

## 1. distinctNum

`==` 改为 `!=`

## 2. sumDistinct

初始化sum数组写在sort之后

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The function `sumDistinct` accepts two arguments:  
`size` - an integer representing the size of the array  
`inputArray` - a list of integers representing an array (assume this will not be null or empty)

It is supposed to return an integer representing the sum of the distinct elements in the given array.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of function/method `sumDistinct`. Your task is to fix the code so that it passes all the test cases

```
1 import java.util.Arrays;
2 class Distinct
3 {
4     int sumDistinct (int size, int[] inputArray)
5     {
6         int sum = inputArray[0];
7         Arrays.sort(inputArray);
8         int point = inputArray[0];
9         for(int i = 1 ; i < size ; i++)
10        {
11            if(point != inputArray[i])
12            {
13                sum+=inputArray[i];
14                point = inputArray[i];
15            }
16        }
17    }
18    return sum;
19 }
20 }
```

## 3. eliminateVowel

`default` 去掉 `i++`

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The function/method `eliminateVowel` accepts a string as an argument and returns a string after eliminating all vowels, {*A, E, I, O, U, a, e, i, o and u*}, from the input string.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of the function `eliminateVowel`. Your task is to fix the code so that it passes all the test cases

```
1 class Solution
2 {
3     String eliminateVowel(String str)
4     {
5         String newString = "";
6         int i=0;
7         char[] S = str.toCharArray();
8         int len = str.length();
9         if(len == 0)
10         {
11             return "";
12         }
13         while(i < S.length)
14         {
15             switch(S[i])
16             {
17                 default:
18                     newString +=S[i];
19                     i++;
20                 case 'a': i++;
21                     break;
22                 case 'e': i++;
23                     break;
24                 case 'i': i++;
25                     break;
26                 case 'o': i++;
27                     break;
28                 case 'u': i++;
29                     break;
30                 case 'A': i++;
31                     break;
32                 case 'E': i++;
33                     break;
34                 case 'I': i++;
35                     break;
36                 case 'O': i++;
37                     break;
38                 case 'U': i++;
39                     break;
40             }
41         }
42         return newString;
43     }
44 }
```

## 4. checkPalindrome

把最后返回值改成 `return (result == temp);`

`result = result * 10 + remainder` ????

are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *System.out.println* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

A palindromic number is a number that remains the same if its digits are reversed, for example, 13631.

The function/method *checkPalindrome* accepts an integer *num* and returns 'true' if the given number is a palindromic number. Otherwise, it returns 'false'.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of the function *checkPalindrome*. Your task is to fix the code so that it passes all the test cases.

```
1 class Solution
2 {
3     boolean checkPalindrome (int num)
4     {
5         int result=0;
6         int remainder;
7         int temp = num;
8         while(num != 0)
9         {
10            remainder = num % 10;
11            result = remainder + result * 10;
12            num=num/10;
13        }
14    }
15 }
16 }
```

SUBMIT

## 5. distinctElementCount

if(flag == 1) 改成 if(flag == 0)

Problem | Test Cases | Output | Compile and Run

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *System.out.println* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method *distinctElementsCount* returns the count of the unique elements in an array.

It accepts two arguments - *size*, representing the number of elements in the array, and *elements*, an array of integers representing the elements.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of the function *distinctElementsCount*. Your task is to fix the code so that it passes all the test cases.

```
1 class Solution
2 {
3     int distinctElementsCount (int size, int[] elements)
4     {
5         int[] counted = new int[size];
6         int count, flag;
7         counted[0] = elements[0];
8         count = 1; //one element is counted
9         for(int i=0; i < size; i++)
10        {
11            flag = 0;
12            for(int j=0; j < count; j++)
13            {
14                if(elements[i] == counted[j])
15                {
16                    flag = 1;
17                }
18            }
19            if(flag == 1)
20            {
21                count++;
22                counted[count-1] = elements[i];
23            }
24        }
25    }
26 }
27 }
```

SUBMIT

下面是改后

13m  
02s

**QUESTION**  
**1 out of 7**

**Problem** | **Test Cases** | **Output** | **Compile and Run**

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method `distinctElementsCount` returns the count of the unique elements in an array.

It accepts two arguments - `size`, representing the number of elements in the array; and `elements`, an array of integers representing the elements.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of the function `distinctElementsCount`. Your task is to fix the code so that it passes all the test cases.

**HELP**

```

1 class Solution
2 {
3     int distinctElementsCount (int size, int[] elements)
4     {
5         int[] counted = new int[size];
6         int count, flag;
7         counted[0] = elements[0];
8         count = 1; //one element is counted
9         for(int i=0; i < size; i++)
10        {
11             flag = 0;
12             for(int j=0; j < count; j++)
13            {
14                if(elements[i] == counted[j])
15                {
16                    flag = 1;
17                }
18            }
19            if(flag == 0)
20            {
21                count++;
22                counted[count-1] = elements[i];
23            }
24        }
25    }
26    return count;
27 }
```

**SUBMIT ANSWER**

## 6. matrixSum

```
while(j < n)
    sum += matrix[i++][j++]
```

改成

```
while(j < n) { sum += matrix[i][j++]} i++;
```

或者 `i++` 和 `j++` 换位置？

**Problem** | **Test Cases** | **Output** | **Compile & Run**

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The method `matrixSum(int matrix[][], int m, int n)` of class `Matrix` is supposed to return the sum of elements of the input array `matrix`.

The method compiles successfully but fails to return the desired result due to logical errors.

Your task is to fix the program so that it passes all test cases.

```

1 // You can print the values to stdout for debugging
2 public class Matrix {
3     public int matrixSum(int[][] matrix) {
4         int m = matrix.length;
5         int n = matrix[0].length;
6         int i = 0, j, sum = 0;
7         while (i < m) {
8             j = 0;
9             while (j < n) {
10                 sum += matrix[i++][j++];
11             }
12         }
13     }
14 }
```

## 7. labelProduct

加入入字符串串的方方向反了了 while 循环中

**Problem** | Test Cases | Output | **Compile and Run**

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *System.out.println* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

A company has decided to label the products using its product ID. Each product has an alphanumeric Product ID, and a Product Label that is the reverse of the product ID after removing any numeric characters.

The function *labelProduct* accepts an argument - *productID*: a non-null, lower-case string representing the product ID

The function returns a string representing the product label.

The code compiles successfully but fails to return the desired result for some test cases due to an incorrect implementation of function *labelProduct*. Your task is to fix the code so that it passes all the test cases.

```

1 class Product
2 {
3     public String labelProduct(String productID)
4     {
5         String str = "";
6         int i=0;
7         productID = productID.toLowerCase();
8         while(i < productID.length())
9         {
10             if((productID.charAt(i) >= 'a') && (productID.charAt(i) <= 'z')
11             {
12                 str+= productID.charAt(i);
13             }
14             i++;
15         }
16         return str;
17     }
18 }
19 
```

**QUESTION: 3**

**SUBMIT ANSWER**

改之后

**Problem** | Test Cases | Output | **Compile and Run**

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *System.out.println* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

A company has decided to label the products using its product ID. Each product has an alphanumeric Product ID, and a Product Label that is the reverse of the product ID after removing any numeric characters.

The function *labelProduct* accepts an argument - *productID*: a non-null, lower-case string representing the product ID

The function returns a string representing the product label.

