Contents

1 homeworks

1.1 Print Prime Numbers

```java
/*
Write a program to input an integer N and print all
    prime numbers less than N.
Please print 10 prime numbers in each line.
*/

import java.util.Scanner;
import java.util.Vector;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();

        // 建質數表
        // java 中沒有全域變數
            所以這邊建表目前是每次輸入都重建一次
            但不夠漂亮
        boolean[] prime = new boolean[10000000+5];

        // 用 vector 存質數們
        Vector<Integer> v = new Vector<Integer>();
        for( int i = 2; i <= n; i++ ){
            if(!prime[i]){
                v.add(i);
                for( int j = i * i; j <= n; j += i ){
                    prime[j] = true;
                }
            }
        }

        for( int i = 0, k = 0; i < v.size(); i++, k++
            ){

            if( k == 9 || i == v.size() - 1){
                System.out.println(v.get(i));
                // 把 k 歸回去 -1 是因為下次輪迴會先
                    k++
                k = -1;
            }
            else{
                System.out.print(v.get(i) + " ");
            }
        }
        v.clear();
    }
}
```

1.2 P1 Infix Notation to Postfix Notation

```java
/*
為了要便於中序轉後序的程式運行，請先將算式的數字與運算符號做分
1. 使用String類別 & 字串輸入
2. 使用StringTokenizer類別 或
    String類別之split()method來切割中序運算式。
3. 使用String [] array 儲存所有tokens。
4. 將數字token轉換成實數，加總後輸出總和。Hint: Use
    BigDecimal in java.math package

輸入
每組測資會給予不一定長度的算式，符號包含+,-,*,/,(,),%，算式中數

輸出
第一行輸出運算式的每個運算元，運算元間以空格字元隔開。
第二行輸出運算式的每個運算子，運算子間以空格字元隔開。
第三行印出運算元的總和，小事點以下印三位。
*/

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String str = input.nextLine();
        String token = "+-*/()%";

        // 當字串中帶有 + * | \ 等符號時要寫 \\
            轉義，因為他們在正則表達示中有相應的不同意義
        // limit is 0; array contains all substrings
        // Positive Lookahead or Lookbehind 觀念
        // “((?=@)|(?<=@))” -> 才會把 運算子 還有
            運算元 都單獨切開
        String[] array =
            str.split("((?=\\+|-|\\*|/|\\)|\\(|%)|(?<=\\+|-|\\*

        // 預設 ans 賦值 可控制小數點位數 ex 0.000 ->
            小數點後三位
        BigDecimal ans = new BigDecimal("0.000");

        boolean flag = true;
        for( String check : array){
            // string.contains
                一次查詢多個關鍵字的寫法
                要另外存一個數組 token
            // 且在查詢時要以此 for ( String check :
                array ) 去做掃描
            if( !token.contains(check) ){
                // 將字串型態的數字轉成實數並相加
                BigDecimal b = new BigDecimal(check);
                // (boolean 判斷式) ? (true的輸出) :
                    (false 的輸出)
                System.out.print( flag ? check : " "
                    + check);
                flag = false;

                // BigDecimal 內建 .add 可以直接相加
                ans = ans.add(b);
            }
        }
        System.out.println();

        flag = true;
        for( String again : array){
            if( token.contains(again) ){
                System.out.print( flag ? again : " "
                    + again);
                flag = false;
            }
        }
```

```
        System.out.println();

        // BigDecimal 四捨五入的寫法 .setScale(想要留取的位數, RoundingMode.HALF_UP)
        System.out.println(ans.setScale(3, RoundingMode.HALF_UP));
    }
}
```

