1 L3A1-1

1.1 Problem description : Logic between methods to accomplish the problem:

The program first calls the isValid method through the main method to determine if the credit card number entered by the user is valid` The isValid method first calls sumOfDoubleEvenPlace and sumOfOddPlace to calculate the sum of even and odd digits in the card number, respectively. After adding the two, it continues to check if the length of the card number is appropriate, and calls the prefixMatched method to verify if the card number starts with the specified prefix (4, 5, 6, or 37)` The prefix of the card number is extracted using the 'getInitialize' method, while the 'getSize' method is used to obtain the total length of the card number. These methods work together to verify the validity of credit card numbers based on the Luhn algorithm and prefix rules.

I think if we can understand what the methods will be and how to connect each other can help programmer make the programs work.

1.2 Analysis: Solving my own question : At the end of whole codes, MY COMMENTS shows my methods before I learnt how to use `String.valueOf and StringBuilder.` `The longToDigits method breaks down long type numbers into individual digits and stores them in an array` The reverseArray method inverts the array. But there are easy ways in Java.

1.3 Source codes:

```
package L3A1;

import java.util.Scanner;

public class q1 {

    public static void main(String[] args) {
        System.out.print("Enter a credit card number as a long integer:");
        long number;
        Scanner input = new Scanner(System.in);
        number = input.nextLong();

        // Example credit card numbers
        // 4388576018402626
        // 5117275325077359
        if (isValid(number))
            System.out.printf(number + " is valid");
        else
            System.out.printf(number + " is invalid");
    }

    /** Check if the card number is valid */
    public static boolean isValid(long number) {
        int stepTwo = sumOfDoubleEvenPlace(number);
        int stepThree = sumOfOddPlace(number);
```

```
        int sum_2_3 = stepTwo + stepThree;
        int size = getSize(number);

        if (size >= 13 && size <= 16) {
            // Check if the sum is divisible by 10
            if (sum_2_3 % 10 == 0) {
                // Check if the card starts with 4, 5, 6 or 37
                if (prefixMatched(number, 1) || prefixMatched(number, 2)) {
                    return true; // All conditions met, return true
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } else {
            return false;
        }
    }

    /** Calculate the sum of double even place digits */
    public static int sumOfDoubleEvenPlace(long number) {
        String cardNumber = new
StringBuilder(String.valueOf(number)).reverse().toString(); // Reverse the card
number
        int sum = 0;
        for (int i = 1; i < cardNumber.length(); i += 2) { // Start from the
second digit to get each even place
            int digit = Character.getNumericValue(cardNumber.charAt(i)); // Get
the current digit
            sum = sum + getDigit(digit * 2); // Double the digit and add its
value
        }
        return sum;
    }

    /** Return the digit or the sum of the two digits if > 9 */
    public static int getDigit(int number) {
        if (number < 10) {
            return number;
        } else {
            return number / 10 + number % 10; // Return the sum of tens and
units
        }
    }

    /** Calculate the sum of odd place digits */
    public static int sumOfOddPlace(long number) {
```

```
        String cardNumber = new
StringBuilder(String.valueOf(number)).reverse().toString(); // Reverse the card
number
        int sum = 0;
        for (int i = 0; i < cardNumber.length(); i += 2) { // Get odd place
digits
            int digit = Character.getNumericValue(cardNumber.charAt(i)); // Get
the current digit
            sum = sum + digit;
        }
        return sum;
    }

    /** Check if the digit d is a prefix of the number */
    public static boolean prefixMatched(long number, int d) {
        long prefix = getPrefix(number, d);
        if (d == 1) {
            return prefix == 4 || prefix == 5 || prefix == 6; // Check the first
digit
        } else if (d == 2) {
            return prefix == 37; // Check the first two digits
        }
        return false;
    }

    /** Return the number of digits in d */
    public static int getSize(long d) {
        return String.valueOf(d).length(); // Count the digits
    }

    /** Return the first k digits from the number */
    public static long getPrefix(long number, int k) {
        int size = getSize(number);
        if (size < k) {
            return number; // If there are less than k digits, return the whole
number
        }
        // Convert the number to a string, take the first k digits, and convert
back to long
        String numStr = String.valueOf(number);
        return Long.parseLong(numStr.substring(0, k));
    }
}


// Before I learn how to use String.valueOf and StringBuilder, the following
are my solutions
// //the following two methods are my methods to help me accomplish the program
```

```
// //This method mainly converts integers of type long into their corresponding
array types
//
// public static int[] longToDigits(long number) {
//
//    int length = (int) Math.log10(number) + 1; // calculate the length of
the number
//
//     int[] digits = new int[length];// create a new array
//
//     for (int i = length - 1; i >= 0; i--) {
//         digits[i] = (int) (number % 10);  // stay the last number each time
//         number = number / 10;    // kick out the last number then
//     }
//     return digits;
// }
// //This method is to invert the return array of the longToDigits function for
subsequent calculations
// public static int[] reverseArray(long number) {
//     int[] array = longToDigits(number);
//     int[] arr_rev = new int[array.length];
//     for (int i = 0; i < array.length; i++)
//        arr_rev[i] = array[array.length - 1 - i];
//     return arr_rev;
// }
```