The code compiles successfully but fails to return the desired result for some test cases due to an incorrect implementation of function *labelProduct*. Your task is to fix the code so that it passes all the test cases.

```

1 class Product
2 {
3     public String labelProduct(String productID)
4     {
5         String str = "";
6         int i = productID.length() - 1;
7         productID = productID.toLowerCase();
8         while(i >= 0)
9         {
10             if((productID.charAt(i) >= 'a') && (productID.charAt(i) <= 'z')
11             {
12                 str+= productID.charAt(i);
13             }
14             i--;
15         }
16         return str;
17     }
18 }
19 
```

**SUBMIT ANSWER**

## 8. medianValue

arr2 越界，改为 arr2[i-size]

else arr[i] = arr2[i] → else arr[i] = arr2[i - size]

**Problem** | Test Cases | Output | **Compile and Run**

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods. Given a sorted list of numbers, the median value of a list having even number of elements is determined by adding together the middle pair and dividing the sum by two.

The function `medianValue` accept three arguments:  
`size` - an integer representing the size of the two given arrays  
`arr1` - a list of integers representing the first array (assume this will not be null or empty)  
`arr2` - a list of integers representing the second array (assume this will not be null or empty)

It is supposed to return the median of the even sized array obtained after merging the two arrays and sorting its elements.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of function/method `medianValue`. Your task is to fix the code so that it passes all the test cases.

```

1 import java.util.Arrays;
2 class Median
3 {
4     int medianValue(int size, int[] arr1, int[] arr2)
5     {
6         int[] arr = new int[2 * size];
7
8         for(int i = 0; i < 2 * size; i++)
9         {
10             if(i < size)
11                 arr[i] = arr1[i];
12             else
13                 arr[i] = arr2[i];
14         }
15         Arrays.sort(arr);
16
17         int length = 2*size;
18         int medi = (arr[length / 2 - 1]+arr[length/2]);
19         return medi;
20     }
21
22 }
23

```

改之后

**Problem** | Test Cases | Output | **Compile and Run**

**5 out of 7**

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

Given a sorted list of numbers, the median value of a list having even number of elements is determined by adding together the middle pair and dividing the sum by two.

The function `medianValue` accept three arguments:  
`size` - an integer representing the size of the two given arrays  
`arr1` - a list of integers representing the first array (assume this will not be null or empty)  
`arr2` - a list of integers representing the second array (assume this will not be null or empty)

It is supposed to return the median of the even sized array obtained after merging the two arrays and sorting its elements.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of function/method `medianValue`. Your task is to fix the code so that it passes all the test cases.

```

1 import java.util.Arrays;
2 class Median
3 {
4     int medianValue(int size, int[] arr1, int[] arr2)
5     {
6         int[] arr = new int[2 * size];
7
8         for(int i = 0; i < 2 * size; i++)
9         {
10             if(i < size)
11                 arr[i] = arr1[i];
12             else
13                 arr[i] = arr2[i - size];
14         }
15         Arrays.sort(arr);
16
17         int length = 2*size;
18         int medi = (arr[length / 2 - 1]+arr[length/2])/2;
19         return medi;
20     }
21
22 }
23

```

**SUBMIT ANSWER**

## 9. countDays

`(year % 4 == 0) && (year %100 != 0) || (year % 400 == 0)`

`((y % 4) == 0 || (y % 100) != 0)` 应改为 `((y % 4) == 0 && (y % 100) != 0)`

`((year % 400 == 0 && year % 3200 != 0) || (year % 100 != 0)) && (year % 4 == 0))`

Problem | Test Cases | Output |

Compile and Run

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

A method `countDays(int month, int year)` of class `DaysCount` is written such that it returns the number of days in the `month` of the given `year`. The method compiles fine but fails to return the desired result in some test cases. Your task is to modify the code to give correct result for all the cases.

**Note**  
A year that is evenly divisible by 100 is a leap year only if it is also evenly divisible by 400. For example, 2000 is a leap year (because it's divisible by 400), but 1900 is not (because it's divisible by 100, but not by 400).

```

1 public class DaysCount
2 {
3     public int countDays(int month, int year)
4     {
5         switch (month)
6         {
7             case 1:
8             case 3:
9             case 5:
10            case 7:
11            case 8:
12            case 10:
13            case 12:
14                return 31;
15            case 4:
16            case 6:
17            case 9:
18            case 11:
19                return 30;
20            case 2:
21                if (((year % 4 == 0) ||
22                    ((year % 100 != 0) ||
23                     ((year % 400 != 0))
24                ) || (year % 400 != 0))
25                return 29;
26            else
27                return 28;
28        default:
29            // Should not happen.
30            return -1;
31        }
32    }
}

```

## 10. countElementRange

把 `||` 改成 `&&`

Problem | Test Cases | Output |

Compile and Run

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The function/method `countElementRange` accepts four arguments:

- `size` - an integer representing the size of the array
- `inputArray` - a list of integers representing an array (assume this will not be null or empty)
- `low` - an integer representing the lower bound of the range (inclusive) This will be smaller than "high"
- `high` - an integer representing the upper bound of the range (inclusive) This will be larger than "low"

It is supposed to return an integer representing the number of elements in the given range.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of function/method `countElementRange`. Your task is to fix the code so that it passes all the test cases.

```

1 class Range
2 {
3     int countElementRange (int size, int[] inputArray,
4     {
5         int count = 0;
6         for(int i=0;i<size;i++)
7         {
8             if(inputArray[i] >= low && inputArray[i] <=
9                 count++;
10        }
11    }
12 }
13
14

```

## 11. countNumParity getDigitSumParity DigitSum

(1) 主函数 `result % 2 == 0`

(2) 主函数 `if (result == 0) {return 1}`

(3) `sumDigit` 函数把 `int temp = num % 10` 和 `num = num/10` 换一下位置

Problem | Test Cases | Output |

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *System.out.println* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The method ***getDigitSumParity(int arr[])*** of class ***DigitSum*** accepts an integer array *arr*. It is supposed to calculate the sum of digits of the smallest element in the input array. It returns 1 if the calculated sum is even and returns 0 otherwise. The method compiles successfully but fails to return the desired result due to logical errors.

Your task is to fix it so that the program works for all input values. Note: The method ***getDigitSumParity*** uses another method ***getSum(int num)*** which returns the sum of digits of the input number *num*.

```
1 // You can print the values to stdout for debugging
2 public class DigitSum {
3     public int getDigitSumParity(int[] arr) {
4         int min = getMin(arr);
5         int result = getSum(min);
6         if (result == 0) {
7             return 0;
8         }
9         if (result % 2 != 0) {
10            return 1;
11        } else {
12            return 0;
13        }
14    }
15
16    public int getMin(int[] arr) {
17        if (arr == null || arr.length <= 0) {
18            throw new IllegalArgumentException(
19                "Input array should contain at least 1");
20        }
21        int min = arr[0];
22        for (int i = 1, len = arr.length; i < len; i++) {
23            if (arr[i] < min) {
24                min = arr[i];
25            }
26        }
27        return min;
28    }
29
30    public int getSum(int num) {
31        int sum = 0;
32        while (num != 0) {
33            num = num / 10;
34            int temp = num % 10;
35            sum = sum + temp;
36        }
37        return sum;
38    }
39 }
```

## 12. countProduct

for (*j* = 0; *j* < value; *j*++) 改为 for (*j* = 0; *j* < size; *j*++);

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *System.out.println* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method ***countProduct*** returns a count of the number of store products with prices that are strictly less than a minimum-decided price K. The function/method ***countProduct*** accepts three arguments - *size*, an integer representing the number of products in the store; *valueofK*, an integer representing the minimum decided price K and *priceList*, a list of integers representing the prices of the products.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of the function ***countProduct***. Your task is to fix the code so that it passes all the test cases.

```
1 class Solution {
2 {
3     int countProduct(int size, int valueofK, int[] priceList)
4     {
5         if(size == 0)
6             return 0;
7         int j, count = 0;
8         for(j=0; j<valueofK; j++)
9         {
10             if(priceList[j]<valueofK)
11                 count = count + 1;
12         }
13     }
14 }
15 }
```