1.3 P2 Infix Notation to Postfix Notation

```
/*
請利用 P1 中序轉後序-分割的答案輸出後續的算式，並輸出後序運算式。

程式要求
1.必須使用java.util.Stack 實作，否則不予計分。
2.請利用part1求得的token 字串陣列轉後序

輸入
每組測資會給予不一定長度的算式，符號包含+,-,*,/,(,),%，算式[illegible]

輸出
將後序算式的每個運算元與運算子後加入一個空格字元印出。
格式請詳看sample output
*/
import java.util.Scanner;
import java.util.Stack;

public class Main {

    public static int icpfunc(String check){
        // icp 是該 token 的順序
        // icp -> +- 1 -> */% 2 -> ( 4
        int icp;
        if(check.equals("(")){
            icp = 4;
            return icp;
        }
        else if(check.equals("*") || check.equals("/") || check.equals("%") ){
            icp = 2;
            return icp;
        }
        else{
            icp = 1;
            return icp;
        }
    }
    public static int ispfunc(String check){
        // isp 是堆疊最上層 stack.peek 中的順序
        // isp -> ( 0 -> +- 1 -> */% 2
        int isp;
        if(check.equals("(")){
            isp = 0;
            return isp;
        }
        else if(check.equals("*") || check.equals("/") || check.equals("%") ){
            isp = 2;
            return isp;
        }
        else{
            isp = 1;
            return isp;
        }
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String str = input.nextLine();
        String token = "+-*/(%";

        // 用 split 切割字串
        String[] array = str.split("((?=\\+|-|\\*|/|\\)|\\(|%)|(?<=\\+|-|\\*
        // 使用 stack 儲存運算子
        Stack<String> stack = new Stack<>();
        int icp = 0, isp = 0;

        for (String s : array) {
            // 如果是 ) 則輸出所有 ( 以前的 stack 內的運算子
            // 但 () 都不輸出
            if (s.equals(")")) {
                while (!stack.peek().equals("(")) {
                    System.out.print(stack.peek() + " ");
                    stack.pop();
                }
                // 把 ) 也 pop 掉
                stack.pop();
            }
            // 如果是 + - * / %
            else if (token.contains(s)) {
                // 運算此運算子的 icp 和 當前 stack 頂端的運算子的 isp
                icp = icpfunc(s);
                if (stack.empty()) {
                    stack.push(s);
                }
                else {
                    isp = ispfunc(stack.peek());
                    if (icp > isp) {
                        stack.push(s);
                    } else {
                        while (icp <= ispfunc(stack.peek())) {
                            System.out.print(stack.peek() + " ");
                            stack.pop();
                            if (stack.empty()) {
                                break;
                            }
                        }
                        stack.push(s);
                    }
                }
            }
            // 如果是運算元則直接輸出
            else {
                System.out.print(s + " ");
            }
        }
        // 最後因為有可能 array 在切割的時候最後一個值是運算元
        // 那樣就還會有運算子在 stack 還沒輸出
        // 所以這裡在確認一次
        while(!stack.empty()){
            System.out.print(stack.peek() + " ");
            stack.pop();
        }
        System.out.println();
    }
}
```

