Problem 1: Arrays and Functions (15 points)

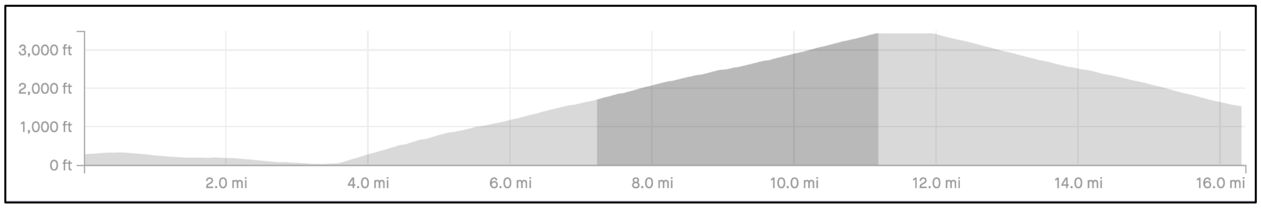
Cyclists training for biking competitions such as the Tour de France, make it a point to train on hills that are comparable to those they will encounter in the race. In cycling, hill climbs are classified based on their distance and their grade (% steepness).
To be a categorized climb, the following conditions must exist:

1. The minimum steepness must be at least 3% grade.
2. The minimum length of the climb must be at least 500 meters. (*Note that 1 mile = 1609.334 meters)*
3. The total calculated points must be 8000 or more. Points are calculated by multiplying the percent by the length of climb in meters. Example: If segment is 2000 meters in length and % grade is 7%, point calculation is 2000 * 7 = 14,000.

Climb classifications are:

- Cat 4: at least 8,000 pts
- Cat 3: at least 16,000 pts
- Cat 2: at least 32,000 pts
- Cat 1: at least 64,000 pts
- HC: at least 80,000 pts *(**HC – Hors Categorie -** From the French term meaning beyond categorizing. The toughest of the tough. The longest or steepest climbs, often both combined.)*

An elevation graph of the Alpe d'Huez route, a famous route among cyclists, is below:



The table below gives information about climb segments.

Length of segment (miles)	0.52mi	0.92mi	7.59mi	1.85mi	3.93mi	5.8mi	0.7	0.3
Steepness grade (%)	1.9%	10.1%	8.5	9.3%	8.3%	7.5%	3.0%	9%
Points	0 (percent < 3%)	14,953.93	103,826.18	27,688.59	52,494.87	70,006.03	3379.60	0 (distance too short)
Category	Not Categorized	CAT 4	HC	CAT 3	CAT 2	CAT 1	Not Categorized	Not Categorized

A program to answer the following questions has declared and initialized two **parallel** arrays:

```
double[] catLimits = {80000,64000,32000,16000,8000};
String[] categories = {"HC", "CAT 1", "CAT 2", "CAT 3", "CAT 4"};
```

Parallel Arrays Note: *a climb >= 80000 is categorized as "HC"; a climb >= 64000 and less than 80000 is categorized as "CAT 1"; a climb >= 32000 and less than 64000 is categorized as "CAT 2"; a climb >= 16000 and less than 32000* *is categorized as “CAT 3”; a climb >= 8000 and less than 16000 is categorized as “CAT 4”; a climb < 8000 is “Not Categorized”*

You are to write the methods below. The method headers and comments are provided.

1. **(2 points) Given distance in miles, determine the distance in meters**
   // parameters: `miles` is the length of the climb in miles
   // return the distance in meters (1 mile = 1609.334 meters)
   ```
   public static double milesToMeters(double miles)
   ```

2. **(6 points) Given the % grade of climb and the distance (in miles), determine the total points of the climb. You must call method `milesToMeters` in your solution.**
   // parameters: percent **is the %grade of the climb; `distanceInMiles` in the length of climb in miles**
   // return 0 if distance < 500 meters or %grade < 3%; Otherwise return points of climb calculated as
   // distance in meters * %grade.
   ```
   public static double getPoints(double percent, double distanceInMiles)
   ```

3. **(7 points) Given the points of the climb, determine the category of the climb ("HC", "CAT 1", "CAT 2", "CAT 3", "CAT 4") or return “Not Categorized” if the climb does not meet the conditions of a categorized climb.**
   Recall that the following declarations are made:
   ```
   double[] catLimits = {80000, 64000, 32000, 16000, 8000};
   String[] categories = {"HC", "CAT 1", "CAT 2", "CAT 3", "CAT 4"};
   ```