改之后

**Problem** | **Test Cases** | **Output** | **Compile and Run** |     

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The function/method `countProduct` returns a count of the number of store products with prices that are strictly less than a minimum-decided price K. The function/method `countProduct` accepts three arguments - `size`, an integer representing the number of products in the store; `valueofK`, an integer representing the minimum decided price K and `priceList`, a list of integers representing the prices of the products.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of the function `countProduct`. Your task is to fix the code so that it passes all the test cases.

```

1 class Solution
2 {
3     int countProduct(int size, int valueofK, int[] priceList)
4     {
5         if(size == 0)
6             return 0;
7         int j, count = 0;
8         for(j=0; j< size; j++)
9         {
10            if(priceList[j]<valueofK)
11                count = count + 1;
12        }
13    }
14 }
15 
```

**QUESTION**

**SUBMIT ANSWER**

## 13. countElement

改成 `i < len; i++`

后面面 `arr[i+1]` 改成 `arr[i]`

**Problem** | **Test Cases** | **Output** | **Compile and Run** |    

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The method `countElement(int arr[], int n)` of class `ElementCount` is supposed to return the number of elements in the input array `arr` which are greater than twice the input number `n`. The method compiles successfully but fails to return the desired result due to logical errors. Your task is to fix the program so that it passes all test cases.

```

1 // You can print the values to stdout for debugging
2 public class ElementCount {
3     public int countElement(int arr[], int n) {
4         int count = 0, len = arr.length;
5         int doubleN = 2 * n;
6         for (int i = 0; i < n; ) {
7             if (arr[i + 1] > doubleN) {
8                 count += 1;
9             }
10        }
11    }
12 }
13 
```

**SUBMIT TEST** | **SUBMIT**

下面是改后

**Problem** | **Test Cases** | **Output** | **Code saved** | **Compile and Run**

7 out of 7

You are required to fix all logical errors in the given code. You can click on **Compile & Run** anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The method `countElement(int arr[], int n)` of class `ElementCount` is supposed to return the number of elements in the input array `arr` which are greater than twice the input number `n`.

The method compiles successfully but fails to return the desired result due to logical errors.

Your task is to fix the program so that it passes all test cases.

```

1 // You can print the values to stdout for debugging
2 public class ElementCount {
3     public int countElement(int arr[], int n) {
4         int count = 0, len = arr.length;
5         int doubleN = 2 * n;
6         for (int i = 0; i < len; i++) {
7             if (arr[i] > doubleN) {
8                 count += 1;
9             }
10        }
11    }
12 }
13 
```

## 14. checkArmstrong

`result+=math.pow(result, digitCount)`应该改为  
`result+=math.pow(remainder, digitCount)`

**Problem** | **Test Cases** | **Output** | **Code saved** | **Compile and Run**

You are required to fix all logical errors in the given code. You can click on **Compile & Run** anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

Armstrong number is an  $n$ -digit number where the sum of  $n$ -th power of its each digits is equal to the same  $n$ -digit number. Thus an Armstrong number of three digit is that number in which the sum of the cubes of its digits is equal to the number itself. ... For example, 371 is an Armstrong number since  $3^3 + 7^3 + 1^3 = 371$ .

The function/method `checkArmstrong` accepts an integer `num` and returns '1' if the given number is an Armstrong number. Otherwise, it returns '0'.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of the function `checkArmstrong`. Your task is to fix the code so that it passes all the test cases.

```

1 class Solution
2 {
3     int checkArmstrong(int num)
4     {
5         int digitCount=0, result=0;
6         // calculate number of digits
7         int temp = num;
8         while (temp != 0)
9         {
10             temp = temp/10;
11             digitCount++;
12         }
13         // sum digits to nth power
14         temp = num;
15         while (temp != 0)
16         {
17             int remainder = temp%10;
18             result += Math.pow(result, digitCount);
19             temp /= 10;
20         }
21         if(result == num)
22             return 1;
23         else
24             return 0;
25     }
26 }
27 
```

## 15. reverseNumber

应该是 `reverseNum = reverseNum*10 + Reminder`

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *System.out.println* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method *reverseNumber* accepts an integer and returns an integer which is the reverse of the digits of the input number. For example, if the input is 376, the output will be 673.

The function/method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of the function *reverseNumber*. Your task is to fix the code so that it passes all the test cases.

```

1 class Solution
2 {
3     int reverseNumber (int num)
4     {
5         int reversedNum = 0;
6         while(num != 0)
7         {
8             int remainder = num % 10;
9             num = num / 10;
10            if(num != 0){
11                reversedNum = (reversedNum + remainder) * 10;
12            }else{
13                reversedNum = (reversedNum + remainder);
14            }
15        }
16     return reversedNum;
17 }
18 ]]
```

QUESTION 4

1  
2  
3  
4  
5  
6  
7

## 16. appearsKTimes

while loop 结束后 添加

```
if (count == k) { res = element }
```

if 和 else 内容 换一一下位置，

先判断 if(element!=inputArray[i])

**while loop 后加个check, 并且  
res = Math.max(res, element)**

**Problem | Test Cases | Output |**

Compile and Run

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *System.out.println* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method *appearsKTimes* accepts three arguments - *size*: a positive integer representing the size of the given array *inputArray*: a non-null array of integers. *k*: a positive integer representing the number of occurrences to search for.

The function/method *appearsKTimes* is supposed to return an integer representing the element that occurs *k* times in the given array. If more than one element occurs *k* times, return the largest one and if no element occurs *k* times, return '-1'.

The code compiles successfully but fails to return the desired result for some test cases due to an incorrect implementation of function/method *appearsKTimes*. Your task is to fix the code so that it passes all the test cases.

```

1 import java.util.Arrays;
2 class AppearsK
3 {
4     public int appearsKTimes(int size, int inputArray[]
5     {
6         Arrays.sort(inputArray);
7         int i=1, count = 1;
8         int element = inputArray[0];
9         int res = -1;
10        while(i < size)
11        {
12            if(element == inputArray[i])
13            {
14                count++;
15            }
16            else
17            {
18                if(count == k)
19                {
20                    res = element;
21                }
22                element = inputArray[i];
23                count = 1;
24            }
25            i++;
26        }
27    }
28    return res;
29 }
```

下面是改后

**Problem** | **Test Cases** | **Output** | **Compile and Run**

**QUESTION**  
**2 out of 7**

**aspiringminds**  
Engineering Simplified

**HELP**

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The function/method `appearsKTimes` accepts three arguments - `size`: a positive integer representing the size of the given array `inputArray`; a non-null array of integers. `k`: a positive integer representing the number of occurrences to search for.

The function/method `appearsKTimes` is supposed to return an integer representing the element that occurs `k` times in the given array. If more than one element occurs `k` times, return the largest one and if no element occurs `k` times, return `-1`.

The code compiles successfully but fails to return the desired result for some test cases due to an incorrect implementation of function/method `appearsKTimes`. Your task is to fix the code so that it passes all the test cases.

```

1 import java.util.Arrays;
2 class AppearsK
3 {
4     public int appearsKTimes(int size, int inputArray[], int k)
5     {
6         Arrays.sort(inputArray);
7         int i=1, count = 1;
8         int element = inputArray[0];
9         int res = -1;
10        while(i < size)
11        {
12            if(element == inputArray[i])
13            {
14                count++;
15            }
16            else
17            {
18                if(count == k)
19                {
20                    res = element;
21                }
22                element = inputArray[i];
23                count = 1;
24            }
25            i++;
26        }
27        if (Count ==k)
28        res = element;
29    return res;
30 }
31 }
```

**SUBMIT ANSWER**

## 17. reverseString

去掉 `len++`?

