1. What is printed when the following program is run?

```java
public class Example {
    public static void main (String [] args) {
        String word = "lurcvkty";
        char[] array = new char[word.length()];
        char achar = '*';
        for (int j = 0; j < array.length; j++) {
            array[j] = word.charAt(j);
        }
        System.out.println("before:");
        for (int j = 0; j < array.length; j++) {
            System.out.print(array[j]);
        }
        System.out.println();
        aMethod (array, achar);
        System.out.println("after:");
        for (int j = 0; j < array.length; j++) {
            System.out.print(array[j]);
        }
        System.out.println();
    }
    public static void aMethod (char[] arr, char ch) {
        int place = arr.length -2;
        for (int i = arr.length - 2; i > 1; i = i -2){
            arr[i] = ch;
        }
    }
}
```

Output:

```
before:
lurcvkty
after:
lu*c*k*y
```
Objects, Classes, and Arrays

2. The `Coins` class has the following instance variables:

```java
public class Coins {
    private String type;   // penny, nickel, dime, or quarter. Lowercase!
    private int value;     // in cents - 5 for nickel, 10 for dime, etc.
    private int count;     // number of coins of that type
}
```

a. Write an instance method for the `Coins` class called `setValue()` that sets the `value` instance variable of the `Coins` object that invokes it to 1 for a penny, 5 for a nickel, 10 for a dime, or 25 for a quarter, as identified by the `type` instance variable of the invoking object. You may assume that the value of the `type` instance variable is always stored as a lowercase value.

```java
public void setValue(){
    if ( this.type.equals("penny") )
        this.value = 1;
    else if ( this.type.equals("nickel") )
        this.value = 5;
    else if ( this.type.equals("dime") )
        this.value = 10;
    else if ( this.type.equals("quarter") )
        this.value = 25;
    else
        this.value = 0;
}
```

b. Write a 2-arg constructor for the `Coins` class that sets the `type` instance variable to the lowercase equivalent of the first argument, the `count` instance variable to the second argument, and uses the `setValue()` instance method from part a to set the `value` instance variable.

```java
public Coins (String aType, int aCount)
    this.type = aType.toLowerCase();
    this.count = aCount;
    this.setValue();
}
```

c. Write a `getValue()` accessor method for the `Coins` class that returns the total value, in change, of the `Coins` object passed to it. For example, if a `Coins` object with a `type` instance variable of “nickel” and a `count` instance variable of 3 invoked this method, then a value of 15 should be returned.

```java
public int getValue (){  
    return this.value*this.count;
}
```
d. Complete the `println` code in the following `UseCoins` class so that the value of `coin1` is printed using the `getValue()` accessor method from part c.

```java
public class UseCoins {
    public static void main(String[] args) {
        coin1 = new Coins("nickel", 3);
        System.out.println("Total value \+ coin1.getValue();
    }
}
```

e. The `Wallet` class defines an array of `Coin` objects. Add a static method to this class called `totalValue()` that is passed an array of `Coin` objects (as shown) and returns the total value, in change, of all of the objects in that array.

```java
public class Wallet {
    public static void main(String[] args) {
        Coins[] coins = new Coins[4];
coins[0] = new Coins("nickel", 14);
coins[1] = new Coins("penny", 12);
coins[2] = new Coins("quarter", 10);
coins[3] = new Coins("dime", 4);
        System.out.println("Total value = " + totalValue(coins));
    }
    // add code here

    public static int totalValue ( Coins[] cn){
        int total = 0;
        for (int i = 0; i < cn.length; i++)
            total = total + cn[i].getValue();
        return total;
    }
}
```
3. Define a new class `Course` to represent course information in the following way: each course object is characterized by a **name**, **maximum enrollment** and is assigned a **classroom type**, that is either “computerized” or “regular”.

Your class must define a separate instance variable to represent **name**, **maximum enrollment** and **classroom type**.

You do not need to define any methods, but you have to define a 2-arg constructor that is passed course name and maximum enrollment. The constructor should assign the values to the instance variables representing course name and maximum enrollment to the values passed in through the parameters, and should assign the value of instance variable representing the classroom type according to the following rule: all classes with a name that starts with “CS” or “IT” must be assigned a computerized classroom, all others – a regular classroom.

```java
public class Course{
    private String name;
    private int maxEnroll;
    private String type;

    public Course( String coursename, int max){
        this.name = coursename ;
        this.maxEnroll = max;

        if (courseName.indexOf("CS") == 0 ||
            courseName.indexOf("IT") == 0)
            this.type = "computerized";
        else
            this.type = "regular";
    }
}
```

Write a code segment that declares a variable of class type `Course`, creates an object of class `Course` representing “CS180” with maximum enrollment of 25 students.

```java
Course cs180 = new Course("CS 180", 25);
```

Write a code segment that declares a 15-element array of type `Course`, and assigns the last element of the array to an object representing “HI130” with enrollment of 30 students.

```java
Course[] myclasses = new Course[15];
myclasses[14] = new Course("HI 130", 30);
```
4. Write a static method called `aveArray` that is passed an integer array as an input. It must first determine the average of the values in the array. Then it determines how many values in the array are greater than the average, and returns a new array containing only those values.

For example, if passed the following array:

\{13, 10, 15, 11, 12\}

then the array to be returned would contain: \{13, 15\} (since the average is 12.2)

Algorithm:
- Traverse the array to compute the average
- Traverse the array to count how many numbers above average
- Create an array of size equal to number of numbers above average
- Traverse the array to copy numbers above average into new array.

```java
public static int[] aveArray (int[] a) {
    int numbersAboveAve = 0;
    double average;  
    double sum = 0;

    //compute the average
    for (int i = 0; i < a.length; i++){
        sum += a[i];
    }
    average = sum/ a.length;

    //count how many numbers are above the average
    for (int i = 0; i < a.length; i++){
        if (a[i] > average){
            numbersAboveAve++;
        }
    }

    // create a new array of size  numbersAboveAve
    int[] newarray = new int[numbersAboveAve];

    // copy numbers above average from a to newarray.
    int j = 0; // index of the newarray
    for (int i = 0; i < a.length; i++)
        if (a[i] > average){
            newarray[j] = a [i];
            j++;  // move to the next position in the newarray
        }
    return newarray;
}
```