1;
        end
end
return (False);
```

3. Let A [1...n] be an array of distinct positive integers, and let t be a positive integer
   a. Assuming that A is sorted, show that in O(n) time it can be decided if A contains two distinct elements x and y such that x + y = t.

      two_sum

      ```
      var ht = []; //Hash Table
      for i = 0 to n do
          if (A[i] > t) then continue; end
          diff = t – A[i];
          if (ht[diff]! = undefined) then
                  return [diff, A[i]];
          end
          ht[A[i]] = A[i];
      end
      return ('Not Found');
      ```

   b. Use part (a) to show that the following problem, referred to as the 3-Sum problem, can be solved in O (n^2) time: 3-Sum Given an array A [1...n] of distinct positive integers that is not (necessarily) sorted, and a positive integer t, determine whether or not there are three distinct elements x, y, z in A such that x + y + z = t

      Without using two-sum

      ```
      second=0; third=0;

      sort A

      for i = 0 to n-2 do

          second=i+1;

          third=n-1;

          while(second<third) do

                  if(A[i]+A[second]+A[third] == t) then

                          return [ A[i], A[second], A[third]];

                  else if (A[i]+A[second]+A[third] < t) then

                          second=second+1;

                  else third= third-1;

                  end
      ```

```
    end
end
return ('Not Found');
```

With using two-sum

```
For i = 0 to n-2
    if (A[i] > t) then continue; end
    diff = t – A[i];
    match = two_sum (A. slice(i+1), diff); // send array from i+1 element and diff
    //to the 2-sum function to find the next two elements using the logic from (a)
    if (match) then
            return [A[i]] + match;
    end
end
return ("Not Found");
```

4. Let A [1...n] be an array of positive integers (A is not sorted). Pinocchio claims that there exists an O(n)-time algorithm that decides if there are two integers in A whose sum is 1000. Is Pinocchio right, or will his nose grow? If you say Pinocchio is right, explain how it can be done in O(n) time; otherwise, argue why it is impossible.

Pinocchio is right, it can be done in O(n) time using the two-sum method using Hash table discussed in 3(a), this solution works for unsorted arrays as well

```
var ht = []; //Hash Table
for i = 0 to n do
    if (A[i] > 1000) then continue; end
    diff = 1000 – A[i];
    if (ht[diff]! = undefined) then
            return [diff, A[i]];
    end
    ht[A[i]] = A[i];
end
```

return (‘Not Found’);