不不是 `str += ? ? ?`

## 18. replaceValues

若 array 长度为奇数全换为 1 偶数换为 0

或者

for loop 中 `i<=len; j<=len` 改成 `i<len, j<len`

**Problem** | **Test Cases** | **Output**

**Save** | **Compile & Run** | **Next Question** | **Navigate to question**

**Reset**

// You can print the values to stdout for debugging

```

1 public class ArrayOperation{
2     public static int[] replaceValues( int arr[]){
3         int i , j , len = arr.length;
4         if( len % 2 == 0 ){
5             for( i = 0 ; i <= len ; i ++ )
6                 arr[i] = 0;
7         }
8         else{
9             for( j = 0 ; j <= len ; j ++ )
10                arr[j] = 1;
11         }
12     }
13     return arr;
14 }
15 }
```

A method `replaceValues(int arr[])` of class `ArrayOperation` accepts an array `arr` as an input and returns an array of the same length.

If the length of `arr` is odd, all the elements of `arr` are replaced by 1s and in case it is even, the elements are replaced by 0s.

For example: given the input array {0,1,2}, the function will return the array {1,1,1}.

The function compiles successfully but fails to return the desired result due to logical

```
na 1  /**
2   * Created by yaqunyu on 10/20/16.
3   */
4  public class ArrayOperation {
5      @
6          public static int[] replaceValues(int arr[]) {
7              int i, j, len = arr.length;
8              if(len % 2 == 0){
9                  for(i = 0; i<len; i++) {
10                     arr[i] = 0;
11                 }
12             } else{
13                 for(j = 0; j<len; j++){
14                     arr[j] = 1;
15                 }
16             }
17         return arr;
18     }
19     public static void main(String[] args){
20         int[] myIntArray = new int[]{20,2,3,5,9,6,16};
21         for(int i = 0; i<myIntArray.length; i++) {
22             System.out.println(replaceValues(myIntArray)[i]);
23         }
24     }
25 }
26
```

## 19. countOccurrences

返回 value 在 array 中出现的次数, while 里面最后加上 i++ ( error: TLE )

The screenshot shows a programming interface with a code editor and a problem description panel.

**Problem Description:**

You are required to fix all logical errors in the given code. You can click on **Compile & Run** anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The method `countOccurrence(int arr[], int value)` of class `Occurrence` is supposed to return the count of occurrences of a number `value` in the input array `arr`. The function compiles successfully but fails to return the desired result due to logical errors.

Your task is to debug the program to pass all the test cases.

**Code Snippet:**

```
1 // You can print the values to stdout for debugging
2 public class Occurrence{
3     public static int countOccurrence( int arr[], int
4         int i = 0 , count = 0 , len = arr.length;
5         while( i < len ){
6             if( arr[i] == value )
7                 count += 1;
8             i++;
9         }
10    return count;
11 }
12 }
```

**Buttons:** Problem, Test Cases, Output, Compile & Run, Submit Answer.

## 20. checkGrade gradingsystem

成绩打分 ABCD , 判断输入的数在什么范围, 它上面是 (  $x \geq 70$  ) || (  $x < 80$  ), 两个|| 改成 &&

## 21. sortArray descendingSortArray

if ( $\max > arr[j]$ ) 改成 <

**Problem** Test Cases Output Auto Saved Compile & Run Next Question Navigate to question

Info You are required to correct the logical/syntactical mistake in the given Java code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code.

The submitted code should not just pass compilation but also should be logically correct, passing all the testcases.

Do not write the `main()` method.  
`java.util` package is allowed. Some Java packages like `java.lang.reflect` are not allowed to be used.  
An algorithm/Javadoc MAY HAVE BEEN provided to help you understand the problem. JRE 1.7 is used.

For this question, you will need to correct the given implementation. **Do Not** attempt to modify the approach or incorporate library methods.

A method `sortArray(int arr[])` of class `ArraySort` accepts an integer array `arr` as an input and performs an inplace sort operation on it. The function is expected to return the input array sorted in descending order.

The function compiles successfully but fails to return the desired result due to logical errors.

Your task is to debug the program to pass all the test cases.

```
1 // You can print the values to stdout for debugging
2 public class ArraySort{
3     public static int[] sortArray( int arr[] ){
4         int i , max , location , j , temp , len = arr.length;
5         for( i = 0 ; i < len ; i ++ ){
6             max = arr[i];
7             location = i;
8             for( j = i ; j < len ; j ++ ){
9                 if( max > arr[j] ){
10                     max = arr[j];
11                     location = j;
12                 }
13             }
14             temp = arr[i];
15             arr[i] = arr[location];
16             arr[location] = temp;
17         }
18         return arr;
19     }
20 }
```

Sort Array

if ( $arr[i] > arr[j]$ ) 改成 <

**Problem** Test Cases Output Save Compile & Run Next Question Navigate to question

Info You are required to correct the logical/syntactical mistake in the given Java code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code.

The submitted code should not just pass compilation but also should be logically correct, passing all the testcases.

Do not write the `main()` method.  
`java.util` package is allowed. Some Java packages like `java.lang.reflect` are not allowed to be used.  
An algorithm/Javadoc MAY HAVE BEEN provided to help you understand the problem. JRE 1.7 is used.

For this question, you will need to correct the given implementation. **Do Not** attempt to modify the approach or incorporate library methods.

A method `sortArray(int arr[])` of class `Array` accepts an integer array `arr` as an input and performs an inplace sort operation on it. The function is expected to return the input array sorted in descending order, but instead, it returns the array sorted in ascending order due to a bug in the code.

Your task is to debug the program to pass all the test cases.

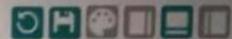
```
1 // You can print the values to stdout for debugging
2 public class Array{
3     public static int[] sortArray( int arr[] ){
4         int len = arr.length;
5         int small , pos , i , j , temp;
6         for( i = 0 ; i <= len - 1 ; i ++ ){
7             for( j = i ; j < len ; j ++ ){
8                 temp = 0;
9                 if( arr[i] > arr[j] ){
10                     temp = arr[i];
11                     arr[i] = arr[j];
12                     arr[j] = temp;
13                 }
14             }
15         }
16         return arr;
17     }
18 }
```

## 22. printPattern

第一种情况 给 for loop 加括号{}

Problem | Test Cases | Output |

Compile & Run



You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The method `printPattern(int n)` of class `DrawPattern` is expected to print the first  $n$  ( $n \geq 0$ ) lines of the pattern shown below.

For example, if  $n = 3$ , the pattern should be:

```
11  
1111  
111111
```

The function compiles successfully but fails to print the desired pattern due to logical errors.

Your task is to debug the program to pass all the test cases.

```
1 // You can print the values to stdout for debugging
2 - public class DrawPattern{
3 -     public static void printPattern( int n ){
4         int i , j , print = 1;
5         for( i = 1 ; i <= n ; i ++ )
6             for( j = 1 ; j <= 2 * i ; j ++ )
7                 System.out.print( ( print ) );
8                 System.out.println( );
9     }
10 }
11 }
```

The screenshot shows an IDE interface with a project structure on the left and a code editor on the right.

