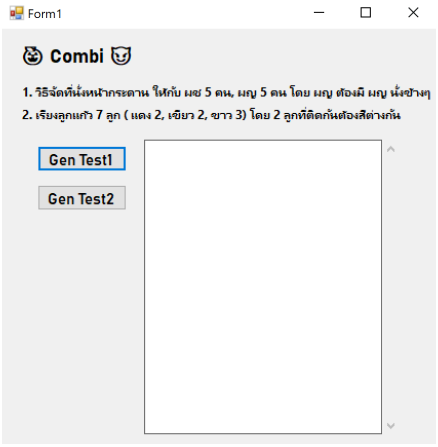
นายชยพล หวง 6330085821 (sec 02)

HW9 Thursday Combi

HW9 Thursday Combi - โปรแกรมแสดงจำนวนวิธีและรูปแบบทั้งหมดที่เป็นไปได้ของปัญหาการเรียงสับเปลี่ยน

รายงานฉบับนี้จะขอนำเสนอการเขียนโปรแกรมแก้โจทย์ปัญหาการเรียงสับเปลี่ยนที่น่าสนใจ 2 ข้อด้วยกัน



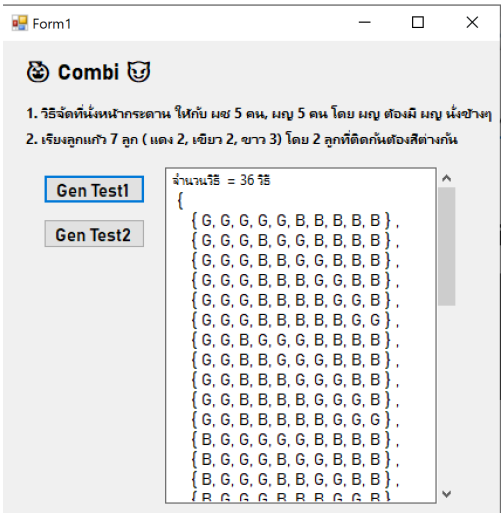
แนวคิด: หลักการในการวิธีเรียงสับเปลี่ยนในลักษณะเส้นตรง (หน้ากระดาน) ทั้งหมดคือการใส่ทุกๆตำแหน่งที่เป็นไปได้ ตามเงื่อนไขของโจทย์ ซึ่งสามารถนำแนวคิดและการเขียนโปรแกรมของฟังก์ชันเวียนเกิด หรือ Recursive Function มาช่วยในการคำนวณได้

โจทย์ข้อที่ 1:

หาวิธีในการจัดที่นั่งหน้ากระดานให้กับผู้ชาย 5 คน และผู้หญิง 5 คน โดยผู้หญิงทุกคนจะต้องมีผู้หญิงนั่งข้างๆ

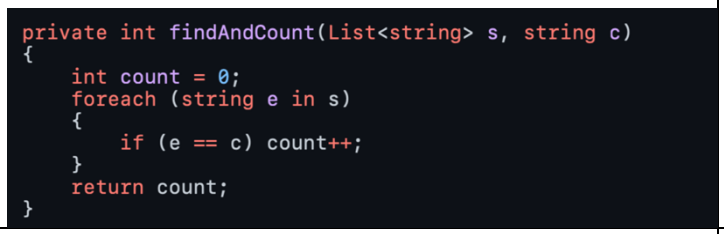
จำนวนวิธีทั้งหมด = 36 วิธี

รูปแบบทั้งหมดที่เป็นไปได้:



หลักการทำงานของโปรแกรม:

Method:

	- private int findAndCount(List<string> s, string c) นับและ return ว่าภายใน List<string> s มี string c ปรากฏอยู่กี่ตัว
	- private List<string> copy(List<string> from, List<string> to) จะทำการ copy ทุกๆ element ใน List<string> from มาเก็บไว้ใน List<string> to
	- สอง method นี้ จะทำการแสดงผลลัพธ์ผ่านทาง testBox โดยวน loop แสดงผลลัพธ์ทุกๆ รูปแบบที่เกิดขึ้น รวมถึงจำนวนวิธีทั้งหมดที่เป็นไปได้

```
41          private void sitting(List<string> s)
42          {
43              if (s.Count() == 10)
44              {
45                  test1Ans.Add(s);
46                  return;
47              }
48
49              if (s.Count() > 1 && s.ElementAt(s.Count() - 1) == "G" && s.ElementAt(s.Count() - 2) != "G")
50              {
51                  if (findAndCount(s, "G") < 5)
52                  {
53                      List<string> tmp = new List<string>();
54                      tmp = copy(s, tmp);
55                      tmp.Add("G");
56                      sitting(tmp);
57                  }
58              }
59              else if (s.Count() == 1 && s.ElementAt(s.Count() - 1) == "G")
60              {
61                  if (findAndCount(s, "G") < 5)
62                  {
63                      List<string> tmp = new List<string>();
64                      tmp = copy(s, tmp);
65                      tmp.Add("G");
66                      sitting(tmp);
67                  }
68              }
69
70              else if (s.Count() == 9 && s.ElementAt(s.Count() - 1) != "G")
71              {
72                  if (findAndCount(s, "B") < 5)
73                  {
74                      List<string> tmp = new List<string>();
75                      tmp = copy(s, tmp);
76                      tmp.Add("B");
77                      sitting(tmp);
78                  }
79              }
80              else
81              {
82                  if (findAndCount(s, "G") < 5)
83                  {
84                      List<string> tmp = new List<string>();
85                      tmp = copy(s,tmp);
86                      tmp.Add("G");
87                      sitting(tmp);
88                  }
89                  if (findAndCount(s, "B") < 5)
90                  {
91                      List<string> tmp = new List<string>();
92                      tmp = copy(s, tmp);
93                      tmp.Add("B");
94                      sitting(tmp);
95                  }
96              }
97          }
```

