Report
Assignment 5 : Postfix Calculator

จัดทำโดย

นายสรธร แก้วโชติช่วงกูล 63070501067 CPE REGULAR

รายงานนี้เป็นส่วนหนึ่งของวิชา CPE111

Programming With Data Structures

King Mongkut's University of Technology Thonburi

- **สิ่งที่ทำใน Assignment**

1. สร้าง class ในการทำงานเกี่ยวกับ calculator ได้แก่ method ในการแปลงจาก Infix -> postfix และ method ในการคำนวณค่าจาก postfix Arraylist
2. สร้าง class ในการทำงานเกี่ยวกับ String โดยเฉพาะและสร้าง method ได้แก่
    - method ที่ใช้ addspace หน้าหลัง operator&function ต่างๆ
    - method check state ว่าสามารถคำนวณได้หรือไม่
    - method check group ว่า String อยู่กลุ่มอะไร
    - method เช็คตัวเลข ค่าคงที่ และฟังก์ชั่น
    - method แปลง – แบบ unary เป็น !

- Source code

- Class Calculator(main)

➢ Change_infix_to_postfix (เปลี่ยน String infix -> postfix)

```java
import java.util.ArrayList;

public class Calculator {
    static ArrayList <String> postfix = new ArrayList<String>(); // จองตัวแปรที่จะเก็บ Postfix
    static Stack <Double> numstack = new Stack<Double>(); // จองตัวแปรที่จะเก็บเฉพาะตัวเลขในตอนคำนวณ
    static Stack <String> oprstack = new Stack<String>(); // จองตัวแปรเก็บเครื่องหมายตอนคำนวณ
    static double ans =0; // สร้างตัวแปร ANS ไว้เก็บคำตอบ
    public static void Change_infix_to_postfix(String [] token) { // method เปลี่ยน Infix -> postfix
        int group,cur,prior=0,i;
        String buff;
        postfix.clear(); // เคลียร์พวกข้อมูลที่อยู่ใน postfix
        oprstack.clear(); // เคลียร์พวกข้อมูลที่อยู่ใน operater stack
        for(i=0;i<token.length;i++) { // วนรอบเพื่ออ่านค่าของ token ที่ split มา
            group = TokenAnalysis.checkgroup(token[i]); // หากลุ่มของ token ตัวนั้น
            if(group == 1) { // เป็นเลข
                postfix.add(token[i]); //ให้เพิ่มเข้าไปใน Array list เลย
            }
            else if(group >= 2 && group <= 6) {// เป็น function and operater
                do {
                    buff = oprstack.peek(); //ให้ดูเครื่องหมายของตัวล่าสุดใน operater stack ก่อน
                    prior = TokenAnalysis.checkgroup(buff); // เช็คความสำคัญของเครื่องหมายที่ peek ออกมาดู
                    cur = TokenAnalysis.checkgroup(token[i]); // เช็คความสำคัญของเครื่องหมายที่เรากำลังจะเพิ่ม
                    if(prior >= cur && prior <= 6) { // ถ้าตัวเรา peek ออกมาดูมันสำคัญกว่าตัวที่กำลังจะเพิ่มเข้าไป
                        buff = oprstack.pop(); // ให้ pop ออกมาจากตัวที่เก็บเครื่องหมาย
                        postfix.add(buff); // แล้วเพิ่มมันเข้าไปใน Postfix Arraylist
                    }
                }while(prior >= cur && prior <= 6); // ทำไปเรื่อยๆถ้าเครื่องหมายใน operator stack ยังสำคัญกว่าตัวที่จะเพิ่มเข้ามา
                oprstack.push(token[i]); // ทำเสร็จแล้วจึงค่อยเพิ่มตัวล่าสุดเข้าไปใน Operator stack
            }
            else if(group == 7) // ถ้าเป็น "("
                oprstack.push(token[i]); // ให้เพิ่มเข้าไปใน operator stack เลย
            else if(group == 8) { // ถ้าเป็น ")"
                do {
                  buff = oprstack.pop(); // ดึงข้อมูลใน Operator stack ออกมา
                  if(!buff.equals("(")) // ถ้าไม่ใช่ "("
                      postfix.add(buff); // เพิ่มข้อมูลตัวนั้นลงใน Postfix เลย (ก็คือทำไปจนกว่าจะเจอคู่นั้นก็คือวงเล็บเปิด)
               }while(!buff.equals("(")); // ถ้าเจอวงเล็บเปิดแล้วก็หยุดทำ
           }
           else // ถ้าไม่เข้าในกรณีด้านบนเลย
               System.out.println("answer > Postfix Error!.");  // บอก Postfix Error!
       }
    }
```