**Project Structure:**

- OA1 [OA1\_sortArray] (~/面试题集/Ama...)
  - .idea
  - amazon\_oa
  - Amazon全解析
  - debug
  - my OA1 code
  - OA
  - oa\_donut\_mapper
  - out
  - src
    - Array
    - ArrayOperation
    - ArraySort
    - Digits
    - EvenOddPattern
    - PrintPattern2
    - PrintPattern3
    - ShortArray
    - SortArray
  - OA1\_sortArray.iml
- External Libraries

**Code Editor (PrintPattern3.java):**

```
1  /** * Created by yaqunyu on 10/24/16. */
2  *
3  */
4  public class PrintPattern3 {
5      public static void print3(int row) {
6          int x = 1;
7          for (int i = 1; i <= row; i++) {
8              for (int j = i; j > 0; j--) {
9                  System.out.print(x + " " + x);
10             }
11         }
12     }
13 }
14
15 public static void main(String[] args) {
16     print3(4);
17 }
18
19 }
```

A red box highlights the closing brace of the inner for loop at line 12. The code runs successfully in the run window below.

**Run Window:**

```
Run PrintPattern3
▶ /Library/Java/JavaVirtualMachines/jdk1.8.0_65.jdk/Contents/Home/bin/java ...
11
1111
111111
11111111
Process finished with exit code 0
```

第二种情况 输出 a ab abc abcd cout(ch++) 改为 cout(print++) ,

改为

```
char ch = 'a';
char print = ch;
for(int j = 0; j <=i; j++) {
    System.out.print((print++));
}
System.out.println("");
```

The screenshot shows the IntelliJ IDEA interface. On the left is the project tree for 'OA1 [OA1\_sortArray] (~/面试题集/Amazon\_OA)'. The 'src' folder contains several source files: Array, ArrayOperation, ArraySort, Digits, EvenOddPattern, PrintPattern2 (which is selected), ShortArray, and SortArray. Below these is 'OA1\_sortArray.iml' and 'External Libraries'. The right pane displays the code for 'PrintPattern2'. A red box highlights the following code block:

```
1  /*
2  * Created by yaqunyu on 10/24/16.
3  */
4  public class PrintPattern2 {
5      public static void print2(int row){
6          for(int i = 0; i < row; i++){
7              //char ch = 'a';
8              char ch = 'a';
9              char print = ch;
10             for(int j = 0; j <= i; j++){
11                 //System.out.println("At this time ch is: " + ch);
12                 //if char++ it will go with alpha beta order
13                 System.out.print((print++));
14             }
15         }
16     }
17 }
18 public static void main(String[] args){
19     print2(4);
20 }
```

The screenshot shows the 'Run' tab with the title 'PrintPattern2'. It lists the command used: '/Library/Java/JavaVirtualMachines/jdk1.8.0\_65.jdk/Contents/Home/bin/java ...'. Below the command is the output of the program, which prints four lines: 'a', 'ab', 'abc', and 'abcd'. At the bottom, it says 'Process finished with exit code 0'.

### 23. selectionSort selectionSortArray

ascending order, arr[index\_of\_min] > arr[x] 改为 arr[y]

distinctNum: 把==改成!=

sunDistince: 初始化 sum 数组写在 sort 之后

```
1 import java.util.Arrays;
2 class Distinct
3 {
4     int sumDistinct (int size, int[] inputArray)
5     {
6         Arrays.sort(inputArray);
7         int sum =inputArray[0];
8
9         int point = inputArray[0];
10        for(int i = 1 ; i < size ; i++)
11        {
12            if(point != inputArray[i])
13            {
14                sum+=inputArray[i];
15                point = inputArray[i];
16            }
17        }
18    }
19    return sum;
20 }
21 }
```

eliminatedVowel: default condition 后面的 i++ 删掉

checkPanlidrome: 把 result == num 改成 result == temp

DistinctElementCount: 感觉好像和之前看到的题库有些不同? ==改了!=还是不对。

**if (flag == 1) → if (flag == 0)**

MatrixSum: i++ 和 j++ 位置

LabelProduct: 加入字符串的方向反了

CheckPalindrome: return 的 num 改成 temp

SumArray: 初始化 sum 数组写在 sort 之后

median: arr2 改成 arr2[i-size]

countDays: 国年 (year %4 == 0) && (year %100 != 0) || (year%400 == 0)

elementRange: 把 || 改成 &&

countNumParity: (1) 主函数 result%2 ==0 (2) 主函数 if(result == 0) return 1 (3) sum digit 函数把 int temp = num %10 和 num = num/10 换一下顺序

countProduct: 改成 j < size

*countElement*: 改成 `i < len; i++,` 后面 `arr[i+1]` 改成 `arr[i]`

*ArmstrongNumber*: `Math.pow(remainder, digitcount)`

*reverseNum*: `reverseNum = reverseNum * 10 + i remainder`

*KtimeElement* : while Loop 结束后 `check if(count == k) res = element`

*distinctNumber\** == 改成 !=

*sumDistinct\** 把排序移到最前

*reverseString* 的题来着 不是 `str+=`

*eliminateVowel\** default case 不需要 `i++`

*checkPalindrome\** 改成 `result==tem`

*median\** 把 `i` 改成 `i-size`

*countDays\** 建议上网看看闰年的定义

*elementRange\** 把`&&`改成 |

*getDigitSumParity*: 1. 返回的判断条件写的都不对; 2. `getSum` 里面 `temp =` 的顺序要提前

1.1 判断是否为偶数的条件 `result%2!=0` 应该改为 `result%2==0`

1.2 当 `result` 为 0 时, 它返回了 0, 应该返回 1 才对

(1) 主函数 `if (result % 2 ==0)`

(2) 主函数 `if (result == 0) return 1`

(3) sum digit 函数把 `int temp = num %10` 和 `num = num/10` 换一下顺序

```

public class DigitSum {
    public int getDigitSumParity(int[] arr) {
        int min = getMin(arr);
        int result = getSum(min);
        if (result % 2 == 0) {
            return 1;
        }
        else {
            return 0;
        }
    }

    public int getMin(int[] arr) {
        if (arr == null || arr.length <= 0) {
            throw new IllegalArgumentException(
                "Input array should contain at least 1 element.");
        }
        int min = arr[0];
        for (int i = 1, len = arr.length; i < len; i++) {
            if (arr[i] < min) {
                min = arr[i];
            }
        }
        return min;
    }

    public int getSum(int num) {
        int sum = 0;
        while (num != 0) {
            int temp = num % 10;
            num = num / 10;
            sum = sum + temp;
        }
        return sum;
    }
}

```

replaceValues: 把判断条件改成==1 就行

Occurrence: 给 while loop 后加个 i++;

GradingSystem: 把||改成&&,

sortArray 把 if 条件的>改成<,

printPattern: 给 for loop 都加括号,

selectionsort: if 条件那里把 arr[x]改成 arr[y]

1. countOccurrence: while 循环里加上 i++
2. printPattern: 两个 for loop 括号加上
3. checkGrade: ||改成&&
4. reverseArray: +=改成-=
5. descendingSortArray: >改成<
6. drawPrintPattern: 第二个 for loop 后面分号删掉
7. selectionSortArray: 条件里面的 array[x]改成 array[y]

sumDistinct: 把 sort 提前

```

1 import java.util.Arrays;
2 class Distinct
3 {
4     int sumDistinct (int size, int[] inputArray)
5     {
6         Arrays.sort(inputArray);
7         int sum =inputArray[0];
8
9         int point = inputArray[0];
10        for(int i = 1 ; i < size ; i++)
11        {
12            if(point != inputArray[i])
13            {
14                sum+=inputArray[i];
15                point = inputArray[i];
16            }
17        }
18    }
19    return sum;
20 }
21 }
```