private void sitting(List<string> s) จะเป็น method หลักในการทำงานเพื่อหาจำนวนวิธีจัดที่นั่งโจทย์ข้อที่ 1

การทำงานจะเป็นรูปแบบ recursive โดยจะรับ parameter เป็น List<string> s จะเวียนเกิดทุกๆรูปแบบที่เป็นไปได้ โดย หากจำนวนผู้ที่มานั่งยังไม่ครบ 10 คน และผู้ชายและผู้หญิงที่มานั่งไม่ครบตามจำนวน ก็จะทยอยใส่ผู้ที่จะมานั่งใหม่ (จาก ชาย 5 คน หญิง 5 คน) โดยได้แยกการทำงานของโปรแกรมเป็นกรณีย่อย ๆเพิ่มเติมดังนี้

หากผู้ที่มานั่งคนแรกเป็นผู้หญิง คนถัดไปจะต้องเป็นผู้หญิงด้วย

หากคนรองสุดท้าย (คนที่9) เป็นผู้หญิง แต่คนที่ 8 ไม่ใช่ผู้หญิง คนสุดท้ายแล้วจะต้องเป็นผู้หญิง

เมื่อครบ 10 คน ตามเงื่อนไขแล้ว จะทำการ Add ไปยัง List<List<string>> test1Ans

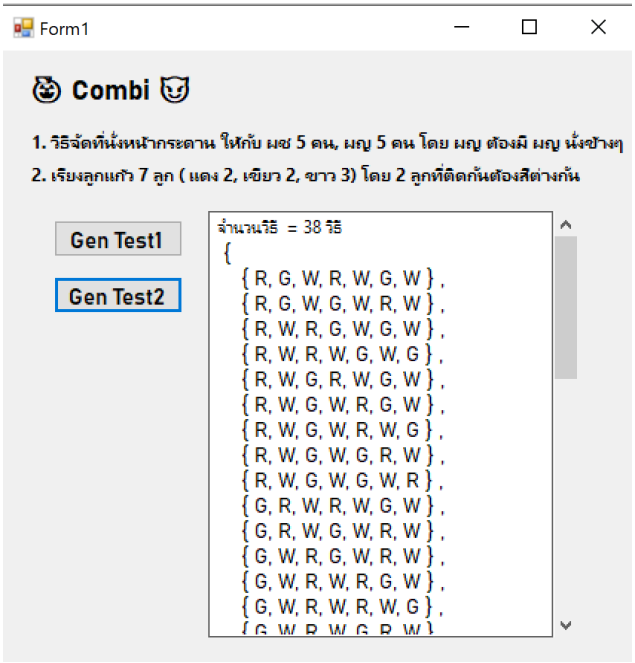
และจบการทำงานของ method นั้นๆ

โจทย์ข้อที่ 2:

เรียงลูกแก้ว 7 ลูก (ประกอบไปด้วย สีแดง 2 ลูก สีเขียว 2 ลูก และสีขาว 3 ลูก) โดยทุกๆ 2 ลูกที่อยู่ติดกันต้องมีสีแตกต่างกัน

จำนวนวิธีทั้งหมด 38 วิธี

รูปแบบทั้งหมดที่เป็นไปได้:



private void lookkaew(List<string> s) จะเป็น method หลักในการทำงานเพื่อหาจำนวนวิธีจัดที่นั่งโจทย์ข้อที่ 2 ซึ่งจะทำงานในลักษณะเดียวกับ private void sitting(List<string> s)