➢ Convert String(แปลง String เป็น number)

```java
    public double Convert_String(String token) { // method ในการแปลง String
        if(token.equalsIgnoreCase("E")) // ถ้าเจอ E
            return Math.E; // ให้ค่า E กลับ
        else if(token.equalsIgnoreCase("PI")) // ถ้าเจอ pi
            return Math.PI; // ให้ค่า pi กลับ
        else if(token.equalsIgnoreCase("ans")) // ถ้าเจอ ans
            return ans; // ให้ค่า ans กลับ
        else
            return Double.parseDouble(token); // ถ้าไม่ตรงกลับด้านบนให้แปลงค่ามันเป็นตัวเลขจำนวนจริง
    }
```

➢ Calculate (คำนวณค่า)

```
55•    public double Calculate() { // method คำนวณค่าของเครื่องคิดเลข
56         int i,group;
57         double num,num1,num2;
58         String token = new String("");
59         for(i=0;i<postfix.size();i++) { // วนรอบเพื่อเอาค่าที่เราแปลงจาก infix -> postfix แล้วมาคิดให้ครบทุกตัว
60             token = postfix.get(i); // ดึงข้อมูลจาก postfix ออกมาทีละตัว
61             group = TokenAnalysis.checkgroup(token); // แล้วเอามาเช็กกลุ่ม
62             if(group == 1) { // อยู่กลุ่ม 1
63                 num = Convert_String(token); // ไปแปลง String เป็นเลขก่อน
64                 numstack.push(num); // ใส่เข้าไปใน numstack
65             }
66             else if(group >= 2 && group <= 4) { // เป็นเครื่องหมาย
67                 num1 = numstack.pop(); // pop ข้อมูลออกมารอคำนวณ 2 ตัวเลย
68                 num2 = numstack.pop();
69                 if (group == 2 && token.equals("+")) { // เป็น +
70                     num = num1 + num2; // เอาข้อมูลที่ Pop ออกมามาบวก
71                     numstack.push(num); // ใส่เข้าไปใน numstack
72                 }
73                 else if (group == 2 && token.equals("-")) { // เป็น -
74                     num = num2 - num1;// เอาข้อมูลที่ Pop ออกมามา -
75                     numstack.push(num);// ใส่เข้าไปใน numstack
76                 }
77                 else if (group == 3 && token.equals("*")) { // เป็น *
78                     num = num2 * num1;// เอาข้อมูลที่ Pop ออกมามาคูณ
79                     numstack.push(num);// ใส่เข้าไปใน numstack
80                 }
81                 else if (group == 3 && token.equals("/")) { // เป็น "/"
82                     num = num2 / num1;// เอาข้อมูลที่ Pop ออกมามาหาร
83                     numstack.push(num);// ใส่เข้าไปใน numstack
84                 }
85                 else if (group == 4 ) { // เป็น ^
86                 num = Math.pow(num2,num1); // เอามายกกำลัง
87                 numstack.push(num); // ใส่เข้าไปใน numstack
88                 }
89             }
90             else if (group == 5) { // !
91                 num1 = numstack.pop(); // เอาข้อมูลออกมาตัวเดียวพอ
92                 num = -num1; // ใส่ลบเข้าไป
93                 numstack.push(num); // ใส่เข้าไปใน numstack
94             }
95             else if (group == 6) { // เป็นพวก Function
96                 num1 = numstack.pop(); // ดึงข้อมูลออกมาแค่ตัวเดียว
97                 if(token.equalsIgnoreCase("sin")) { // เป็น sin
98                     num = Math.sin(num1 * (Math.PI / 180)); // เอาข้อมูลที่ Pop ใส่ลงไปใน sin (ต้องทำเป็น Radian ด้วย)
99                     numstack.push(num); // คำนวณแล้วเพิ่มเข้าไปใน numstack
100                }
101                else if(token.equalsIgnoreCase("cos")) { // เป็น cos
102                    num = Math.cos(num1*(Math.PI / 180)); // เอาข้อมูลที่ Pop ใส่ลงไปใน sin (ต้องทำเป็น Radian ด้วย)
103                    numstack.push(num); // คำนวณแล้วเพิ่มเข้าไปใน numstack
104                }
105                else if (token.equalsIgnoreCase("tan")) { // เป็น tan
106                    num = Math.tan(num1*(Math.PI / 180)); // เอาข้อมูลที่ Pop ใส่ลงไปใน sin (ต้องทำเป็น Radian ด้วย)
107                    numstack.push(num); // คำนวณแล้วเพิ่มเข้าไปใน numstack
108                }
109                else if(token.equalsIgnoreCase("asin")) { // เป็น asin
110                    num = Math.asin(num1)*(180 /Math.PI); // เอาข้อมูลที่ Pop ใส่ลงไปใน sin (ต้องทำเป็น Degree ด้วย)
111                    numstack.push(num); // คำนวณแล้วเพิ่มเข้าไปใน numstack
112                }
```

```
                else if(token.equalsIgnoreCase("acos")) { // เป็น acos
                    num = Math.acos(num1)*(180 /Math.PI); // เอาข้อมูลที่ Pop ใส่ลงไปใน sin (ต้องทำเป็น Degree ด้วย)
                    numstack.push(num); // คำนวณแล้วเพิ่มเข้าไปใน numstack
                }
                else if(token.equalsIgnoreCase("atan")) { // เป็น atan
                    num = Math.atan(num1)*(180 /Math.PI); // เอาข้อมูลที่ Pop ใส่ลงไปใน sin (ต้องทำเป็น Degree ด้วย)
                    numstack.push(num); // คำนวณแล้วเพิ่มเข้าไปใน numstack
                }
                else if(token.equalsIgnoreCase("sqrt")) { // เป็น sqrt
                    num = Math.sqrt(num1); // เอาเลขที่ Pop ไว้ก่อนหน้าใส่เข้าไปในสแควร์รูท
                    numstack.push(num); // คำนวณแล้วเพิ่มเข้าไปใน numstack
                }
                else if(token.equalsIgnoreCase("exp")) { // เป็น exp
                    num = Math.exp(num1); // เอาเลขที่ Pop ไว้ก่อนหน้าใส่เข้าไปใน exponential
                    numstack.push(num); // คำนวณแล้วเพิ่มเข้าไปใน numstack
                }
                else if(token.equalsIgnoreCase("log")) { // เป็น log
                    num = Math.log10(num1); // เอาเลขที่ Pop ไว้ก่อนหน้าใส่เข้าไปใน Log ฐาน 10
                    numstack.push(num); // คำนวณแล้วเพิ่มเข้าไปใน numstack
                }
                else if(token.equalsIgnoreCase("abs")){ // เป็น abs
                    num = Math.abs(num1); // เอาเลขที่ Pop ไว้ก่อนหน้าใส่เข้าไปใน absolute
                    numstack.push(num); // คำนวณแล้วเพิ่มเข้าไปใน numstack
                }
            }
        }
        ans = numstack.pop(); // ทำครบทุกตัวจะได้คำตอบอยู่ใน Numstack แล้วก็ดึงออกมาใส่ Ans
        return ans; // ส่งค่ากลับ
    }
```