1.4 P3 Infix Notation to Postfix Notation

```
/*
請利用作業一(Part 2)中序轉後序-轉後序的答案輸出後續的算式，並輸出運算結果

輸入
每組測資會給予不一定長度的算式，符號包含+,-,*,/,(,),%。

程式要求 (未依規定以0分記)
```

```
Write a Class XXXX (name it yourself) (which is not the Main class)  for computing the input expression.
Class XXXX must include the following three static methods
  String[] AAAA(String s) which returns an array of string tokens for the input expession s.
String[] BBBB(String s) which returns an array of string tokens (Postfixe Expression) for the input expression s. Note that BBBB() can call AAAA().
BigDecimal CCCC(String s) which returns the computing result of input expresdion s. Note that CCCC() can call AAAA() and BBBB().
Please give a meaningful name for AAAA, BBBB, and CCCC.

輸出
輸出算式的答案(BigDecimal)
*/

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Scanner;
import java.util.Stack;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String str = input.nextLine();

        BigDecimal output;
        // 呼叫並使用另一個叫做 computing 的 class 的內部 method
        output = computing.caculating(str);

        System.out.println(output.setScale(2, RoundingMode.HALF_UP));
    }
}
// 同一份 java 檔案內只能有一個 public class 所以第二個直接叫 class 就好
class computing {

    // 用來判斷後續先後順序的兩個 method
    public static int icpfunc(String check){
        // icp 是該 token 的順序
        // icp -> +- 1 -> */% 2 -> ( 4
        int icp;
        if(check.equals("(")){
            icp = 4;
            return icp;
        }
        else if(check.equals("*") || check.equals("/") || check.equals("%") ){
            icp = 2;
            return icp;
        }
        else{
            icp = 1;
            return icp;
        }
    }
    public static int ispfunc(String check){
        // isp 是堆疊最上層 stack.peek 中的順序
        // isp -> ( 0 -> +- 1 -> */% 2
        int isp;
        if(check.equals("(")){
            isp = 0;
            return isp;
        }
        else if(check.equals("*") || check.equals("/") || check.equals("%") ){
            isp = 2;
            return isp;
        }
        else{
            isp = 1;
            return isp;
        }
    }

    // 等同於老師要求的第一個 method AAAA
    // 把輸入的 s 做切割後放入陣列 arr 中
    public static String[] splitmethod (String s){
        // 用 split 切割字串
        String[] arr =
            s.split("((?=\\+|-|\\*|/|\\)|\\(|%)|(?<=\\+|-|\\*|/
        return arr;
    }

    // 等同於老師要求的 BBBB
    // 呼叫 splitmethod method
        去切割字串並按照後續排序
    public static String[] postfixe (String s){
        String[] postfixearr;
        String token = "+-*/(%";

        // 呼叫 splitmethod method 切割字串
        // 用 postfixearr 接回傳的陣列裡
        postfixearr = computing.splitmethod(s);

        String[] ans = new String[postfixearr.length];

        // 使用 stack 儲存運算子
        Stack<String> stack = new Stack<>();
        int icp = 0, isp = 0, i = 0;

        for (String str : postfixearr) {
            // 如果是 ) 則輸出所有 ( 以前的 stack 內的運算子
            // 但 () 都不輸出
            if (str.equals(")")) {
                while (!stack.peek().equals("(")) {
                    ans[i++] = stack.peek();
                    stack.pop();
                }
                // 把 ) 也 pop 掉
                stack.pop();
            }
            // 如果是 + - * / %
            else if (token.contains(str)) {
                // 運算此運算子的 icp 和 當前 stack 頂端的運算子的 isp
                icp = icpfunc(str);
                if (stack.empty()) {
                    stack.push(str);
                }
                else {
                    isp = ispfunc(stack.peek());
                    if (icp > isp) {
                        stack.push(str);
                    } else {
                        while (icp <=
                            ispfunc(stack.peek())) {
                            ans[i++] = stack.peek();
                            stack.pop();
                            if (stack.empty()) {
                                break;
                            }
                        }
                        stack.push(str);
                    }
                }
            }
            // 如果是運算元則直接輸出
            else {
                ans[i++] = str;
            }
        }
```

```
        // 最後因為有可能 array
            在切割的時候最後一個值是運算元
        // 那樣就還會有運算子在 stack 還沒輸出
            所以這裡在確認一次
        while(!stack.empty()){
            ans[i++] = stack.peek();
            stack.pop();
        }
        return ans;
    }

    // 相當於老師要求的 CCCC
    public static BigDecimal caculating (String s){
        String answer = "";
        String[] frompostfixe;
        Stack<String> stack = new Stack<>();
        String token = "+-*/%";

        // 呼叫 postfixe method 去排序後續
        // 而 postfixe method 會先去呼叫 splitmethod
            method 去切割字串
        // 所以 caculating 可以 call postfixe 和
            splitmethod
        frompostfixe = postfixe(s);

        for (String str : frompostfixe) {

            // 如果跑道 frompostfixe
                沒放後續字串的地方了 就 break 掉
            if( str == null ){
                break;
            }

            // 當碰到運算子時 先到 stack
                去找最上面的兩個運算元
            // 分別讀取後 將兩者皆 pop 出來 (
                因為如此一來他們用過了 )
            // 對他們進行該運算子的運算
            // 最後將目前的運算結果 push 進 stack 裡
            //
                這樣就算之後有更要先乘除後加減的數也不會影響到之前運算過的數
            if (token.contains(str)) {

                BigDecimal top = BigDecimal.valueOf(
                    Double.parseDouble(stack.peek())
                    );
                stack.pop();
                BigDecimal sec = BigDecimal.valueOf(
                    Double.parseDouble(stack.peek())
                    );
                stack.pop();

                if (str.charAt(0) == '+'){
                    sec = sec.add( top );
                    stack.push(sec.toString());
                }
                else if (str.charAt(0) == '-'){
                    sec = sec.subtract( top );
                    stack.push(sec.toString());
                }
                else if (str.charAt(0) == '*'){
                    sec = sec.multiply( top );
                    stack.push(sec.toString());
                }
                else if (str.charAt(0) == '/'){
                    sec = sec.divide( top, 2,
                        RoundingMode.HALF_UP );
                    stack.push(sec.toString());
                }
                else if (str.charAt(0) == '%'){
                    sec = sec.remainder( top );
                    stack.push(sec.toString());
                }
                // 判斷目前運算的結果是多少 (
                    無論是暫時的 還是最終結果
                    answer = stack.peek();
                }
                // 如果是運算子就直接 push 進 stack
                else{
                    stack.push(str);
                }
            }
            // 最後回傳 Bigdecimal 型態的 answer
            // 根據老師要求的 method 型別
            return BigDecimal.valueOf(
                Double.parseDouble(answer) );
        }
    }
```

