LAB 3 – ASSIGNMENT 1

PROBLEM 1

Description:

- Credit card numbers are composed of 13 to 16 digits and their first digit identifies the card type (Visa=4, Master=5, American Express=37, Discover=6).
- To validate a credit card number, first double every second digit starting from the right and add the two digits if the result is a two-digit number.
- Then, add the resulting single-digit numbers and add all the odd-positioned digits from right to left.
- Finally, the sum of both results must be divisible by 10 for the credit card number to be considered valid.

Analysis:

- Parsed the input as a string for easy manipulation.
- Checked if the length is between 13 and 16.
- Separated the prefix number and identified the card type.
- Applied Mod 10 check to identify validity of card number.
- During input, noticed some exceptions could occur, therefore handled exceptions using try catch.
- Added feasibility to enter input again in case of incorrect inputs in the same run.

The main difficulty I faced in this problem was trying to understand how to implement all the provided functions. That took a lot of time to distribute the logic correspondingly.

Source Code:

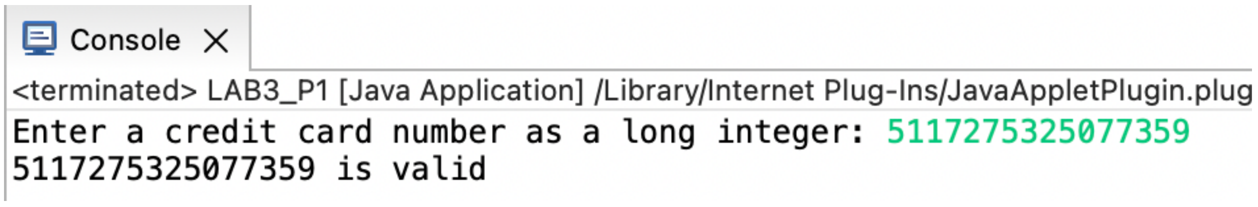
```
package edu.northeastern.csye6200;

import java.util.Scanner;

public class LAB3_P1 {

public static void main(String[] args) {

boolean flag = true;

Scanner sc = new Scanner(System.in);

while(flag) {

try {

System.out.print("Enter a credit card number as a long integer: ");
long number = sc.nextLong();

if(isValid(number)) {

System.out.println(number + " is valid");

break;

}

else {

System.out.println(number + " is invalid");

break;

}

}catch (Exception e) {

System.out.println("Incorrect Input! Input has to be a Long Integer \n
");

sc.next();

}

}

System.exit(1);

}

/** Return true if the card number is valid */

public static boolean isValid(long number) {

boolean isValid = false;

isValid = (getSize(number)>=13 && getSize(number)<=16) &&

(prefixMatched(number, 4) || prefixMatched(number, 5) ||
prefixMatched(number, 37) || prefixMatched(number, 6)) &&

( (sumOfDoubleEvenPlace(number) + sumOfOddPlace(number)) %10 == 0);

return isValid;
```

```
}

/** Get the result from Step 2 */

public static int sumOfDoubleEvenPlace(long number) {

int sum = 0;

String num = String.valueOf(number);

for (int i = getSize(number) -2;i>=0;i-=2) {

sum += getDigit(Integer.parseInt(num.charAt(i)+"") *2);

}

return sum;

}

/**

* Return this number if it is a single digit, otherwise, return the
sum of

* the two digits

*/

public static int getDigit(int number) {

if(number < 9)

return number;

else

return number / 10 + number % 10;

}

/** Return sum of odd place digits in number */

public static int sumOfOddPlace(long number) {

int sum = 0;

String num = String.valueOf(number);
```