➢ Main (method ที่ใช้รัน)

```
    public static void main(String[] args) {
        Calculator cal = new Calculator();          // จอง instance cal เป็นตัวคำนวณ , in เป็น Scanner
        Scanner in = new Scanner(System.in);
        String str; // จอง String
        int state,i;
        do {
            System.out.printf("expression > ");
            str = in.nextLine(); // อ่าน String จนหมดบรรทัด
            if(str.equalsIgnoreCase("help")) // ถ้าเจอคำว่า help
                System.out.println("sin, cos, tan, asin, acos, atan, sqrt, log, exp, abs +, -, *, /, ^, (, ), pi, ans");
            else { // ถ้าไม่ใช่ help
                if(!str.equalsIgnoreCase("end")) { // และไม่ใช่ end
                    str = "(" + str + ")"; // ใส่วงเล็บเพิ่มหน้าหลังของ String
                    str = TokenAnalysis.Transfrom(str); // เติมช่องว่าง
                    //System.out.println(""+str+"");  แสดงข้อความหลังจาก Transform
                    String [] token = str.trim().split("\\s+"); // แบ่ง String ตามช่องว่าง
                    int count = token.length; // นับจำนวนที่ Spilt ได้
                    TokenAnalysis.change_sign_operater(token); // แปลง - แบบ unary ให้เป็น ! เพื่อง่ายต่อการแยกแยะ
                    state = TokenAnalysis.check_state(token); // ส่งไปเช็ค State ว่าสามารถคำนวณชุดเลขที่ผู้ใช้ป้อนเข้ามาได้หรือไม่
                    if(state != -1) { // ถ้าคำนวณได้
                        cal.Change_infix_to_postfix(token); // เปลี่ยนเป็น Postfix ก่อนคำนวณ
                        System.out.println("Postfix (Queue) = " + postfix); // แสดง  postfix หลังจากแปลงแล้ว
                        ans = cal.Calculate(); // คำนวณค่า
                        System.out.println("answer > " + ans);  // แสดงคำตอบ
                    }
                    else
                        System.out.print("answer > Syntax Error!\n"); // ถ้าไม่สามารถคำนวณได้ให้แสดงข้อความบอก
                }
            }
        }while(!str.equalsIgnoreCase("end")); // ทำไปเรื่อยๆถ้ายังไม่ได้รับคำว่า end
        System.out.print("End Program!."); // แสดงข้อความจบโปรแกรม
        System.out.print("This Program wrote by Sorathorn Kaewchotchuangkul 63070501067 CPEREGULAR.");
    }
```

