1. (5 points) Using the reverse of the algorithm in Assignment 10 Decode the following phrase

   it:1,4,7
   is:2,5,8
   what:3
   until:6
   not :9

Answer: It is what it is until it is not

2. (5 points) What is the value of t after the following code?

```
var a = []; var s = "COP3530";var t = "";
for (var i = 0; i < s.length; i++) a.push(s.charAt(i));
for (var i = 0; i < s.length; i++) t += a.pop();
```

   Answer: 0353POC

3. (5 points) Calculate the following contents placed on a Stack Computer.

   2, 3, 5, *, *, 10, /

   Answer: 3 2 3 5 -> * –> 2 15 -> * -> 30 -> 30 10 -> / -> 3

4. (10 points) In hash tables linear probing is more efficient in element lookup than chaining except when?

   Answer: When you have a high load factor (your table is nearly full), poor hash function good for partial credit (5 points)

5. (5 points) A full expression tree evaluates to [2, 3, +, 15, 10, -, *] which gives the results 25 when using left-first post order depth first traversal when applied to a calculation stack. What result will it give using right first post order depth first traversal?

   answer: -25

```
        *
    +       -
  2   3   15  10
```

   The 15 and 10 will reverse subtraction and give –5 * 5

6. (25 points) Cyclic Redundancy Check (CRC) is used to validate data transmission. It is a very simple algorithm that allows you to check if sent information over a network matches the received information over the network – essentially error checking. Both are run through the

algorithm and the results are compared. Because the check value is fixed length it can also be used as a hash function. The Adler-32 is a specific algorithm that can be used for hashing or CRC. Write a Javascript algorithm to implement the Adler 32 algorithm with user input.

You can use the starter shell program at https://jsfiddle.net/reaglin/mymbgqcv/
You must use fork to copy my shell to your fiddle account if you choose to use it.

```
function adler32(s){
  var MOD_ADLER = 65521;

  var a = 1;
  var b = 0;

  // Process each byte of the data in order
  for (var i = 0; i < s.length; i++) {
    a = (a + s.charCodeAt(i)) % MOD_ADLER;
    b = (b + a) % MOD_ADLER;
  }
  return (b << 16) | a;   // 122093998 or  74701ae
}
```

7. (5 points) What is the Adler32 hash value of COP3530 ?

   Answer: `122093998 or 74701ae (in hex)`

8. (5 points) A tree structure is designed for a specific application where a single node holds data until it gets to 5 data elements. It then keeps the middle value spawning a leaf with lower values (left leaf) and a leaf with higher values (right leaf). For example; if the values held by one node were 1, 3, 9, and 13 and a 7 was added the node would retain the 7 as its value and spawn a left leaf with 1 and 3 and a right leaf with 9 and 13 as their node content. What is the maximum number of values a tree of depth 4 could hold? Non-leaf nodes only hold 1 value, leaf nodes can hold up to 4.

   answer: 39

   Level 1: 1
   Level 2: 2
   Level 3: 4
   Level 4: 32, 8 nodes of 4 each
   Total: 39 nodes

9. (10 points) A full n-ary (or k-ary) tree can have the following number of leaves. Check all that are correct

4 yes
7 no
9 yes
16 no
23 yes

10. (5 points) You are asked to select a sorting algorithm for a specific application where you know only one element is not in the correct place. Of the following which is the best if you are dealing with lots of elements and efficiency is extremely important?

    Answer: Insertion Sort

11. (20 points) Using only Stack commands (**push** and **pop**) implement a Queue. You can use more than one stack but your final code must implement a single object that implements **Enqueue** and **Dequeue** operations. Note internally you must implement the enqueue and dequeue operations using only push and pop commands of the internal objects of the Queue.

    You may start with my base code for this programming problem it is at https://jsfiddle.net/reaglin/y2uxL2yn/

    If you use my code make sure you are logged in to JSFiddle and you Fork the code over to your own. The structure of my code will make understanding what to do here a lot easier. Note – this is NOT the most efficient way to create a Queue.

```
this.stack = [];

this.enqueue = function(value) {
   this.stack.push(value);
}

this.reverse = function(){
   this.stack.reverse();
}

this.dequeue = function(){
  this.reverse();
  this.stack.pop();
  this.reverse();
}
```

Alternately you can implement with 2 stacks that will allow you to reverse a Stack. Reversing the contents of a String as a Stack is demonstrated in Question 2.