Problem Description 1

(i)What are the contents of the two arrays, x[] and y[] after the following code fragment executes?

```
int[] x = {23, 55, 83, 19};
 int[] y = {36, 78, 12, 24};
for(int a = 0; a < x.length; a++)
{
        int temp = x[a];
        x[a] = y[a]; y[a] = x[a];
 }
```

(ii) The following code is the beginning of a class that stores Strings in a singly linked list. Complete the method named, "add()". The new Node should be added to the beginning of the list. DO NOT re-write the entire class, just the method, add(String s).

```
public class LwtLinkedList
{
        class Node {
                String data;
                Node next;
                public Node( String d, Node nxt )
                {
                        data = d; next = nxt;
                }
                public String getData( )
                {
                        return data;
                }
        }
        private Node first;
        public void add( String s )
        {
                // TODO: add code here to add the String parameter
                // s to the beginning of the list
        }
}
```

(iii) Implement a class Car. The class Car should have the following fields:
brand (Honda, Ford, Toyota etc.)
type (sedan, SUV, truck, etc.)
year
milage (current milage)
class Car { //Complete this class }
Create an array of cars. const cars = [//complete the array];
Implement a function that searchCarByBrand(cars, brand). This function accepts an array of cars and a brand name. It will return an array of cars whose brand matches the input brand name.
function searchCarByBrand(cars, brand)

{ //Complete the function }

Problem Description 2

Credit Card Validation

You're starting your own credit card business. You've come up with a new way to validate credit cards with a simple function called `validateCreditCard` that returns `true` or `false`.

Here are the rules for a valid number:

- Number must be 16 digits, all of them must be numbers
- You must have at least two different digits represented (all of the digits cannot be the same) the
- final digit must be even
- The sum of all the digits must be greater than 16

The following credit card numbers are valid:

- `9999-9999-8888-0000`
- `6666-6666-6666-1666`

The following credit card numbers are invalid:

- `a923-3211-9c01-1112` *invalid characters*
- `4444-4444-4444-4444` *only one type of number*
- `1111-1111-1111-1110` *sum less than 16*
- `6666-6666-6666-6661` *odd final number*

You will need to the realine approach that is introduced in the discussion to allow your users to enter credit card numbers and on change, validate if the credit card is valid and display a message to the user accordingly. **3 points extra:** Make your credit card scheme even more advanced! What are the rules, and what are some numbers that pass or fail? Ideas: check the expiration date!

Problem Description 3

Encryption is the process of obscuring information to make it unreadable without special knowledge. For centuries, people have devised schemes to encrypt messages - some better than others - but the advent of the computer and the Internet revolutionized the field. These days, it's hard not to encounter some sort of encryption, whether you are buying something online or logging into a shared computer system.
Encryption lets you share information with other trusted people, without fear of disclosure.

A cipher is an algorithm for performing encryption (and the reverse, decryption). The original information is called plaintext. After it is encrypted, it is called ciphertext. The ciphertext message contains all the information of the plaintext message, but it is not in a format readable by a human or computer without the proper mechanism to decrypt it; it should resemble random gibberish to those for whom it is not intended.

A cipher usually depends on a piece of auxiliary information, called a key. The key is incorporated into the encryption process; the same plaintext encrypted with two different keys should have two different ciphertexts. Without the key, it should be difficult to decrypt the resulting ciphertext into readable plaintext.