2 note

2.1 array

```
#define inf 10000


// Array 自動排序
Arrays.sort(arr);

// System.arraycopy(來源, 起始索引, 目的, 起始索引,
    複製長度)
System.arraycopy(arr1, 0, arr2, 0, arr1.length);
```

2.2 BigDecimal

```
// 利用 BigDecimal 去取小數點並四捨五入
            BigDecimal ans = new BigDecimal(a);
            System.out.println( ans.setScale(3,
                RoundingMode.HALF_UP));
```

2.3 date

```
/**
java 中獲取當前日期和時間方法
使用 Date 和 toString 去打印當前日期和時間
*/

import java.util.Date;

public class DateDemo {
   public static void main(String[] args) {
       // 初始化 Date 对象
       Date date = new Date();

       // 使用 toString() 函数显示日期时间
       System.out.println(date.toString());
   }
}

// Output:
// Mon May 04 09:51:52 CDT 2013

/*
日期比较
Java使用以下三种方法来比较两个日期：

1. 使用 getTime()
    方法获取两个日期（自1970年1月1日经历的毫秒数值），然后比较这
2. 使用方法 before()，after() 和 equals()
    例如，一个月的12号比18号早，则 new Date(99, 2,
        12).before(new Date (99, 2, 18)) 返回true。
3. 使用 compareTo() 方法，它是由 Comparable
    接口定义的，Date 类实现了这个接口。
```

```java

/**
使用 SimpleDateFormat 格式化日期
SimpleDateFormat
    是一个以语言环境敏感的方式来格式化和分析日期的类
SimpleDateFormat
    允许你选择任何用户自定义日期时间格式来运行
例如：
*/

import  java.util.*;
import java.text.*;

public class DateDemo {
   public static void main(String[] args) {

        Date dNow = new Date( );
        SimpleDateFormat ft = new SimpleDateFormat
            ("yyyy-MM-dd hh:mm:ss");

        System.out.println("當前時間為: " +
            ft.format(dNow));
    }
}

// Output:
// 當前時間為: 2018-09-06 10:16:34

/*
實際範例
*/

import java.util.Date;

public class DateDemo {

  public static void main(String[] args) {
    // 初始化 Date 对象
    Date date = new Date();

    // c的使用
    System.out.printf("全部日期和時間：%tc%n",date);

    //f的使用
    System.out.printf("年-月-日格式：%tF%n",date);

    //d的使用
    System.out.printf("月/日/年格式：%tD%n",date);

    //r的使用
    System.out.printf("HH:MM:SS
        PM格式（12時制）：%tr%n",date);

    //t的使用
    System.out.printf("HH:MM:SS格式（24時制）：%tT%n",date

    //R的使用
    System.out.printf("HH:MM格式（24時制）：%tR",date);
  }
}
```

2.4 Math

```java
/* 三角函數 */
sin(double a)
cos(double a)
tan(double a)
Math.PI

/*四捨五入*/

// 無條件向上取整數
Math.ceil(2.1) = 3.0
// 無條件向下取整數
Math.floor(2.1) = 2.0
// 取最接近整數 如果一樣近就往偶數的那個取
Math.rint(2.5) = 2.0
Math.rint(-2.5) = -2.0

/* 取亂數 */
Math.random()
```

2.5 pascal triangle

```java
import java.math.BigInteger;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();

        BigInteger[] triangle = new BigInteger[n+5];
        Arrays.fill(triangle, BigInteger.ONE);

        for( int i = 0; i <= n; i++ ){
            for( int j = i; j >0; j-- ){
                triangle[j] =
                    triangle[j].add(triangle[j-1]);
            }
            for( int j = n-i; j > 0; j-- ){
                System.out.print(" ");
            }
            for( int j = 0; j <= i; j++ ){
                System.out.printf(" %d", triangle[j]);
                if( i % 2 == 0 && i != 0 && j == i ){
                    break;
                }
            }
            if( i != 0 ){
                System.out.print(" 1");
            }
            System.out.println();
        }
    }
}
```

