1)Generate series and find Nth element:

```
package com.w3epic.wiprotraining;
import java.io.*;
import  java.util.*;
class GenerateSeriesAndFindNthElement {
        public int seriesN(int input1,int input2,int input3,int input4){
                int gap1 = (input2 - input1);
                int gap2 = (input3 - input2);
                int output = input1;
                for (int i = 1; i < input4; i++) {
                        if (i % 2 == 1)
                                output += gap1;
                        else
                                output += gap2;
                        System.out.print(output + ", ");
                }
                return output;
        }
}
```

2)Find result after alternate add_sub on N

```
package com.w3epic.wiprotraining;
import java.io.*;
import java.util.*;
class FindResultAfterAlternateAdd_subOnN {
        public int AddSub(int input1,int input2){
                int N = input1;
                int result = N;
                int var = 0;
                if (input2 == 1) var = 1;
                else var = 0;
                for (int i = N - 1, j = 0; i >= 1; i--, j++) {
                        if (j % 2 == var) result += i;
                        else result -= i;
                        System.out.println(result + " ");
                }
                return result;
        }
}
```

3)Find Password (stable unstable)

```
package com.w3epic.wiprotraining;
import java.io.*;
import  java.util.*;
class FindPasswordStableUnstable {
	public int findPassword(int input1,int input2,int input3,int input4,int input5){
		int sumOfStable = 0;
		int sumOfUnstable = 0;

		if (isStable(input1)) sumOfStable += input1;
		else sumOfUnstable += input1;

		if (isStable(input2)) sumOfStable += input2;
		else sumOfUnstable += input2;

		if (isStable(input3)) sumOfStable += input3;
		else sumOfUnstable += input3;

		if (isStable(input4)) sumOfStable += input4;
		else sumOfUnstable += input4;

		if (isStable(input5)) sumOfStable += input5;
		else sumOfUnstable += input5;

		System.out.println(sumOfStable + " :: " + sumOfUnstable);
		System.out.println("isStable: " + isStable(input1) + isStable(input2) +
isStable(input3) + isStable(input4) + isStable(input5));
		return sumOfStable - sumOfUnstable;
	}
	public static boolean isStable(int num) {
		boolean isStable = true;
		int[] freq = new int[10];
		String numStr = String.valueOf(num);

		for (int i = 0; i < numStr.length(); i++) {
			freq[Integer.parseInt(String.valueOf(numStr.charAt(i)))]++;
		}

		System.out.println(Arrays.toString(freq));

		int firstFreq = 0;
		for (int i = 0; i < 10; i++) {
			if (freq[i] > 0) {
				firstFreq = freq[i];
```

```
                    break;
                }
            }
            System.out.println("firstFreq: " + firstFreq);

            for (int i = 0; i < 10; i++) {
                if (freq[i] != 0 && freq[i] != firstFreq) {
                    isStable = false;
                    break;
                }
            }
            System.out.println("isStable: " + isStable);

            return isStable;
        }
}
```

4) Calculate the sum of non-prime index values

```
package com.w3epic.wiprotraining;
import java.io.*;
import  java.util.*;
class CalculateSumOfNonPrimeIndexValues {
        public int sumOfNonPrimeIndexValues(int[] input1,int input2){
            int sum = 0;
            for (int i = 0; i < input2; i++) {
                if (!isPrime(i)) {
                    System.out.print(i + ":: " + input1[i] + " ");
                    sum += input1[i];
                }
            }
            return sum;
        }
        public static boolean isPrime (int input1) {
            if (1 == input1 || 0 == input1) return false;
            for (int i = 2; i < input1; i++) {
                if (i == input1) continue;
                if (input1 % i == 0) {
                    return false;
                }
            }
            return true;
        }
}
```

5) Find the one digit to be removed to form palindrome

```
package com.w3epic.wiprotraining;
import java.io.*;
import java.util.*;
class FindTheOneDigitToBeRemovedToFormPalindrome {
        public int digitRemove_Palin(int input1){
                StringBuilder num = new StringBuilder(String.valueOf(input1));
                for (int i = 0; i < num.length(); i++) {
                        if (palindromeCheck(num.toString())) return -1;
                        char removedChar = num.charAt(i);
                        String newNum = num.deleteCharAt(i).toString();
                        if (palindromeCheck(newNum)) {
                           System.out.println(i + ":: " + newNum + " :: " + removedChar);
                                return Integer.parseInt(String.valueOf(removedChar));
                        } else {
                                num.insert(i, removedChar);
                        }
                }
                return -1;
        }
        public static boolean palindromeCheck(String input1) {
                input1 = input1.toLowerCase();
                int digitCount = input1.length();
                boolean isPalindrome = true;
                int range = digitCount / 2;
                if (digitCount % 2 == 0) range--;
                for (int i = 0; i <= range; i++) {
                        if (input1.charAt(i) != input1.charAt(digitCount - i - 1)) isPalindrome =
false;
                }
                return isPalindrome;
        }
}
```

