อ-นามสกุล_____รหัส_____ห้อง_____

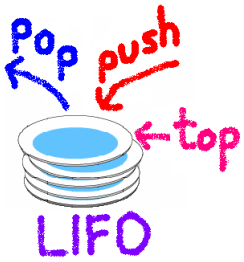
หมายเหตุ: นำหนังสือและเอกสารที่ใช้ในการสอนทั้งหมดเข้าห้องทดลองทุกครั้งที่ทำการทดลอง

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
01072118 วิชา Data Structure and Algorithm Laboratory

การทดลองที่ 1 สแตก (Stacks)

วัตถุประสงค์

1. อิมพลิเมนต์ stack abstract data type ที่ใช้อาร์เรย์เป็นโครงสร้างข้อมูลพื้นฐาน (array-based implementation) โดยสร้างคลาส Stack
2. เขียนโปรแกรมทำแอพพลิเคชั่น Parenthesis Matching ที่ใช้สแตกแก้ปัญหา โดยใช้คลาส Stack ที่สร้างขึ้น
3. แบบฝึกหัดเพิ่มเติม
   - 3.1. Evaluate Postfix Notation
   - 3.2. Parking Lodge



ความรู้พื้นฐาน

สแตกในความเป็นจริง(real world) คือกองซ้อนทับของสิ่งของ ของในกองมีลำดับการเอาของใส่กอง (push) และเอาของออกจากกอง (pop) ทำที่ปลายด้านบนเรียกว่าทอป (top) ของสแตก ทำให้ของที่เอาเข้าไปเป็นลำดับสุดท้ายถูกเอาออกจากสแตกก่อน สแตกจึงเป็น (Last-In First-Out) LIFO list ตัวอย่างสแตกในชีวิตประจำวันได้แก่ กองจานในโรงอาหาร การใส่ถุงเท้ารองเท้า ลูกชิ้นปิ้งเสียบไม้

แอบสแตรกดาต้าไทป์ (Abstract Data Type) หมายถึง การกำหนดส่วนดาต้าและส่วนโอเปอร์เรชั่นเพื่อแทนรูปแบบ (model) ที่ต้องการ เช่น integer array stack queue linked list tree graph

สแตกแอบสแตรกดาต้าไทป์ (Stack Abstract Data Type)

1. **ส่วนดาต้า** เป็นกองซ้อนทับของสิ่งของ ของในกองมีลำดับ มีปลาย 1 ด้านเรียกว่าทอป (top) ของสแตก
2. **ส่วนโอเปอร์เรชั่นพื้นฐาน** เมื่อ s เป็นสแตก และ i เป็นข้อมูลใดๆของสแตก s
   - 2.1. push(s, i) ใส่ข้อมูล i เข้าไปที่ทอปของสแตก
   - 2.2. i = pop(s) นำข้อมูลที่ทอปของสแตกออกมาเก็บไว้ที่ i
   - 2.3. init(s) ให้ค่าตั้งต้นแก่สแตก s ให้เป็นสแตกว่าง

   อาจต้องการ**โอเปอร์เรชั่นเพิ่มเติม** เช่น
   - 2.4. empty(s) ตรวจสอบว่าสแตก s ว่าง (ไม่มีของใดๆ อยู่ในสแตก) หรือไม่
   - 2.5. i = top(s) ดูข้อมูลที่ทอปของสแตกมาเก็บไว้ที่ i โดยไม่ pop ข้อมูลออกจากสแตก
   - 2.6. full(s) ตรวจสอบว่าสแตก s เต็มหรือไม่

การทดลองที่ 1 อิมพลิเมนต์(หรือเรพพรีเซนท์)สแตกแอบสแตรกดาต้าไทป์ **(Stack Abstract Data Type Implementation/Representation)**

1. **ส่วนดาต้า** คิดตามและตอบคำถามข้างล่าง หากไม่แน่ใจ ตรวจสอบคำตอบจากฟุตโนตท้ายหน้า

1. จะใช้อะไรเป็นกองซ้อนทับของสิ่งของ ?_____[1]

   (ดาต้าไทป์[2]อะไร ? ที่สามารถเก็บของ(ชนิดเดียวกัน)ได้หลายๆ อัน และของที่เก็บไว้สามารถมีลำดับได้)