2.6 point2D

```java
import java.awt.geom.Point2D;

public class Main {

    public static void main(String[] args) {
        double x1 = 2.0, y1 = 6.0;
        Point2D pt = new Point2D.Double(x1, y1);

        System.out.println(pt.distance(0,0));
    }
}

// Output:
// 6.324555320336759
```

2.7 split stringTokenizer

```java
/*
當字串中帶有 `+ * | \` 等符號時要寫 `\\`
    轉義，因為他們在正則表達示中有相應的不同意義
最後放 0 的原因 -> limit is 0; array contains all
    substrings
Positive Lookahead or Lookbehind 觀念
```

```
“((?=@)|(?<=@))” -> 才會把 運算子 還有 運算元 都單獨切開
*/
String[] array = str.split("((?=\\+|-|\\*|/|\\)|\\(|%)|(?<=\\+|-|\\*|/|\\)|\\(|%))",0);

/*
預設 ans 賦值 可控制小數點位數 ex 0.000 -> 小數點後三位
*/
BigDecimal ans = new BigDecimal("0.000");

/*
取運算子的部分用 StringTokenizer 更為方便
*/
StringTokenizer st = new StringTokenizer(str,"0123456789.");
boolean flag = true;
while(st.hasMoreTokens()){
    System.out.print( flag ? st.nextToken() : " " + st.nextToken());
    flag = false;
}
```

2.8 str contain

```
boolean flag = true;
for( String check : array){

    // string.contains 一次查詢多個關鍵字的寫法 要另外存一個數組 token
    // 且在查詢時要以此 for ( String check : array ) 去做掃描
    if( !token.contains(check) ){

        // 將字串型態的數字轉成實數並相加
        BigDecimal b = new BigDecimal(check);

        // (boolean 判斷式) ? (true的輸出) : (false 的輸出)
        System.out.print( flag ? check : " " + check);
        flag = false;

        // BigDecimal 內建 .add 可以直接相加
        ans = ans.add(b);
    }
}

/*
找 BC 是 ABCD 的子字串
可以用 String.contains 找
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        String str = input.nextLine();

        // 已確認 : 切割後的空白不會存進 array
        String[] array = str.split(" ");

        // 原來可以用 str.contains 去找子字串 我還以為只能找字元..
        if(array[0].contains(array[1])){
            System.out.println(array[1] + " is a substring of " + array[0]);
        }
        else{
            System.out.println(array[1] + " is not a substring of " + array[0]);
        }

    }

}
```

2.9 Stringbuilder

```
// 先把 i 轉成 String ( 因為這樣比較方便 reverse )
String str = String.valueOf(i);

// 使用 StringBuilder 宣告 rev -> 因為 StringBuilder 才有 .reverse() 可以用
StringBuilder rev = new StringBuilder();

// 這裡 append (附加) str 的值給 rev
rev.append(str);

// reverse
rev.reverse();

/* 置換 */

// replace
import java.lang.*;

public class StringBuilderDemo {

  public static void main(String[] args) {

    StringBuilder str = new StringBuilder("Java Util Package");
    System.out.println("string = " + str);

    // 從 index 5 到 9 換掉
    str.replace(5, 9, "Lang");

    System.out.println("After replacing: " + str);
  }
}

// string = Java Util Package
// After replacing: Java Lang Package
```

2.10 this

```
/**
你看到在建構式參數與物件資料成員同名時，可用this加以區別
*/

public class CashCard {

    private String number;
    private int balance;
    private int bonus;

    public CashCard(String number, int balance, int bonus) {
        this.number = number;
        // 參數number指定給這個物件的number

        this.balance = balance;
        // 參數balance指定給這個物件的balance

        this.bonus = bonus;
        // 參數bonus指定給這個物件的bonus
    }
    ...
}
```

2.11 type changing

```java
/*
變數型態轉換
*/

// Integer to String
int num;
Integer.toStriing(num);
String.valueOf(num);

// String to Integer
String str;
Integer.parseInt(str);
Integer.valueOf(str);

// String to Array
// but will store like [1,2,3]
char[] ch = str.toCharArray();

// Integer to Double
int num;
double d1 = double.valueOf(num);
double d2 = new double(num);

// Double to
```