6)The “Nambiar Number” Generator

```
package com.w3epic.wiprotraining;
import java.io.*;
import  java.util.*;
class TheNambiarNumberGenerator {
        public int nnGenerator(String input1){
                String mobileNo = input1;
                StringBuilder numbiarNo = new StringBuilder();
                for (int i = 0; i < mobileNo.length(); i++) {
                        int firstDigit = Integer.parseInt(String.valueOf(mobileNo.charAt(i)));
                        int firstDigitEvenOrOdd = firstDigit % 2 == 0 ? 0 : 1; // even=0; odd=1
                        int sum = firstDigit;
                        int j = i + 1;
                        if (j == mobileNo.length()) {
                            numbiarNo.append(firstDigit);
                            break;
                        }
                        while (true) {
                        sum += Integer.parseInt(String.valueOf(mobileNo.charAt(j++)));
                            if (sum % 2 != firstDigitEvenOrOdd || j >= mobileNo.length()) {
                                numbiarNo.append(sum);
                                i = j - 1;
                                break;
                            }
                        }
                }
                return Integer.parseInt(numbiarNo.toString());
        }
}
```

7) User ID Generation

```
package com.w3epic.wiprotraining;
import java.io.*;
import  java.util.*;
class UserIDGeneration {
        public String userIdGeneration(String input1,String input2,int input3,int input4){
            String firstName = input1;
            String lastName = input2;
            int pin = input3;
            int N = input4;
            String longerName;
            String smallerName;
            StringBuilder userId = new StringBuilder();
            if (firstName.length() > lastName.length()) {
                longerName = firstName;
                smallerName = lastName;
            } else if (firstName.length() < lastName.length())  {
                longerName = lastName;
                smallerName = firstName;
            } else {
                if (firstName.compareTo(lastName) < 1 ) {
                    longerName = lastName;
                  smallerName = firstName;
                } else {
                    longerName = firstName;
                  smallerName = lastName;
                }
            }
            userId.append(smallerName.charAt(smallerName.length() - 1));
            userId.append(longerName);
            for (int i = 0; i < userId.length(); i++) {
                if (Character.isUpperCase(userId.charAt(i)))
                    userId.setCharAt(i, Character.toLowerCase(userId.charAt(i)));
                else
                    userId.setCharAt(i, Character.toUpperCase(userId.charAt(i)));
            }
            userId.append(String.valueOf(pin).charAt(N - 1));
            userId.append(String.valueOf(pin).charAt(String.valueOf(pin).length() - N));
            return userId.toString();
        }
}
```

8) Message controlled Robot movement

```
package com.w3epic.wiprotraining;
import java.io.*;
import  java.util.*;
class MessageControlledRobotMovement {
    public String moveRobot(int input1,int input2,String input3,String input4){
        int X = input1;
        int Y = input2;
        String currentPos = input3;
        String msg = input4;
        int currX = Integer.parseInt(currentPos.split("-")[0]);
        int currY = Integer.parseInt(currentPos.split("-")[1]);
        String currD = currentPos.split("-")[2]; // E/W/N/S
        String[] instructions = msg.split(" "); // M L R M M L M ...
        StringBuilder output = new StringBuilder();
        System.out.println(Arrays.toString(instructions));
        System.out.println("Curr: " + currX + currY + currD);
        for (int i = 0; i < instructions.length; i++) {
            System.out.print(instructions[i] + ":: ");
            if (instructions[i].equals("M")) {
                if (currD.equals("E") && (currX + 1 > X )) {
                    output.append("-ER");
                    break;
                }
                if (currD.equals("W") && (currX - 1 < 0 )) {
                    output.append("-ER");
                    break;
                }
                if (currD.equals("N") && (currY + 1 > Y )) {
                    output.append("-ER");
                    break;
                }
                if (currD.equals("S") && (currY - 1 < 0 )) {
                    output.append("-ER");
                    break;
                }
                if (currD.equals("E")) currX++;
                else if (currD.equals("W")) currX--;
                else if (currD.equals("N")) currY++;
                else if (currD.equals("S")) currY--;
            } else {
                if (currD.equals("E") && instructions[i].equals("L"))
                   currD = "N";
                else if (currD.equals("E") && instructions[i].equals("R"))
                    currD = "S";
                else if (currD.equals("W") && instructions[i].equals("L"))
                    currD = "S";
                else if (currD.equals("W") && instructions[i].equals("R"))
                    currD = "N";
                else if (currD.equals("N") && instructions[i].equals("L"))
                    currD = "W";
                else if (currD.equals("N") && instructions[i].equals("R"))
                    currD = "E";
                else if (currD.equals("S") && instructions[i].equals("L"))
                    currD = "E";
                else if (currD.equals("S") && instructions[i].equals("R"))
                    currD = "W";
            }
            output.delete(0, output.length());
            output.append(currX + "-" + currY + "-" + currD);
        }
        return output.toString();
    }
}
```