2. วาดรูปดาต้าไทป์ที่เลือกเพื่อแทนสแตกในความเป็นจริงด้านซ้าย

3. จะใช้ดาต้าไทป์อะไรแทน top? เพื่อทำโอเปอร์เรชั่นต่างๆ เช่น push/pop ทำที่ top ทั้งสิ้น ตอบ _____[3]
4. วาดรูปแสดงดาต้าไทป์และค่าของ top ลงในรูปสแตกที่วาดข้างต้น top มีค่า _____[4]
5. จากข้อมูลข้างต้น สร้างคลาส stack ของ char ขนาด 10 โดยเติมส่วน data ลงในคลาส

```
class stack {
private :
        _____[5]          //for keeping stack items
        _____[6]          //to show the top of stack
};
```

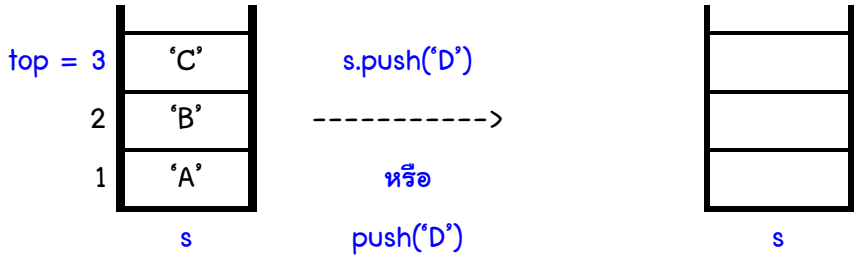
6. เพื่อให้ไทป์ของของในสแตกและขนาดของอะเรย์เปลี่ยนได้ ทำ template โดยให้ทั้งสองส่วนเป็นพารามิเตอร์

```
template <class T,int MAX = 10>
class stack {
private :
      int top;
      T items[MAX];
};
stack sc<char>;              //define sc เป็น สแตกของ char ขนาด 10 ซึ่งเป็นค่า default ของ MAX
stack si<int, 50>;           //define si เป็น สแตกของ int ขนาด 50
```

2. **ส่วนโอเปอร์เรชั่นพื้นฐาน** เมื่อ s เป็นสแตก และ i เป็นข้อมูลใดๆ ของสแตก s

   1. push(s, i) ใส่ข้อมูล i เข้าไปที่ทอปของสแตก

      1. วาดรูปสแตก s หลังทำโอเปอร์เรชั่น s.push('D') พร้อมทั้งแสดงค่า top ใหม่



---

[1] array

[2] หาจากดาต้าไทป์พื้นฐาน (char int double struct array pointer ...) หรือดาต้าไทป์ที่มีอยู่แล้ว (string vector ....)

[3] int top;

[4] 3 (เริ่มอันแรกที่ index 0)

[5] `char items[10];`

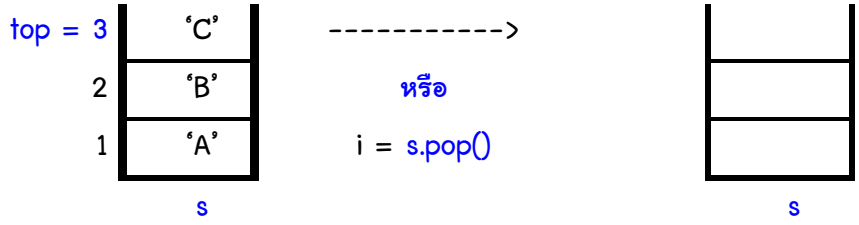
[6] `int top;`

2. push push เมมเบอร์ฟังก์ชั่น ต้องการ parameter อะไร? กี่ค่า? typeใด? และต้อง pass โดยวิธีใด?
   เขียน parameter ของฟังก์ชั่น push _____[7]
3. push ต้องมีค่า return หรือไม่? return type ต้องเป็น _____[8]
4. push ทำให้ data ของสแตกเปลี่ยนแปลงอย่างไรบ้าง? ต้องเขียนโคดอย่างไร?
   1. ส่วน top _____[9]
   2. ส่วน array _____[10]
   3. push ทำให้ออปเจคเปลี่ยน state (มีการเปลี่ยนค่าของออปเจค) จึงไม่เป็น **const** ฟังก์ชั่น
5. เขียน push จากข้อมูลข้างต้น ทั้งโปรโตไทป์ในคลาสเดฟฟินิชั่น และตัวฟังก์ชั่นเดฟฟินิชั่นนอกคลาส[11]

```
template <class T,int MAX = 10>
class stack {
      int top;
      T items[MAX];
public :
      _____[12]      //push prototype เพื่อ push i ลงบน top ของสแตก
};

template <class T,int MAX = 10>
void stack<T, MAX>::push(T& i){ //push function definition ของคลาส stack
                                                    [13]
   ____________________________________________
}
stack sc<char>;                 //define sc เป็น สแตกของ char ขนาด 10
```