   // parameters: `points` is the total calculated points of the climb; the array `catLimits` will be passed
   // to the parameter `limits`; the array `categories` will be passed to the parameter `cats`.
   // return the category associated with the points. You must use all parameters in your solution.
   // Example calls:
   // getCategory(14953.93, catLimits, categories) returns “CAT 4”
   // getCategory(3379.60, catLimits, categories) returns “Not Categorized”

   ```
   public static String getCategory(double points, double[] limits, String[] cats)
   ```

Problem 2: Recursion (15 points)

Suppose we have a 1-D array of Strings:

- Each string records individual scores for four quizzes for one student,
- Each string has the form "d1;d2;d3;d4;" where d1 to d4 are all double values and they are separated using the character ';'. Note that there is one ';' after the last double value.
- Each string contains the same number of double values.

Write a recursive method that takes four inputs: a 1-D array of Strings (i.e., exams), the length of this 1-D array (i.e., n), an index to access array elements (i.e., i) and a 2-D array containing double values (i.e., ret). Note that the 2-D array is n by 4, where each row is used to record scores of one student and each column is to contain scores of one exam for all students. Some of the cells in the 2-D array may be empty. The return value of the method is a two 2-D array. The properties of the return value are the same as the parameter 2-D array.

For example, if we call the method with the parameters {"1;2;3;4;", "5;6;7;8;", "9;10;11;12;"}, 3, 0, and {{ , , ,}, { , , , }, { , , , }}, then the final output would be a 3*4 2-D array which has the value {{1.0, 2.0, 3.0, 4.0}, {5.0, 6.0, 7.0, 8.0}, {9.0, 10.0, 11.0, 12.0}}. Use the provided String library methods to operate these string values. Here is the documentation for String: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html

The method header is given below:

```
public static double[][] convert(String[] exams, int n, int i, double[][] ret)
```

and the sample client code is as below:

```
public static void main(String[] args) {
        String[] exams = {"1;2;3;4;", "5;6;7;8;", "9;10;11;12;"};
        double[][] examArray = new double[exams.length][4];
        examArray = convert(exams, exams.length, 0, examArray);
  }
```

Problem 3: Object Oriented Programming (25 points)

1. Write a Java method, `hasAz`, that has one `String` parameter, `s`. If `s` contains a lowercase 'z', then `hasAz` will return a `String` that is the reverse of `s`. Otherwise, `hasAz` will return the original `String`, `s`.

   Examples:

   `hasAz("one")` returns "one".

   `hasAz("zero")` returns "orez".

2. Consider the following class:

```
public class GiftBox{
     private double length;
     private double width;
     private double height;

     public GiftBox(double length, double width, double height){

          //code goes here
     }

     public GiftBox getNewBox(GiftBox box, double m){

          //code goes here
     }

     public boolean fitsInside(GiftBox box){
          //code goes here
     }
}
```

   a) Write the code for the `GiftBox` constructor using the header given. The constructor will initialize the private instance variables.

   b) Write the code for the method `getNewBox`. The method `getNewBox` will return a `GiftBox` that has dimensions that are `m` times the dimensions of its `GiftBox` parameter, where `m` is a double parameter of the method.
   For example, given the following code segment:
   `GiftBox gift = new GiftBox(3.0, 4.0, 5.0);`
   The call
   `getNewBox(gift, 0.5)`
   would return a `GiftBox` whose dimensions are: length = 1.5, width = 2.0, and height = 2.5.

c) Write the code for method `fitsInside`. The method `fitsInside` will return **true** if all three of the dimensions of its parameter `box` are smaller than the corresponding parameters of **this** box.
For example, if given the following code segment:

```
GiftBox gift = new GiftBox(3.0, 4.0, 5.0);
GiftBox other = new GiftBox(2.1, 3.2, 4.3);
GiftBox yetAnother = new GiftBox(2.0, 5.0, 4.0);
```

Then:

- `gift.fitsInside(other)` would return **true** because 2.1 < 3.0 and 3.2 < 4.0 and 4.3 < 5.0.
- `gift.fitsInside(yetAnother)` would return **false** because 5.0 > 4.0.