vowelsString: while 循环 i++放外面，最后的返回判断 len % 2 改成 len / 2

```

+ public class StringCheck {
+
    private static final Set<Character> VOWELS = new HashSet<>();
    static {
        VOWELS.add('a');
        VOWELS.add('e');
        VOWELS.add('i');
        VOWELS.add('o');
        VOWELS.add('u');
        VOWELS.add('A');
        VOWELS.add('E');
        VOWELS.add('I');
        VOWELS.add('O');
        VOWELS.add('U');
    }
+
    public int vowelsString(String inputstr) {
        if (inputstr == null) {
            return 0;
        }
        int i = 0, vcount = 0, len = inputstr.length();
        while (i < len) {
            if (VOWELS.contains(inputstr.charAt(i))) {
                vcount += 1;
            }
            i++;
        }
        if (vcount > (len / 2))
            return 1;
        else
            return 0;
    }
}
```

checkPalindrome: result = result \* 10 + remainder, 最后返回要比较 temp

```
1 class Solution
2 {
3     boolean checkPalindrome (int num)
4     {
5         int result=0;
6         int remainder = 0;
7         int temp = num;
8         while(num != 0)
9         {
10             remainder = num % 10;
11             result = result * 10 + remainder;
12             num=num/10;
13         }
14         return (result == temp);
15     }
16 }
```

matrixSum: i++拿到内循环外面

```
2 public class Matrix {
3     public int matrixSum(int[][] matrix) {
4         int m = matrix.length;
5         int n = matrix[0].length;
6         int i = 0, j, sum = 0;
7         while (i < m) {
8             j = 0;
9             while (j < n) {
10                 sum += matrix[i][j++];
11             }
12             i++;
13         }
14         return sum;
15     }
16 }
```

**matrixSum:** while(j < n) sum += matrix[i][j++]; put i++ out of the while(j < n) loop

countDays: 参照 wiki 上的写法, 按照 note 写只能过 11 个 case (总共 12 个)  
((y % 4) == 0 || (y % 100) != 0) 应改为 ((y % 4) == 0 && (y % 100) != 0)

(year % 4 == 0) && (year % 100 != 0) || (year % 400 == 0)

```

public class DaysCount
{
    public int countDays(int month, int year)
    {
        switch (month)
        {
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12:
                return 31;
            case 4:
            case 6:
            case 9:
            case 11:
                return 30;
            case 2:
                if (((year % 400 == 0 && year % 3200 != 0) ||
                    (year % 100 != 0)) &&
                    && (year % 4 == 0))
                    return 29;
                else
                    return 28;
            default:
                // Should not happen.
                return -1;
        }
    }
}

```

countElementRange: || -> &&

```

class Range
{
    int countElementRange (int size, int[] inputArray, int low, int high)
    {
        int count = 0;
        for(int i=0;i<size;i++)
        {
            if(inputArray[i] >= low && inputArray[i] <= high)
                count++;
        }
        return count;
    }
}

```

AppearsKTimes

return 前 check 一下指针是不是在 array 的最后

```

if (input[size - 1] == element && count++ == k) {
    res = element;
}

```

4. **AppearsK:**
  1. add follow code after while loop: if (count == k) res = element;
  2. switch contents in if and else, judge if(element != inputArray[i]) first

Mergewholists

## 16. mergeTwoLists:

```
1 *      ListNode next;
2 *      ListNode(int x) { val = x; }
3 */
4
5 class Solution {
6     public ListNode mergeTwoLists(ListNode l1, ListNode l2) {
7         if (l1 == null) {
8             return l2;
9         } else if (l2 == null) {
10            return l1;
11        } else if (l1.val <= l2.val) {
12            l1.next = mergeTwoLists(l1.next, l2);           // 小的下一个指向返回的头
13            return l1;          // 每次都要返回小的那个，作为newhead
14        } else if (l1.val > l2.val) {
15            l2.next = mergeTwoLists(l1, l2.next);
16            return l2;
17        }
18    }
19}
20}
21}
22}
23}
```

## selection sort

### 12. Selection sort

```
bug: if里面的判断它写的是arr[min]> arr[x]，改成arr[min]> arr[y]
for (x = 0; x < n; x++) {
    int index_of_min = x;
    for (y = x; y < n; y++) {
        if (arr[index_of_min] > arr[y]) {
            y = index_of_min;
        }
        int temp = arr[x];
        arr[x] = arr[y];
        arr[y] = temp;
    }
}
// selection sort in ascending order 升序
for (int x = 0; x < n; x++) {
    int index_of_min = x;
    for (int y = x; y < n; y++) {
        if (arr[index_of_min] > arr[y]) {
            index_of_min = y;
        }
        int temp = arr[x];
        arr[x] = arr[y];
        arr[y] = temp;
    }
}
// bubble sort in descending order 降序
for (int x = 0; x < n; x++) {
    for (int y = x; y < n; y++) {
        if (arr[x] < arr[y]) {
            int temp = arr[x];
            arr[x] = arr[y];
            arr[y] = temp;
        }
    }
}
```

### 24. reverseArray

改法 1: += 改成 -=

Problem	Test Cases	Output
<p>You can click on <i>Compile &amp; Run</i> anytime to check the compilation/execution status of the program. You can use <code>System.out.println</code> to debug your code.</p> <p>The submitted code should not just pass compilation but also should be logically correct, passing all the testcases.</p> <p>Do not write the <code>main()</code> method.</p> <p><code>java.util</code> package is allowed. Some Java packages like <code>java.lang.reflect</code> are not allowed to be used.</p> <p>An algorithm/Javadoc MAY HAVE BEEN provided to help you understand the problem. JRE 1.7 is used.</p> <p>For this question, you will need to correct the given implementation. <b>Do Not</b> attempt to modify the approach or incorporate library methods.</p> <p>A method <code>reverseArray(int arr[])</code> of class <code>SortArray</code> accepts an input array <code>arr</code> as an argument. The function is expected to reverse the elements of the input array in-place.</p> <p>For example if the input array <code>arr</code> is {20, 30, 10, 40, 50}, the function is expected to return {50, 40, 10, 30, 20}.</p> <p>The function compiles successfully but fails to return the desired result due to logical errors.</p>	<p>Save    Compile &amp; Run    Next Question    Navigate to question</p> <p>Reset</p> <pre>1 // You can print the values to stdout for debugging 2 public class SortArray{ 3     public static int[] reverseArray( int arr[] ){ 4         int i , temp , originalLen = arr.length; 5         int len = originalLen; 6         for( i = 0 ; i &lt; originalLen / 2 ; i ++ ){ 7             temp = arr[ len - 1 ]; 8             arr[ len - 1 ] = arr[i]; 9             arr[i] = temp; 10            len -= 1; 11        } 12        return arr; 13    } 14 }</pre>	

```

3  /*
4   * Created by yaqunyu on 10/21/16.
5   */
6  public class SortArray {
7      public static int[] reverseArray(int arr[]){
8          int i, temp, originallen = arr.length;
9          int len = originallen;
10         for(i=0; i<originallen/2; i++){
11             System.out.println("i is: "+ i);
12             //System.out.println("arr[i] is: "+ arr[i]);
13             temp = arr[len-1];
14             arr[len-1] = arr[i];
15             // System.out.println("arr[len-1] is: "+ arr[len-1]);
16             arr[i]=temp;
17             //System.out.println("after reverse arr[i] is: "+ arr[i]);
18             len -= 1;
19             //System.out.println("len is: "+ len);
20         }
21         return arr;
22     }
23     public static void main(String[] args){
24         int[] myIntArray = new int[]{20,2,3,5,9,6,16};
25         System.out.println("Original Array is [20,2,3,5,9,6,16]");
26         System.out.println("After revers array is: "+ Arrays.toString(reverseArray(myIntArray)));
27     }
28 }
29
30