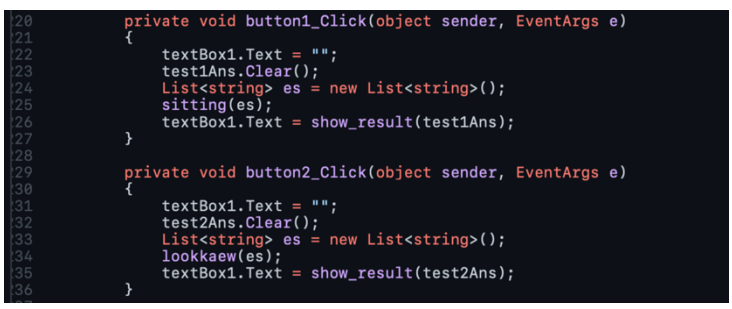
การทำงานจะเป็นรูปแบบ recursive โดยจะรับ parameter เป็น List<string> s จะเวียนเกิดทุกๆรูปแบบที่เป็นไปได้ โดย หากจำนวนผลลัพธ์ใน List<string> ยังไม่ครบ 7 ลูก และจำนวนลูกแก้วสีต่างๆที่ใส่ก็ยังไม่ครบตามจำนวน ก็จะทยอยใส่ผู้ลูกแก้วสีต่างๆ (จาก สีแดง 2 ลูก สีเขียว 2 ลูก และสีขาว 3 ลูก) โดยได้แยกการทำงานของโปรแกรมเป็นกรณีย่อย ๆเพิ่มเติมดังนี้

หากลูกแก้วลูกสุดท้ายของ List<string> มีสีหนึ่ง ก็จะใส่ลูกแก้วสีที่ไม่ซ้ำเข้าไป โดยทำงานเวียนเกิดไปจนครบ

เมื่อครบ 7 ลูก ตามเงื่อนไขแล้ว จะทำการ Add ไปยัง List<List<string>> test2Ans

และจบการทำงานของ method นั้นๆ

(สามารถดูการทำงานของโปรแกรม private void lookkaew(List<string> s) ได้ในหน้าถัดไป)



```
        private void button1_Click(object sender, EventArgs e)
        {
            textBox1.Text = "";
            test1Ans.Clear();
            List<string> es = new List<string>();
            sitting(es);
            textBox1.Text = show_result(test1Ans);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            textBox1.Text = "";
            test2Ans.Clear();
            List<string> es = new List<string>();
            lookkaew(es);
            textBox1.Text = show_result(test2Ans);
        }
```

- private void button1_Click(object sender, EventArgs e)
- private void button2_Click(object sender, EventArgs e)

จะทำการเรียก main method และแสดงผลลัพธ์ที่คำนวณได้จากข้อที่ 1 หรือ ข้อที่ 2


```
        private void lookkaew(List<string> s)
        {
            if (s.Count() == 7)
            {
                test2Ans.Add(s);
                return;
            }

            // R R , G G ,  W W W

            if (s.Count() > 0 && s.ElementAt(s.Count() - 1) == "R")
            {
                if (findAndCount(s, "G") < 2)
                {
                    List<string> tmp = new List<string>();
                    tmp = copy(s, tmp);
                    tmp.Add("G");
                    lookkaew(tmp);
                }
                if (findAndCount(s, "W") < 3)
                {
                    List<string> tmp = new List<string>();
                    tmp = copy(s, tmp);
                    tmp.Add("W");
                    lookkaew(tmp);
                }
            }
            else if(s.Count() > 0 && s.ElementAt(s.Count() - 1) == "G")
            {
                if (findAndCount(s, "R") < 2)
                {
                    List<string> tmp = new List<string>();
                    tmp = copy(s, tmp);
                    tmp.Add("R");
                    lookkaew(tmp);
                }
                if (findAndCount(s, "W") < 3)
                {
                    List<string> tmp = new List<string>();
                    tmp = copy(s, tmp);
                    tmp.Add("W");
                    lookkaew(tmp);
                }
            }

            else if (s.Count() > 0 && s.ElementAt(s.Count() - 1) == "W")
            {
                if (findAndCount(s, "R") < 2)
                {
                    List<string> tmp = new List<string>();
                    tmp = copy(s, tmp);
                    tmp.Add("R");
                    lookkaew(tmp);
                }
                if (findAndCount(s, "G") < 2)
                {
                    List<string> tmp = new List<string>();
                    tmp = copy(s, tmp);
                    tmp.Add("G");
                    lookkaew(tmp);
                }
            }

            else
            {
                if (findAndCount(s, "R") < 2)
                {
                    List<string> tmp = new List<string>();
                    tmp = copy(s, tmp);
                    tmp.Add("R");
                    lookkaew(tmp);
                }
                if (findAndCount(s, "G") < 2)
                {
                    List<string> tmp = new List<string>();
                    tmp = copy(s, tmp);
                    tmp.Add("G");
                    lookkaew(tmp);
                }
                if (findAndCount(s, "W") < 3)
                {
                    List<string> tmp = new List<string>();
                    tmp = copy(s, tmp);
                    tmp.Add("W");
                    lookkaew(tmp);
                }
            }
```