3. Consider the following class.

```
public class SomeClass{

public void methodA(int one)
     {/*code not shown*/}

public void methodA(int one, int two)
     {/*code not shown*/}

public void methodA(int one, String two)
     {/*code not shown*/}
}
```

Which of the following methods can be added to `SomeClass` without causing a compile-time error?

```
I.   public void methodA(int n)
     {/*code not shown*/}

II.  public void methodA(String one, int two)
     {/*code not shown*/}

III. public void methodA(int one, String two, int three)
     {/*code not shown*/}
```

(a) I only
(b) I and II only
(c) II and III only
(d) I, II, and III

Problem 4: Searching and Sorting (20 points)

Part 1 (8 points) Given a sorted array of n items, where n > 100.

a) (3 points) Which algorithm would you use if you are searching for an item that is around the middle of the array? The item you are searching is not the middle item. Binary search or sequential search.

b) (5 points) Why?

Part 2 (6 points) Given a sorted array of n items. When a new item is inserted, it is first added as the last item of the array and then insertion sort is used to place the item in the array so that the resulting array is in order.

<u>Example:</u>
D F I K L M N Q S ← initial array

D F I K L M N Q S G ← G is added as the last item of the array

D F G I K L M N Q S ← array after inserting G

a) (3 points) Describe the best case scenario that would result in the minimum number of comparisons to add the new item and maintain the sorted order?
b) (3 points) Describe the worst case scenario that would result in the maximum number of comparisons to add the new item and maintain the sorted order?

Part 3 (8 points) Suppose that you are to sort the vehicles array in the Vehicles Scenario Analysis assignment. What algorithm would you choose if you need to sort the vehicles by name and:

a) (4 points) You know that the array is mostly sorted?
b) (4 points) You don't know anything about the order of the items in the array?

Problem 5: Complexity Analysis (25 points)

Part 1 (10 points) Classifying Algorithms

1. Suppose that you are asked to analyze the following code fragments for their performance. Algorithms operate on 1D array of size n or 2D array of size n x n. Assume that all variables are declared, and the provided code is not a complete program. For each of the algorithms below, write down the Big O or order of growth (worst case scenario) in terms of n. If the algorithm cannot be applied to the array, write NA. Select answers from O(1), O(log n), O(n), O(n log n), O(n^2), O(n^3), O(n!). The first answer is given. Full credit will only be given for reasonable Big O answers.

code	Big O (order of growth) or NA
(1 pt) count++	O(1)
(1 pt) for (int i=1; i < n; i *=2) count++;	

(1 pt) for (int i=0; i < n; i += 2)
        if (a[i] == 0)
            count++;	

(2 pts) for (int i=0; i < n; i++)
        for (int j=i+1; j < n; j++)
            if (a[i] + a[j] == 0)
            count++;	

(2 pts) for (int i=1; i < n; i++)
        for (int j=i+1; j < n; j *=2)
        count++;	

// aux is an array of size n
int i = 0, hi=n; j = mid;
for (int k = 0; k < hi; k++) {
    if (i == mid)
        aux[k] = a[j++];
    else if (j == hi)
        aux[k] = a[i++];
    else if (a[j].**compareTo**(a[i]) < 0)
        aux[k] = a[j++];
    else
        aux[k] = a[i++];
}	

(2 pts) for (int i=0; i < n;) {
        int j=i+1;
            while (j<n && a[i] == a[j])
                j++;
            i=j;
    }	

Part 2 (15 points) Counting Operations

2. (15 points) Consider the following Java code fragment.

    1. int[ ] a = new int[n];

```
2.  a[0] = 1;
3.  int i = 1;
4.  while (i < n) {
5.      int sum = 0;
6.      for (int j = 0; j < i ; j++)
7.          sum = sum + a[j];
8.      a[i] = 1 + (2 * sum) / i;
9.      i++;
10. }
```

(a) For n > 4, fill in the values of sum and a[i] just after each of the first 4 iterations. Also write down how many times the inner j loop run for each i.

i	sum	a[i]	j loop runs
1	1	3	1
2			
3			
4			

(b) For an array of size n, how many times the code in line 7 is executed? The answer should be given in terms of n (show work to receive full credit).

(c) Using the answer in part (b), what is the big O performance measure for the algorithm above? Show your work.