1.4 Screenshot

```
Enter a credit card number as a long integer:5117275325077359
5117275325077359 is valid
```

2 L3A1-2

2.1 Problem description: I set the array for 5 digits, the same as example;

2.1.1 reverseArray : Using the double pointer method, start from both ends of the array and invert it by swapping elements.

2.1.2 rotateArray : First, calculate the number of bits that need to be rotated, construct a new array, and use modular operations to rearrange the original array elements in their new positions, achieving a right-handed operation.

2.1.3 sortArray: The bubble sort algorithm was used to compare adjacent elements and swap them through two layers of loops, resulting in an ordered array.

2.1.4 largestElement : By traversing the array and comparing them one by one, the current maximum value is updated in real-time, and the maximum element in the array is finally

output.

2.2 Analysis :

'int [] nums2=Arrays. copyOf (nums, nums. length)' is used to create an independent copy of the array and avoid modifying the original array 'nums'. And ` int [] nums1=nums` Just creating a reference to the same array, so 'nums1' and 'nums' actually point to the same array.

When you call reverseArray (nums1), both nums1 and nums will be reversed because they point to the same array. So if you don't want nums2 to be affected by changes in nums, you need to create a separate copy using Arrays. copyOf so that nums2 won't be modified by reverseArray or other methods. This is different from the C language I learned before