- Class ที่ทำงานเกี่ยวกับ String

  - method เช็ค Number Constant และ Function

```
1
2  public class TokenAnalysis {
3      public static boolean isNumber(String token) { // method เช็คว่าเป็น number ไหม
4          boolean check = true;
5          try {
6              Double.parseDouble(token); // ให้ลองแปลงเป็น จำนวนจริง
7          }
8          catch(Exception e){ // ถ้าแปลงไม่ได้
9              check = false;
10         }
11         return check; // ส่งค่ากลับว่าแปลงได้ไหม TRUE FALSE
12     }
13     public static int isConstant(String token){ // method เป็นค่าคงที่หรือไม่
14         String [] myfunc = {"pi","e","ans"};
15         int i,ans=-1;
16         for(i=0;i < myfunc.length;i++) { // วนรอบเปรียบเทียบกับ String ใน Myfunc
17             if(myfunc[i].equalsIgnoreCase(token))
18                 ans = i;
19         }
20         return ans; // ถ้าตรงจะให้ค่าตำแหน่งกลับซึ่ง >= 0 แต่ถ้าไม่ก็จะให้ค่า -1
21     }
22     public static int isFuntion(String token) { // method เช็คว่าเป็น Function ไหม
23         int i,ans=-1;
24         String [] myfunc = {"sin","cos","tan","asin","acos","atan","sqrt","log","exp","abs"};
25         for(i=0;i<myfunc.length;i++) { // วนรอบเปรียบเทียบกับ String ใน Myfunc
26             if(myfunc[i].equalsIgnoreCase(token))
27                 ans = i;
28         }
29         return ans; // ถ้าตรงจะให้ค่าตำแหน่งกลับซึ่ง >= 0 แต่ถ้าไม่ก็จะให้ค่า -1
30     }
```

  - Method check group (เช็คกลุ่มของ String) และ Transfrom เพิ่มช่องว่าง

```
31     public static int checkgroup(String token) { // method เช็คกลุ่มของสตริง
32         if(isNumber(token)) // เป็นเลข
33             return 1;
34         else if(isConstant(token) >= 0) // เป็นค่าคงที่
35             return 1;
36         else if(token.matches("[+-]"))// เป็น + -
37             return 2;
38         else if(token.matches("[*/]"))// เป็นเลข * /
39             return 3;
40         else if(token.equals("^"))// เป็นยกกำลัง
41             return 4;
42         else if(token.equals("!"))// เป็นนิเสธ
43             return 5;
44         else if(isFuntion(token)>=0)// เป็น function
45             return 6;
46         else if(token.equals("("))// เป็นวงเล็บเปิด
47             return 7;
48         else if(token.equals(")"))// เป็นวงเล็บปิด
49             return 8;
50         else // ไม่ตรงกับด้านบนเลย
51             return 0;
52     }
53     public static String Transfrom(String token) { // method add space หน้าหลัง operator
54         token = token.replace("+"," + "); // ใช้ .replace โดยถ้ามีตัวที่เราต้องการให้แทนที่ใหม่มันก็จะแทนค่าที่เราให้แทนใหม่ลงไป
55         token = token.replace("-"," - ");
56         token = token.replace("*"," * ");
57         token = token.replace("/"," / ");
58         token = token.replace("^"," ^ ");
59         token = token.replace("("," ( ");
60         token = token.replace(")"," ) ");
61         return token;
62     }
```

