```
public class String
{
    // Returns the length of this string.
    public int length ()

    // Returns a substring of this string that begins at the specified
    // beginIndex and extends to the character at index endIndex - 1.
    public String substring (int beginIndex, int endIndex)

    // Returns a substring of this string that begins at the specified
    // beginIndex and extends to the end of the string.
    public String substring (int beginIndex)

    // Returns position of the first occurrence of str, returns -1 if not found
    public int indexOf (String str)

    // Returns the position of the first occurrence of str starting from
    // start position,  returns -1 if str is not found
    public int indexOf (String str, int start)

    // returns character at position index
    public char charAt(int index)

    // returns the string as an array of characters
    public char [] toCharArray()
}
```

PROBLEM 1 : (***What is the output?:*** **(15 pts))**

PART A ***(12 pts)***:

Consider the following Java code. List the output from this code below.

```
String phrase = "Dukeisthisone";
System.out.println(phrase.charAt(2));

int pos = phrase.indexOf("is", 6);
String word = phrase.substring(pos+1, pos+4);
System.out.println(word);

String item = "red";
item = "to" + item;
System.out.println(item);
```

```
String dna = "atgcat";
System.out.println(dna.length());

int count = 0;
for (char ch: dna.toCharArray())
{
   if (ch == 'a')
   {
      count = count + 1;
      System.out.println(count);
    }
}
System.out.println(count);
```

List the output here:

PART B ***(3 pts)***:

Consider the following java code. Assume that *manyStrings* is an array of type String.

```
int sum = 0;
for (String item_from: manyStrings)
{
   if (item_from.indexOf("cat") > -1)
   {
      sum = sum + 1;
   }
}
```

Explain what this code does.

PROBLEM 2 : (*Pick out the pieces:* (8 pts))

Complete the method *extract* that is given one String parameter called *dna*. This method returns a new string that is made up of only the *a*'s and *g*'s from the dna string, in the same order they appear in dna.

For example, `extract("atgcattagcg")` would return the string `"agaagg"`.

```
public String extract(String dna)
{

}
```

PROBLEM 3 : (*How many big ones?:* (8 pts))

Complete the method *NumberLargeEnough* that is given two parameters, an array of Strings called *words*, and an integer called *num*. This method returns the number of strings in the array *words* that have more than *num* characters.

For example, assume the array words contains the following strings in this order: "football", "soccer", "basketball", "fencing", "golf", "track". Then the call `NumberLargeEnough(words, 6)` would return `3`, as only three of the strings are of length greater than 6.

```
public int NumberLargeEnough (String [] words, int num)
{

}
```

PROBLEM 4 : (*An average guy or gal:* (8 pts))

Complete the method *average* that is given one parameter, an array of integers called *values*. This method returns the average of all the numbers in values.

For example, suppose values contains the following numbers in this order: 10, 10, 6, 8. Then the call `average(values)` would return the number `8.5`.

```
public double average (int [] values)
{

}
```