2.3 Source code:

```
package L3A1;



import java.util.Arrays;

import java.util.Scanner;



public class q2 {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int[] nums = new int[5]; // Array for user input



        System.out.println("Input array: ");

        for (int i = 0; i < 5; i++) {

            nums[i] = input.nextInt(); // Read integers into the array

        }



        // Create references for manipulation

        int[] nums1 = nums; //this is not to create a new array, but point nums to a new num1
```

```
        int[] nums2 = Arrays.copyOf(nums, nums.length);
        // so we can just copy nums to nums2, or we rotate the reversed one
        int[] nums3 = nums;
        int[] nums4 = nums;

        reverseArray(nums1); // Reverse the user input array
        rotateArray(nums2, 2); // Rotate the predefined array by 2
        sortedArray(nums3); // Sort the user input array
        largestElement(nums4); // Find the largest element
    }

    /** Reverse the array */
    public static void reverseArray(int[] nums) {
        int length = nums.length;
        for (int i = 0; i < length / 2; i++) {
            // Swap elements
            int temp = nums[i];
            nums[i] = nums[length - i - 1];
            nums[length - i - 1] = temp;
        }
        // Print reversed array
        System.out.println("\nReversed array: ");
        for (int num : nums) {
            System.out.print(num + " ");
        }
```

```
}

/** Rotate the array to the right */
public static void rotateArray(int[] nums, int places) {
    int length = nums.length;
    places = places % length; // Adjust places if larger than length

    int[] rotated = new int[length];
    // Shift elements
    for (int i = 0; i < length; i++) {
        rotated[(i + places) % length] = nums[i];
    }
    // Print rotated array
    System.out.println("\nRotated array:");
    for (int num : rotated) {
        System.out.print(num + " ");
    }
}

/** Sort the array in ascending order */
public static void sortedArray(int[] nums) {
    // Simple bubble sort
    for (int i = 0; i < nums.length; i++) {
        for (int j = i + 1; j < nums.length; j++) {
            if (nums[i] > nums[j]) { // Swap if necessary
```

```
                    int temp = nums[j];
                    nums[j] = nums[i];
                    nums[i] = temp;
                }
            }
        }
        // Print sorted array
        System.out.println("\nSorted array:");
        for (int num : nums) {
            System.out.print(num + " ");
        }
    }

    /** Find and print the largest element */
    public static void largestElement(int[] nums) {
        int largest = nums[0];
        for (int i = 1; i < nums.length; i++) {
            if (largest < nums[i]) {
                largest = nums[i]; // Update largest
            }
        }
        // Print largest element
        System.out.println("\nThe biggest element is " + largest);
    }
}
```

2.4 Screenshot

```
Input array:
2 30 11 99 57

Reversed array:
57 99 11 30 2
Rotated array:
99 57 2 30 11
Sorted array:
2 11 30 57 99
The biggest element is 99
```

3 L3A1-3

3.1 Source code

```
package L3A1;

import java.util.Scanner;

public class q3 {

   public static void main(String[] args) {
      // TODO Auto-generated method stub
      Scanner input = new Scanner(System.in);
      System.out.print("Enter the number of values: ");
      int num = input.nextInt();
      int[] values = new int[num];
      System.out.print("Enter the number: ");
      for(int i = 0; i < num; i++) {
         values[i] = input.nextInt();
      }
      if(isConsecutiveFour(values) == true) {
         System.out.print("The list has consecutive fours ");
      }
      else System.out.print("The list has no consecutive fours ");
   }

   public static boolean isConsecutiveFour(int[] values) {
      // TODO: write your code here
      int flag = 0;
      for (int i = 0; i < values.length - 3; i++) {
         if (values[i] == values[i+1] && values[i] == values[i+2] &&values[i]
== values[i+3]) {
            flag = 1;
         }
         else continue;
      }
      if (flag == 1)
```

```
            return true;
        else return false;
    }
}
```

3.2 Screenshot

```
Enter the number of values: 7
Enter the number: 2 2 6 6 6 6 5
The list has consecutive fours
```

```
Enter the number of values: 9
Enter the number: 2 3 3 5 7 7 7 9 7
The list has no consecutive fours
```

4 L3A1-4

4.1 Source code:

```
package L3A1;

import java.util.Scanner;

public class q4 {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
         System.out.print("Enter strings: ");
         String userInput = input.nextLine();

         String[] strings = userInput.split(",\\s*");

         System.out.print("We have: ");
         for (int i = 0; i < strings.length; i++) {
             System.out.print(strings[i]);
             if (i < strings.length - 1) {
                 System.out.print(", ");
             }
         }

         String longestString = findLongestString(strings);
         System.out.println("\nThe longest string was: " + longestString);
    }

    public static String findLongestString(String[] strings) {
```

```
        String longest = "";
        for (String s : strings) {
            if (s.length() >= longest.length()) {
                longest = s;
            }
        }
        return longest;
    }

}
```

4.2 Screenshot

```
Enter strings: Mountaineering, Gymnastics, Archery, Fencing, Skateboarding, Wrestling
We have: Mountaineering, Gymnastics, Archery, Fencing, Skateboarding, Wrestling
The longest string was: Mountaineering
```