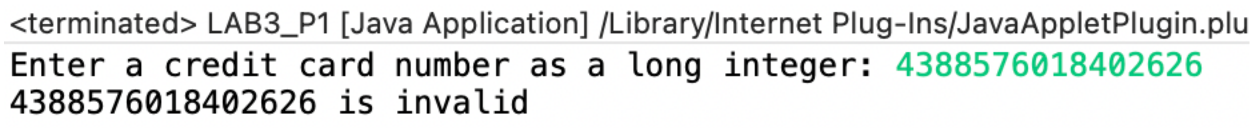
```
for (int i = getSize(number) - 1; i >= 0; i -= 2) {

sum += Integer.parseInt(num.charAt(i) + "");

}

return sum;

}

/** Return true if the digit d is a prefix for number */

public static boolean prefixMatched(long number, int d) {

return getPrefix(number,getSize(d)) ==d;

}

/** Return the number of digits in d */

public static int getSize(long d) {

int count = 0;

while(d>0) {

d = d/10;

count++;

}

return count;

}

/**

* Return the first k number of digits from number. If the number of
digits

* in number is less than k, return number.

*/

public static long getPrefix(long number, int k) {

if(getSize(number) > k) {
```

```
String num = String.valueOf(number);

return Long.parseLong(num.substring(0,k));

}

return number;

}

}
```

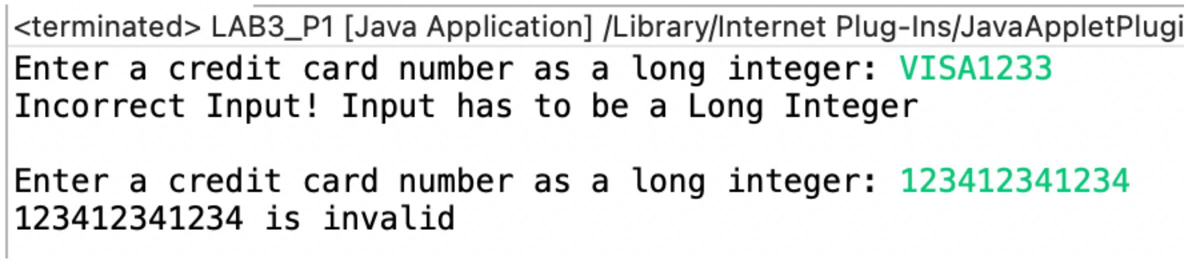
Output:



```
Console
<terminated> LAB3_P1 [Java Application] /Library/Internet Plug-Ins/JavaAppletPlugin.plug
Enter a credit card number as a long integer: 5117275325077359
5117275325077359 is valid
```



```
<terminated> LAB3_P1 [Java Application] /Library/Internet Plug-Ins/JavaAppletPlugin.plu
Enter a credit card number as a long integer: 4388576018402626
4388576018402626 is invalid
```



```
<terminated> LAB3_P1 [Java Application] /Library/Internet Plug-Ins/JavaAppletPlugi
Enter a credit card number as a long integer: VISA1233
Incorrect Input! Input has to be a Long Integer

Enter a credit card number as a long integer: 123412341234
123412341234 is invalid
```

PROBLEM 2

Description:

- Need to ask the user to input the size, i.e., the number of digits and then take in an array of numbers.
- Then Determine if there are four consecutive numbers with same value in the entered array.

Analysis:

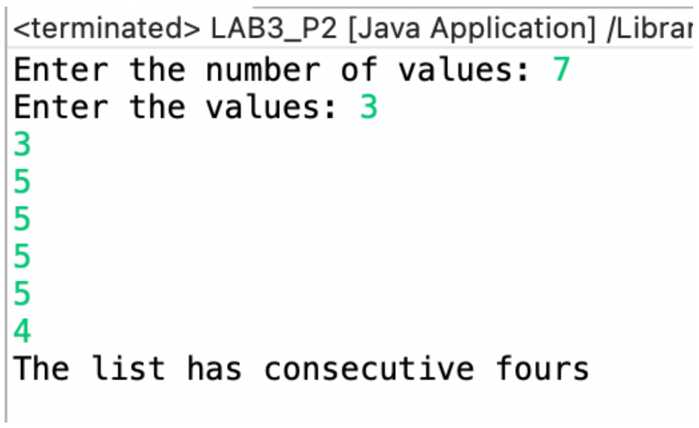
The array values are traversed, and each value is compared to the next value.
If their value matches, we increment the count value which is initialized with value 1.
If the value doesn't match, the count value is reset to 1.
If the count value equals to 4, we exit from the loop and return true.