```

改法2: if (arr[min] > arr[y]) for 循环中 arr[len - 1]改成 arr[len-i-1], 并且去掉 len+=1;  
for(i=0; i<len/2; i++) { temp=arr[len-1-i]; arr[len-1-i]=arr[i]; arr[i]=temp; }

```

1  import java.util.Arrays;
2
3  /*
4   * Created by yaqunyu on 10/21/16.
5   */
6  public class SortArray {
7      public static int[] reverseArray(int arr[]){
8          int i, temp, originallen = arr.length;
9          int len = originallen;
10         for(i = 0; i < originallen/2; i++){
11             System.out.println("i is: "+ i);
12             //System.out.println("arr[i] is: "+ arr[i]);
13             temp = arr[len-1-i];
14             arr[len-1-i] = arr[i];
15             // System.out.println("arr[len-1] is: "+ arr[len-1]);
16             arr[i]=temp;
17             //System.out.println("after reverse arr[i] is: "+ arr[i]);
18             //len -= 1;
19             //System.out.println("len is: "+ len);
20         }
21         return arr;
22     }
23     public static void main(String[] args){
24         int[] myIntArray = new int[]{20,2,3,5,9,6,16};
25         System.out.println("Original Array is [20,2,3,5,9,6,16]");
26         System.out.println("After revers array is: "+ Arrays.toString(reverseArray(myIntArray)));
27     }
28 }
29
30

```

## 25. drawPrintPattern

第二二个 for loop 后面的分号删掉

## 26. vowelsString vowelString

while 循环 i++ 放到外面面

while 循环少了 i++ ? ?

if (vcount > len % 2) 改成 if (vcount > (len / 2))

The screenshot shows a Java code editor with a tab bar at the top labeled "problem | Test Cases | Output |". Below the tabs is a status message: "You are required to fix all logical errors in the given code. You can click on Compile & Run anytime to check the compilation/execution status of the program. You can use System.out.println to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the main() function as it is not required." A note below says "Code Approach: For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods." The code itself is a class named StringCheck with a static final HashSet VOWELS containing 'a', 'e', 'i', 'o', and 'u'. It has a method vowelsString that takes a string inputstr and returns an integer. The logic counts the number of vowels in the string and returns 1 if it's more than half of the length, otherwise 0. A note states: "A method vowelsString(String inputstr) of class StringCheck accepts a string inputstr. This method counts the number of vowels in the input string inputstr. If the number of the vowels is more than half of the input string length then it returns 1 else it returns 0. The method compiles fine but fails to return desired result." Below the code editor is a toolbar with icons for file operations like Open, Save, and Print.

```
1 import java.util.HashSet;
2 import java.util.Set;
3 //CORRECT THE CODE GIVEN BELOW
4 public class StringCheck {
5
6     private static final Set<Character> VOWELS = new HashSet<>();
7+
8     static {
9         VOWELS.add('a');
10        VOWELS.add('e');
11        VOWELS.add('i');
12        VOWELS.add('o');
13        VOWELS.add('u');
14    }
15
16    public int vowelsString(String inputstr) {
17        if (inputstr == null) {
18            return 0;
19        }
20        int l = 0, vcount = 0, len = inputstr.length();
21        while (l < len) {
22            if (VOWELS.contains(inputstr.charAt(l))) {
23                vcount += 1;
24                l++;
25            }
26        }
27        if (vcount > (len % 2))
28            return 1;
29        else
30            return 0;
31    }
32}
33
34}
35}
36}
37}
```

## 27. mergeTwoLists

```
else if (l1.val <= l2.val){
    l1.next = mergeTwoLists(l1.next, l2);
    return l1;
}
```

The screenshot shows a Java code editor with a tab bar at the top. The code is for a class Solution with a static method mergeTwoLists that takes two ListNode objects as parameters. It handles three cases: if either list is null, it returns the other; if both are null, it returns null; and if the values are equal, it merges them by setting l1.next to the result of merging l1.next and l2, and returning l1. If the values are not equal, it merges them by setting l2.next to the result of merging l1 and l2.next, and returning l2. The code includes comments in Chinese explaining the logic. The code editor has a dark theme with syntax highlighting for Java keywords and comments.

```
*      ListNode next;
*
*      ListNode(int x) { val = x; }
*
*/
9 v class Solution {
0 v     public ListNode mergeTwoLists(ListNode l1, ListNode l2) {
1 v         if (l1== null ){
2 v             return l2;
3 v         }else if (l2 == null){
4 v             return l1;
5 v         } else if(l1.val <= l2.val) {
6 v             l1.next = mergeTwoLists(l1.next, l2);      //小的下一个指向返回的头
7 v             return l1;      //每次都要返回小的那个, 作为newhead
8 v         }else if(l1.val > l2.val){
9 v             l2.next = mergeTwoLists(l1,l2.next);
10            return l2;
11        }
12    }
13 }
```

## 28. arraySum

计算一个整数数组的和

sum = arr[i] 改成 sum += arr[i]

## 29. Remove element

for(i = index; i<len-1; i++){ arr[i]=arr[i+1] bug!!! } 改为 arr[i] = arr[i+1]

The screenshot shows a Java code editor with two panes. The left pane displays the code for a `removeElement` method. The right pane shows the original buggy code and the corrected version.

**Left Pane (Problem):**

```
1 // You can print the values to stdout for debugging
2 public class ShortArray{
3     public static int[] removeElement( int arr[], int index ){
4         int i , j , len = arr.length;
5         if( index < len ){
6             for( i = index ; i < len - 1 ; i ++ ){
7                 arr[i] = arr[ i + 1 ];
8             }
9             int rarr[] = new int[len-1];
10            for( i = 0 ; i < len - 1 ; i ++ )
11                rarr[i]= arr[i];
12            return rarr;
13        }
14        else
15            return arr;
16    }
17 }
```

**Right Pane (Code Editor):**

```
1 import java.util.Arrays;
2 /**
3 * Created by yaqunyu on 10/21/16.
4 */
5 public class ShortArray {
6     public static int[] removeElement(int arr[], int index) {
7         int i, j, len = arr.length;
8         if (index < len) {
9             for (i = index; i < len - 1; i++) {
10                 System.out.println("Index now is: " + i);
11                 //if arr[i] = arr[i+1] then after assign value, i will automatically increase 1
12                 // in our case, index = 3, after arr[3] = arr[4] i = 5
13                 arr[i] = arr[i+1];
14                 System.out.println("arr[i] is: " + arr[i]);
15             }
16             System.out.println("Array now is: " + Arrays.toString(arr));
17             int rarr[] = new int[len - 1];
18             for (i = 0; i < len - 1; i++)
19                 rarr[i] = arr[i];
20             return rarr;
21         } else
22             return arr;
23     }
24 }
25
26 public static void main(String[] args) {
27     int[] myIntArray = new int[]{20, 2, 3, 5, 9, 6, 16};
28     System.out.println("Original Array is [20,2,3,5,9,6,16]");
29     System.out.println("After revers array is: " + Arrays.toString(removeElement(myIntArray, 3)));
30 }
31 }
```

The screenshot shows a Java code editor with two panes. The left pane displays the code for a `removeElement` method. The right pane shows the original buggy code and the corrected version.

