Question:

You have the following class.

```
class Node {
  int data;
  Node next;

  Node(){
  }
  Node(int i){
    data=i;
  }
}
```

Write a **recursive** method that will return the count of odd numbers in the data variables of the Node objects in a linked list.
For example, if the linked list contains the following numbers as the data variables of the Node objects, 30, 55, 5, and 20, then the method should return 2 because there are two odd numbers (55 and 5) in the linked list.
You must use the following header.

```
static int countOdds(Node head){
```

Question:

You have the following class that can be used as the node of a linked list.

```
class StrNode {
 String data;
 StrNode next;

 StrNode(){
 }

 StrNode(String s){
  data=s;
 }
}
```

Write a **recursive method** that will return the number of Strings of even length in the `data`variables of a linked list of `StrNode` objects. For example, if the linked list contains the following strings

```
abc
xyzx
axcsd
klmxcv
t123
```

then the method should return 3 because three strings (`xyzx`, `klmxcv`, and `t123`) have even lengths in this linked list.

You must use the following header. Your method must not contain any syntax for a loop. The method must be recursive.

```
  static int countEvenLengthStrings(StrNode head){
```

Question:
The letters **`y`** and **`w`** are known as semivowels. Write a **recursive method** that will accept a string as the parameter and return another string containing all the semivowels of the parameter string but in **reverse order** of their appearance in the paramter.
For example, if the parameter contains the following string
`SnoWyDaYFewDew`

the method should return the following string
`wwYyW`
Note that **`y`**, **`Y`**, **`w`**, and **`W`** are considered semivowels in this problem.
The method must be recursive and cannot contain any syntax for loops.
You must use the following header.

```
  static String getRevSemiVowels(String given){
```

Question:

Two algorithms, A and B, have the time functions f(n) and g(n) respectively, where
f(n)=10n^2 +10n+15
g(n)=9n^2+14n+18
Determine which algorithm runs faster.

You must show the derivations for full credit. You will need to compute a maximum common point after which one algorithm is faster than the other, but the conclusion might be different before that point.
After writing all the derivations and calculations, clearly state which algorithm will run faster by writing either "A runs faster" or "B runs faster".
Note: n^2 is written as n^2 . The character ^ can be typed by pressing the number 6 on the keyboard while keeping the SHIFT key pressed. That is, SHIFT+6 will type the character ^.

Solution:

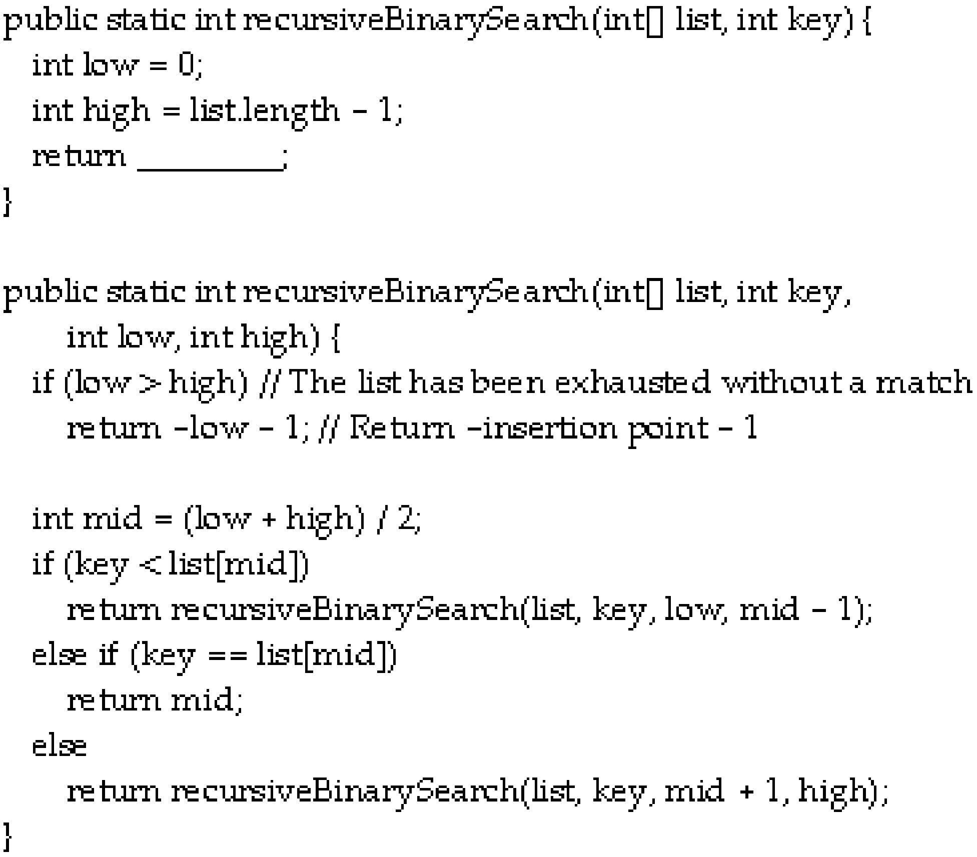
Let us find the roots.
Consider, $f(n)=g(n)$
$10n^2 +10n+15= 9n^2+14n+18$
$n^2-4n+3=0$
$n^2-n-3n+3=0$
$n(n-1)-3(n-1)=0$
$(n-1)(n-3)=0$
$n=1, n=3$

The largest root where both the equations are equal is 3.
$4>3$.
$f(4)=10(4)^2 +10*4+15=160+40+15=215$
$g(4)= 9(4)^2+14*4+18=144+56+18=218$

$f(4)<g(4)$

Therefore algorithm A is the faster algorithm.