Realized I don’t need to check if count value is greater than 4. That case will never occur. No significant difficulties otherwise.

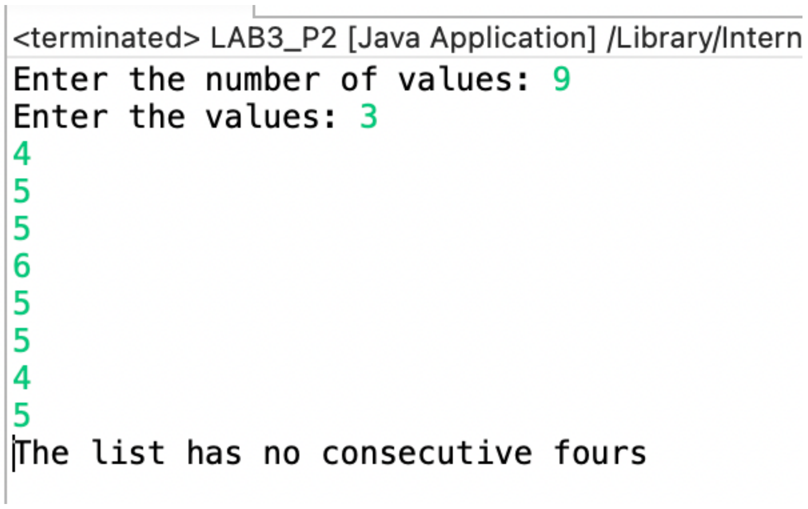
Source code:

```
package edu.northeastern.csye6200;

import java.util.Scanner;

public class LAB3_P2 {

public static void main(String[] args) {

boolean flag = true;

Scanner sc = new Scanner(System.in);

while(flag) {

try {

System.out.print("Enter the number of values: ");

int val = sc.nextInt();

if(val < 4) {

System.out.println("Cannot check consecutive fours with "+val+"
numbers");

break;

}

int[] values = new int[val];

System.out.print("Enter the values: ");

for (int i = 0; i < values.length; i++)

values[i] = sc.nextInt();

if(isConsecutiveFour(values)){

System.out.println("The list has consecutive fours");

break;
}

else{

System.out.println("The list has no consecutive fours");

break;

}

}catch (Exception e) {

System.out.println("Invalid Input! Kindly enter only numbers! \n");

sc.next();

}

}

System.exit(1);

}

public static boolean isConsecutiveFour(int[] values) {

int count = 1;

for(int i=1; i<values.length;i++) {

if(values[i-1] == values[i])

count++;

else

count = 1;

if(count >=4)

return true;

}

return false;

}
```

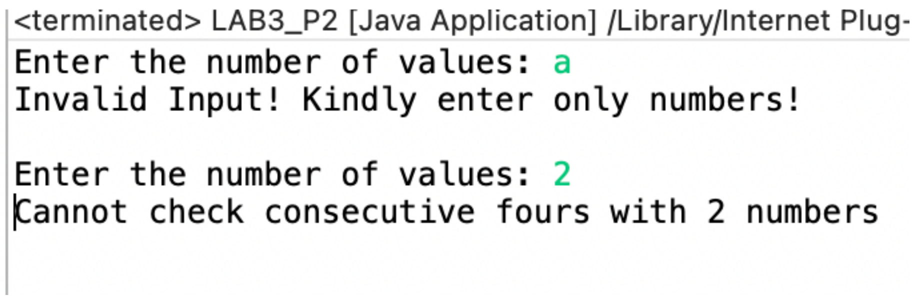
```
}
```



```
<terminated> LAB3_P2 [Java Application] /Librar
Enter the number of values: 7
Enter the values: 3
3
5
5
5
5
4
The list has consecutive fours
```



```
<terminated> LAB3_P2 [Java Application] /Library/Intern
Enter the number of values: 9
Enter the values: 3
4
5
5
6
5
5
4
5
The list has no consecutive fours
```



```
<terminated> LAB3_P2 [Java Application] /Library/Internet Plug-
Enter the number of values: a
Invalid Input! Kindly enter only numbers!

Enter the number of values: 2
Cannot check consecutive fours with 2 numbers
```