➢ Method เปลี่ยนเครื่องหมาย – แบบ unary

```
63●     public static void change_sign_operater(String [] token) { // method เปลี่ยนเครื่องของ - แบบ unary ให้เป็น !
64          int i;
65          if(token[0].equals("-")) // ถ้ามันอยู่ตัวแรกเลย
66              token[0] = "!"; // เปลี่ยนเพราะเป็น unary แน่นอน
67          for(i=0;i<token.length - 1; i++) { // วนรอบเช็คไปจนตัวรองสุดท้าย
68              if(token[i+1].equals("-") && (token[i].matches("[+-/*^()]"))) { // ถ้าตัวถัดไปเป็น - แล้วตัวก่อนหน้าเป็นเครื่องหมายอยู่แล้ว
69                  token[i+1] = "!";  // แสดงว่า - ตัวนั้นเป็น Unary แน่นอน
70              }
71          }
72      }
```

➢ Method check State (เช็คว่าลำดับการคำนวณถูกต้องหรือไม่)

```
73●     public static int check_state(String[] token) { // method เช็ค state
74          int i,next=0,state = 0,pair=0;
75          for(i=0;i<token.length && state != -1;i++) { // วนรอบเช็คทุกตัวใน token
76              state = next; // ให้ state เท่ากับตัวที่ติดได้ก่อนหน้าในแต่ละรอบ
77              next = checkgroup(token[i]); // เช็คกลุ่มของ token ที่กำลังเช็ค
78              if(next == 0) // ถ้า next เป็น 0
79                  state = -1;
80              else {
81                  if(next >= 2 && next <= 4)
82                      next = 2; // รวมกรณีเครื่องหมายให้เป็นอันเดียวกันก่อนจะทำเพื่อความจะได้สะดวกขึ้น
83                  if(next == 7)
84                      pair++;       // นับเพื่อเช็คเผื่อวงเล็บปิดจะมาก่อนวงเล็บเปิด
85                  if(next == 8)
86                      pair--;
87                  if(pair < 0 )  // ถ้ามัน < 0 แสดงว่าวงเล็บปิดมาก่อน
88                      state = -1;
89                  else if(state == 0 && (next == 2 || next == 8)) // ถ้ามันขึ้นต้นด้วย Operator กับวงเล็บปิด
90                      state = -1;
91                  else if(state == 1 && (next == 1 || next == 6 || next == 7 || next == 5)) // ถ้าเป็นตัวเลขแล้วตัวถัดไปต้องเป็น Operator กับวงเล็บปิด
92                      state = -1;
93                  else if(state == 2 && (next == 2 || next == 8)) // ถ้าเป็น operator ตัวต่อไปต้องไม่ใช่ Operator กับวงเล็บปิด
94                      state = -1;
95                  else if(state == 5 && (next == 2 || next == 8)) // ถ้าเป็น ! ตัวต่อไปต้องไม่ใช่ Operator กับวงเล็บปิด
96                      state = -1;
97                  else if(state == 6 && (next != 7)) // ถ้าเป็น Function ตัวต่อไปต้องเป็นวงเล็บเปิด
98                      state = -1;
99                  else if(state == 7 && (next == 2 || next == 8)) // ถ้าเป็นวงเล็บเปิดตัวต่อไปต้องไม่ใช่ Operator กับวงเล็บปิด
100                     state = -1;
101                 }
102             }
103         if(pair != 0 || state == -1) // ถ้าวงเล็บเปิด ปิด ไม่ครบคู่หรือ state = -1 อันใดอันหนึ่ง
104             return -1; // ไม่สามารถคำนวณได้
105         else { // ถ้าไม่ใช่
106             if(next == 1 || next == 8) // เช็คว่าตัวท้ายต้องเป็นวงเล็บปิดเท่านั้นหรือไม่ก็ตัวเลข
107                 return 1; // คำนวณได้
108             else // ถ้าไม่ใช่
109                 return -1; // ไม่สามารถคำนวณได้
110             }
111
112     }
```