**Left Pane (Problem):**

```
1 // You can print the values to stdout for debugging
2 public class ShortArray{
3     public static int[] removeElement( int arr[], int index ){
4         int i , j , len = arr.length;
5         if( index < len ){
6             for( i = index ; i < len - 1 ; i ++ ){
7                 arr[i] = arr[ i + 1 ];
8             }
9             int rarr[] = new int[len-1];
10            for( i = 0 ; i < len - 1 ; i ++ )
11                rarr[i]= arr[i];
12            return rarr;
13        }
14        else
15            return arr;
16    }
17 }
```

**Right Pane (Code Editor):**

```
1 import java.util.Arrays;
2 /**
3 * Created by yaqunyu on 10/21/16.
4 */
5 public class ShortArray {
6     public static int[] removeElement(int arr[], int index) {
7         int i, j, len = arr.length;
8         if (index < len) {
9             for (i = index; i < len - 1; i++) {
10                 System.out.println("Index now is: " + i);
11                 //if arr[i] = arr[i+1] then after assign value, i will automatically increase 1
12                 // in our case, index = 3, after arr[3] = arr[4] i = 5
13                 arr[i] = arr[i+1];
14                 System.out.println("arr[i] is: " + arr[i]);
15             }
16             System.out.println("Array now is: " + Arrays.toString(arr));
17             int rarr[] = new int[len - 1];
18             for (i = 0; i < len - 1; i++)
19                 rarr[i] = arr[i];
20             return rarr;
21         } else
22             return arr;
23     }
24 }
25
26 public static void main(String[] args) {
27     int[] myIntArray = new int[]{20, 2, 3, 5, 9, 6, 16};
28     System.out.println("Original Array is [20,2,3,5,9,6,16]");
29     System.out.println("After revers array is: " + Arrays.toString(removeElement(myIntArray, 3)));
30 }
31 }
```

## 30. Even odd pattern

even 和 odd 处的 for loop 缺少大大括号

**Problem** Test Cases Output

of the program. You can use System.out.println to debug your code.

The submitted code should not just pass compilation but also should be logically correct, passing all the testcases.

Do not write the `main()` method.  
`java.util` package is allowed. Some Java packages like `java.lang.reflect` are not allowed to be used.  
An algorithm/Javadoc MAY HAVE BEEN provided to help you understand the problem. JRE 1.7 is used.

For this question, you will need to correct the given implementation. **Do Not** attempt to modify the approach or incorporate library methods.

A method `printPattern(int num)` of class `EvenOddPattern` prints even or odd numbers based on the value of the input argument `num` (`num ≥ 0`).

If the input number `num` is even, the function is expected to print the first `num` even whole numbers and in case it is odd, is expected to print the first `num` odd numbers. For example: given input 2, the function prints the string "0 2" (without quotes).

The function compiles successfully but fails to print the desired result due to logical errors.

Save Compile & Run Next Question Navigate to question

```
1 // You can print the values to stdout for debugging
2 public class EvenOddPattern{
3     public static void printPattern( int num ){
4         int i , print = 0 ;
5         if( num % 2 == 0 ){
6             print = 0;
7             for( i = 0 ; i < num ; i ++ )
8                 System.out.print( print + " " );
9             print += 2;
10        }
11        else{
12            print = 1;
13            for( i = 0 ; i < num ; i ++ )
14                System.out.print( print + " " );
15            print += 2;
16        }
17    }
18 }
```

```
1 /**
2  * Created by yaqunyu on 10/21/16.
3 */
4 public class EvenOddPattern {
5     public static void printPattern(int num) {
6         int i, print =0;
7         if(num % 2 == 0){
8             print = 0;
9             for(i = 0; i<num; i++) {
10                 System.out.print(print + " ");
11                 print += 2;
12             }
13         }
14         else {
15             print = 1;
16             for (i = 0; i < num; i++) {
17                 System.out.print(print + " ");
18                 print += 2;
19             }
20         }
21     }
22     public static void main(String[] args){
23         printPattern(2);
24     }
25 }
```

### 31. Manchester

这个有两处错误，是要判断 `array` 里每位数字和前一位是不是相等，`index0` 的数字单独和 0 判断一下，所以 0 那句不能直接 `res[0] = 0` 要 `res[0] = (arr[0] != 0) ? 1:0`，后面 for 循环里判断是要把 `arr == arr[i-1]` 改成 `arr != arr[i-1]`

**Manchester Array:** 给一个 `input array`, 如果 `A[i] != A[i-1]`, return 1, 反之 return 0, 需要 output 另一个 array

`result = (A[i] == A[i-1]); => result = A[i] != A[i-1];`

`output[i] = result ? 1 : 0;`

`ret[0]` 也要加一下，不然有一个 case 过不了

### 32. Remove duplicate from unsorted array

报错是 Index out of bound, 因为 for 循环里有 `k < length`, 下面却使用 `arr[k+1]` 改为 `k < length - 1`

### 33. Remove duplicates

循环下标从 `i+1` 开始

### 34.Insertion sort

```
if (arr[i-1] > arr[i])
    while (j > 0 && arr[j - 1] > temp)
```

insertion sort < 改 > 或者 > 改 <

```
for (int i = 1; i < n; i++) {
    if (arr[i - 1] > arr[i]) {
        int temp = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > temp) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = temp;
    }
}
```

### 35.countA

```
if(c == 'A' || c == 'a')
```

**Problem | Test Cases | Output |**

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the `main()` function as it is not required.

**Code Approach:** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods. A method `countA(String str)` of class `ACount` is written such that it returns the number of occurrences of uppercase 'A' or lowercase 'a' in the input string `str`.

The method compiles fine but fails to return the desired result. Your task is to modify the code to give the correct result for all the cases.

**Compile and Run**

```
1 // You can print the values to stdout for debugging
2 public class ACount {
3     public int countA(String str) {
4         if (str == null) {
5             return 0;
6         }
7         int count = 0;
8         for (int i = 0, len = str.length(); i < len; i++)
9             char c = str.charAt(i);
10            if (c == 'A' && c == 'a') {
11                count++;
12            }
13        }
14    return count;
15 }
16 }
```

**QUESTION:**

### 36.countDigits

这里面到最后 num 直接是 0 , 造成 bug , 需要用一个数来储存 num 和执行运算

**Problem | Test Cases | Output |**

**i** You are required to correct the logical/syntactical mistake in the given Java code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use `System.out.println` to debug your code.

The submitted code should not just pass compilation but also should be logically correct, passing all the testcases.

Do not write the `main()` method.

`java.util` package is allowed. Some Java packages like `java.lang.reflect` are not allowed to be used.

An algorithm/Javadoc MAY HAVE BEEN provided to help you understand the problem. JRE 1.7 is used.

For this question, you will need to correct the given implementation. **Do Not** attempt to modify the approach or incorporate library methods.

A method `countDigits(int num)` of class `Digits` returns the remainder when the input argument `num` (`num > 0`) is divided by the number of digits in `num`.

The function compiles successfully but fails to return the desired result due to logical errors.

Your task is to debug the program to pass all the test cases.

**Save | Compile & Run | Submit Code Test | Navigate to question**

```
1 // You can print the values to stdout for debugging
2 public class Digits{
3     public static int countDigits( int num ){
4         int count = 0;
5         while( num != 0 ){
6             num = num / 10;
7             count++;
8         }
9         return ( num % count );
10    }
11 }
```

```
3  */  
4 //function is to dived num by its count  
5 public class Digits {  
6     public static int countDigits(int num){  
7         int count = 0;  
8         int temp = num;  
9         while(temp != 0){  
10             //to check the how many digit is num  
11             //however if num = num/10 then at the last step of while, the num will always be 0  
12             //so that is the error  
13             temp = temp/10;  
14             count++;  
15         }  
16         //use operator % will get the result of remind  
17         return (num%count);  
18     }  
19     public static void main(String[] args){  
20         System.out.println("Remain result for count digit 235 is: "+countDigits(235));  
21     }  
22 }  
23 //System.out.println("count is: "+ count);  
24 //System.out.println("temp is: "+ temp + " num is: "+num);  
25 //System.out.println("result is: "+ num/count);  
26
```

### 37.arrayOperation

<和>写反了