Fill in the code to complete the following method for binary search.



```
public static int recursiveBinarySearch(int[] list, int key) {
  int low = 0;
  int high = list.length - 1;
  return _________;
}

public static int recursiveBinarySearch(int[] list, int key,
     int low, int high) {
  if (low > high) // The list has been exhausted without a match
     return -low - 1; // Return -insertion point - 1

  int mid = (low + high) / 2;
  if (key < list[mid])
     return recursiveBinarySearch(list, key, low, mid - 1);
  else if (key == list[mid])
     return mid;
  else
     return recursiveBinarySearch(list, key, mid  + 1, high);
}
```

recursiveBinarySearch(list, key)
recursiveBinarySearch(list, key, low + 1, high - 1)
recursiveBinarySearch(list, key, low - 1, high + 1)
✓recursiveBinarySearch(list, key, low, high)

For a sorted list of 2048 elements, a binary search takes at most ________ comparisons. Select the closest possible answer.

✓12
100
512
6
8
1024

2base log of 2048 is 11. Therefore 12 is the answer as it is the closest one. The added one in binary search may come from an array of size zero in the base case.

A list is sorted by selecting the smallest element in the list and swapping it with the first one in the array. This technique is called ________.

insertion sort

✓selection sort

bubble sort

merge sort

quick sort

Question

What is the big-Oh notation for the following piece of code, in terms of the variable n.

```
for (int i=0; i<n*n; i++){
  for (int k=0; k<n; k++){
    for (int j=0; j<n; j++){
      s=s+i*j*k;
    }
  }
}
```

Answer

$O(n)$

$O(n^2)$

✓$O(n^4)$

$O(\log n)$

$O(n \log n)$

$O(n^2 \log n)$

$O(2^n)$

$O(\frac{1}{n})$

Infinite loop

$O(1)$

$O(n^3)$

What is the big-Oh notation for the following piece of code, in terms of the variable n.

```
 static void printer(int n){
  if (n<=0)
   return;

  for (int i=0; i<n; i++){
   System.out.print("*");
  }
  System.out.println();

  printer(n-1);

 }
```

Answer:

$O(n)$

[✓ $O(n^2)$]

$O(n^4)$

$O(\log n)$

$O(n \log n)$

$O(n^2 \log n)$

$O(2^n)$

$O(\frac{1}{n})$

Infinite loop

$O(1)$

$O(n^3)$

Question

What is the big-Oh notation for the following time function?
T(n)=T(n+3)+c

Answer

$O(n)$

$O(n^2)$

$O(n^4)$

$O(\log n)$

$O(n \log n)$

$O(n^2 \log n)$

$O(2^n)$

$O(\frac{1}{n})$

✓ Infinite loop

$O(1)$

$O(n^3)$

Incorrect Feedback

The equation reads like this "The time to solve a problem of size n is equal to the time to solve a problem of size n+3 plus some constant. " This is a never ending loop.

Question

Suppose a list is {2, 9, 5, 4, 8, 1}. After the **third** pass of bubble sort, the list becomes ________.

Answer

2, 9, 5, 4, 8, 1

2, 9, 5, 4, 1, 8

2, 5, 9, 4, 8, 1

2, 5, 4, 8, 1, 9

2, 1, 5, 4, 8, 9

{2, 5, 4, 1, 8, 9}

✓{2, 4, 1, 5, 8, 9}

{2, 9, 4, 5, 8, 1}

Feedback:
After the first pass:
{2, 5, 4, 8, 1, 9}

After the second pass:
{2, 5, 4, 1, 8, 9}

After the third pass:
{2, 4, 1, 5, 8, 9}

Question

Which of the following statements is/are **CORRECT**?

Answer

$O(\log n) < O(n^2) < O(n \log n) < O(2^n)$

✓$O(1) < O(n \log n) < O(n^2 \log n) < O(2^n)$

✓$O(n(\log n)^2) < O(n^2 \log n) < O(n^3) < O(3^n)$

$O(n^3) < O((\log n)^3) < O(n^2 \log n) < O(n(\log n)^2)$

$O(n^2 \log n) < O((\log n)^3) < O(n^3 \log n) < O(n(\log n)^3)$

Question:

The time complexity for the selection sort algorithm in the text is ________.

O(nlogn)

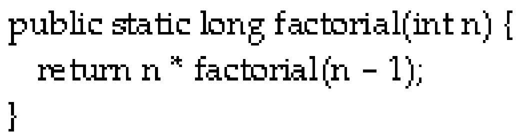
✓O(n^2)

O(logn)

O(2^n)

Question:

Analyze the following recursive method and select the most appropriate statement.



```
public static long factorial(int n) {
  return n * factorial(n - 1);
}
```

Invoking factorial(0) returns 0.

Invoking factorial(1) returns 1.

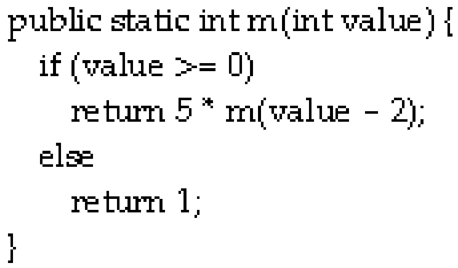
Invoking factorial(2) returns 2.

Invoking factorial(3) returns 6.

✓The method runs infinitely and causes a StackOverflowError.

Question:

Consider the following recursive method.



```
public static int m(int value) {
   if (value >= 0)
      return 5 * m(value - 2);
   else
      return 1;
}
```

What value is returned when invoking m(5)?

15

25

75

100

✓125

225

Feedback:

m(5)
=5*m(3)
=5*5*m(1)
=5*5*5*m(-1)
=5*5*5*1
=125