This assignment will deal with a well-known (though not very secure) encryption method called the Caesar cipher. Some vocabulary to get you started on this problem:

Encryption - the process of obscuring or encoding messages to make them unreadable until they are decrypted
Decryption - making encrypted messages readable again by decoding them
Cipher - algorithm for performing encryption and decryption
Plaintext - the original message
Ciphertext - the encrypted message. Note: a ciphertext still contains all of the original message information, even if it looks like gibberish. The Caesar Cipher

The idea of the Caesar Cipher is to pick an integer and shift every letter of your message by that integer. In other words, suppose the shift is k . Then, all instances of the i-th letter of the alphabet that appear in the plaintext should become the (i+k)-th letter of the alphabet in the ciphertext. You will need to be careful with the case in which i + k > 26 (the length of the alphabet). Here is what the whole alphabet looks like shifted three spots to the right:

Original: a b c d e f g h i j k l m n o p q r s t u v w x y z
3-shift: d e f g h i j k l m n o p q r s t u v w x y z a b c
Using the above key, we can quickly translate the message "happy" to "kdssb" (note how the 3-shifted alphabet wraps around at the end, so x -> a, y -> b, and z -> c).

Note!! We are using the English alphabet for this problem - that is, the following letters in the following order:

We will treat uppercase and lowercase letters individually so that uppercase letters are always mapped to an uppercase letter, and lowercase letters are always mapped to a lowercase letter. If an uppercase letter maps to "A", then the same lowercase letter should map to "a". Punctuation and spaces should be retained and not changed. For example, a plaintext message with a comma should have a corresponding ciphertext with a comma in the same position.

plaintext	shift	ciphertext
'abcdef'	2	'cdefgh'
'Hello, World!'	5	'Mjqqt, Btwqi!'

Extra Credit - 2 pts

Implement a function that breaks the Ceasar cipher using the word counting technique.

Output example of a cipher breaker:

```
Test string: [Jhu fvb ylhk aopz?] -- Cipher: [ 7 ] -- Certainty: [ 100.00% ] -- Text: [Can you read this?]
Test string: [Mxqj q coijuho!] -- Cipher: [ 16 ] -- Certainty: [ 100.00% ] -- Text: [What a mystery!]
Test string: [Umbym ni umbym, xomn ni xomn.] -- Cipher: [ 20 ] -- Certainty: [ 100.00% ] -- Text: [Ashes to ashes, dust to dust.]
```

Problem Description 4

Design an ***Essay*** class that extends the ***GradedActivity*** class presented in chapter 10 (edition 3) of the Gaddis text for the course. The ***Essay*** class should determine the grade a student receives for an essay. The student's essay score can be up to 100 and is determined in the following manner:

- Grammar: 30 points
- Spelling: 20 points
- Correct length: 20 points
- Content: 30 points

In a course, a teacher gives the following tests and assignments:

- A lab activity that is observed by the teacher and assigned a numeric score
- A pass/fail exam that has 10 questions, the minimum passing score is 70
- An essay that is assigned a numeric score
- A final exam that has 50 questions

Write a class named ***CourseGrades***. The class should have a ***GradedActivity*** array named ***grades*** as a field. The array should have four elements, one for each of the assignments previously described. The class should have the following methods:

setLab – This method should accept a ***GradedActivity*** object as its argument. This object should already hold the student's score for the lab activity. Element 0 of the ***grades*** field should reference this object.

setPassFailExam – This method should accept a ***PassFailExam*** object as its argument. This object should already hold the student's score for the pass/fail exam. Element 1 of the ***grades*** field should reference this object.

setEssay – This method should accept as ***Essay*** object as its argument. See Programming Challenge 4 for the ***Essay*** class. This object should already hold the student's score for the essay. Element 2 of the ***grades*** field should reference this object.

setFinalExam – This method should accept a ***FinalExam*** object as its argument. This object should already hold the student's score for the final. Exam. Element 3 of the ***grades*** field should reference this object.

toString – This method should return a string that contains the numeric scores and grades for each element in the ***grades*** array.

Modify the ***CoursesGrades*** class created in Programming Challenge 5 so it implements the following interface:

```
Public interface Analyzable {
     Double getAverage();
     GradedActivity getHighest();
     GradedActivity getLowest();
}
```

The ***getAverage*** method should return the average of the numeric scores stored in the ***grades*** array. The ***getHighest*** method should return a reference to the element of the ***grades*** array that has the highest numeric score. The ***getLowest*** method should return a reference to the element of the ***grades*** array that has the lowest numeric score.