Note! อธิบายโค้ด comment อยู่ในตัวโค้ด

- Test case & อธิบาย

กรณีนี้ **Syntax Error !** เพราะ **State** ได้ **-1** เนื่องจากถ้าเจอ **Number** ตัวต่อไปต้องไม่ใช่ **Number** ไม่งั้นจะ **Error** และกรณี **sin Eror** เพราะตัวถัดไปไม่ใช่ " (" จึง **Error.**

```
expression >  1.23 4.56 +
answer > Syntax Error!
expression > sin 90
answer > Syntax Error!
```

```
expression >  0/1
Postfix (Queue) = [0, 1, /]
answer > 0.0
expression >  1/sin(0)
Postfix (Queue) = [1, 0, sin, /]
answer > Infinity
expression >  sin(0)/sin(0)
Postfix (Queue) = [0, sin, 0, sin, /]
answer > NaN
```

กรณีนี้ สามารถคำนวณได้เพราะ **State** ถูกต้องแต่ค่าที่ออกมาของ **1/0** คือ **infinity** และ **0/0** คือ **NaN** ซึ่งก็ต้องกับค่าทางคณิตศาสตร์

```
expression >  -2^-2-2
Postfix (Queue) = [2, !, 2, !, ^, 2, -]
answer > -1.75
expression >  -1+2^3/(4-5*6)+pi
Postfix (Queue) = [1, !, 2, 3, ^, 4, 5, 6, *, -, /, +, pi, +]
answer > 1.8339003458974854
expression >  sqrt(log(10^2)+exp(3))
Postfix (Queue) = [10, 2, ^, log, 3, exp, +, sqrt]
answer > 4.6995251806100224
expression > sqrt(3^2+4^2)
Postfix (Queue) = [3, 2, ^, 4, 2, ^, +, sqrt]
answer > 5.0
expression >  ans^-2
Postfix (Queue) = [ans, 2, !, ^]
answer > 0.04
expression >  asin(1-cos(0)+sin(90))
Postfix (Queue) = [1, 0, cos, -, 90, sin, +, asin]
answer > 90.0
expression > sin(30)^2+cos(30)^2
Postfix (Queue) = [30, sin, 2, ^, 30, cos, 2, ^, +]
answer > 1.0
expression > end
End Program!. This Program wrote by Sorathorn Kaewchotchuangkul 63070501067 CPEREGULAR.
```

ถ้าเจอข้อความ **End** ก็จบโปรแกรมและแสดงข้อความว่าจบโปรแกรมแล้ว

กรณีนี้ก็คำนวณได้ปกติตาม **algorithm** ที่เราเขียนไว้เลยไม่ได้เกิดปัญหาขึ้นเพราะ **State** นั้นก็ถูกต้องและการคำนวณก็เป็นค่าปกติและของ **asin** ที่ได้ออกมาก็คือมุม ซึ่งเป็น **Degree**

- **สรุปความเข้าใจของตนเอง**

ใน Assignment นี้จะคล้ายๆกับ Assignment ที่ 4 คือมีการทำงานงานเกี่ยวกับ String แต่ใน Assignment นี้จะเขียนเป็นภาษาจาวาซึ่งทำคำสั่งบางอย่างนั้นสั้นกว่าและใช้งานได้ง่ายกว่าภาษาซีมาก นอกจากนี้ยังมีการสร้าง method เพิ่มการทำงานเกี่ยวกับ Stack & Queue มาด้วยโดยการแปลง String จาก infix -> postfix และนำ postfix นั้นไปคำนวณค่าออกมา ตาม Alglorithm ที่เราสร้างไว้สำหรับข้อมูลที่เก็บใน Stack และทำการแสดงผลออกมา

- **ผลการประเมินตนเอง**

ให้ตนเองอยู่ที่ระดับ 80 เพราะสามารถทำงานได้ด้วยตัวเองแต่ก็ไม่ทั้งหมดยังมีบางจุดที่สงสัยและยังไม่เข้าใจจึงถามพี่ TA และถามเพื่อนๆบ้างบางจุดจึงเข้าใจและสามารถทำงานต่อได้จนงานเสร็จและสามารถส่งได้ทันเวลา