Unit 10/Recursion Practice

Name: (Key) 3/10/22

1. What is returned by the call mystery(4)?

```
public static int mystery(int a)
 {
   if(a>11)
     return 5;
   return a + mystery(a+3);
 }
```

4 + 7 + 10 + 5 = 26

2. What is returned by the call mystery(6)?

```
public static int mystery(int a)
 {
   if(a==3)
     return 1;
   return a * mystery(a-1);
 }
```

6 · 5 · 4 · 1 = 120

3. What is printed by the call mystery(4)?

```
public static void mystery(int a)
 {
   System.out.print(a);
   if(a<7)
     mystery(a+2);
 }
```

4 6 8

4. What is returned by the call mystery(4)?

```
public static void mystery(int a)
 {
   if(a<7)
     mystery(a+2);
   System.out.print(a);
 }
```

4 6 8

864

5. Write a recursive method **evenFactorial** which returns the even factorial of a number (that is, the product of all positive integers less than or equal to n). The factorial of 0 is 1.

```
public static int evenFactorial(int n)
{
  if(a==0) return 1;
  if(a%2==0)
     return a*mystery(a-1);
  return mystery(a-1);

}
```

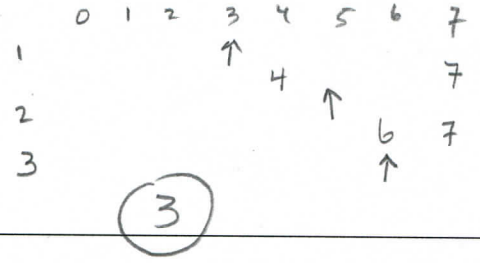
6. Write a recursive method **spaceIt** to print the digits of an integer on a single line with 2 spaces between each digit. So spaceIt(123) prints "1 2 3".

```
public static void spaceIt(int m)
{
  if(a>10)
     mystery(a/10);
  system.out.print(a%10+"__");


}
```

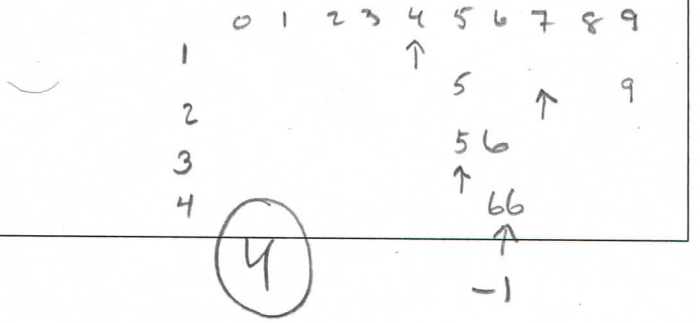
7. Given the following array how many times would a recursive binary search method be called when searching for the value 21?

```
int[] array = {3, 6, 8, 11, 14, 16, 21, 22};
```



8. Given the following array how many times would a recursive binary search method be called when searching for the value 6?

```
int[] array = {0, 1, 2, 3, 4, 5, 7, 8, 9, 9};
```



8. What are two features that must be found in every recursive method?

① a base case

② a call to itself

9. Write a non-recursive binary search method that accepts an int array **nums** and an int **target** and returns the index where target is located or -1 if target is not present in the array. Preconditions: int[] nums contains valid data that is sorted and contains at least one element.

```
public static int binarySearch(int[] nums, int target)
  {
   int low = 0;
   int high = nums.length - 1;
   while (low <= high)
     {
      int mid = (low + high)/2;
      if (nums[mid] == target) return mid;
      else if (nums[mid] < target)
          {
            low = mid + 1;
          }
      else
          {
            high = mid - 1;
          }
     }
   return -1;
}
```