2. i = pop(s) นำข้อมูลที่ทอปของสแตกออกมาเก็บไว้ที่ i
   1. วาดรูปสแตก s หลังทำโอเปอร์เรชั่น i = s.pop() พร้อมทั้งแสดงค่า top ใหม่



   2. pop เมมเบอร์ฟังก์ชั่น ต้องการ parameter หรือไม่? _____[14]
   3. pop ต้องมีค่า return กลับจากฟังก์ชั่นหรือไม่? typeใด? พาสกลับแบบใด?
      _____[15]

---

[7] มีค่าเดียวคือค่า item ใหม่ที่จะใส่เข้าไปในสแตกค่าเดียวเป็น type เดียวกับของในสแตก ควรพาสแบบบายวาลูเพราะเอาไปใช้เฉยๆ ไม่ได้ต้องการเปลี่ยนค่าพารามิเตอร์โดยฟังก์ชั่น push แล้วส่งกลับออกมา แต่การพาสบายวาลูต้องเรียก copy constructor จึงใช้การพาสแบบ const & ดังนี้ T& i

[8] ไม่มี

[9] เปลี่ยนค่า top ไปชี้ที่ top ใหม่ **++top**

[10] ใส่พารามิเตอร์บน top ใหม่ของสแตก **item[top] = i**

[11] เนื่องจากเป็น template ฟังก์ชั่น จึงต้องมีบรรทัดบอกไว้ว่าเป็น template < > หร้อมทั้งระบุพารามิเตอร์ของเทมเพลท และทุกที่ที่ใช้ stack เป็น type ต้องใส่พารามิเตอร์ของเทมแพลทด้วยคือ stack<T, 10> ดังแสดงในส่วนของโปรแกรม

[12] **void push(T&);**

[13] **void stack<T, MAX>::push(T& i){ item[++top] = i;}**

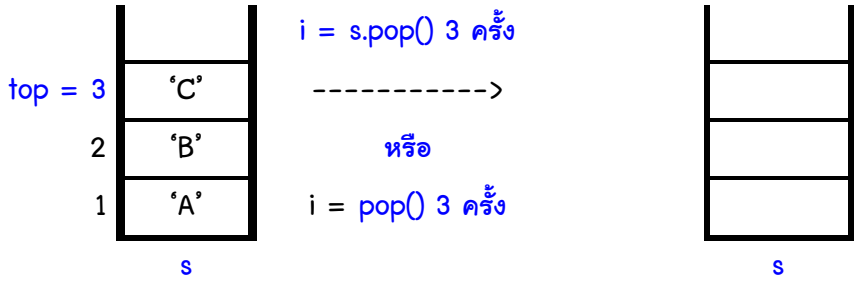
[14] ไม่มี

[15] ต้องการ item ที่อยู่ที่ top เป็น type เดียวกับของในสแตก ในที่นี้คือ T คือ return ค่า items[top] แบบบายวาลู

4. pop ทำให้ data ของสแตกเปลี่ยนแปลงอย่างไรบ้าง?
   1. ส่วน array _____[16]
   2. ส่วน top _____[17]
   3. pop ทำให้ออปเจคเปลี่ยน state จึงไม่เป็น const ฟังก์ชั่น
5. เขียน pop จากข้อมูลข้างต้น ทั้งโปรโตไทป์ในคลาสเดฟฟินิชั่น และตัวฟังก์ชั่นเดฟฟินิชั่นนอกคลาส

```
template <class T,int MAX = 10>
class stack {
      int top;
      T items[MAX];
public :
      _____[18]  //pop prototype เพื่อ pop top ของสแตกออกมา
};
template <class T,int MAX = 10>
T stack<T, MAX>::pop(){//pop function definition ของคลาส stack
                                                    [19]
   ____________________________________________________
}
```

3. init(s) ให้ค่าตั้งต้นแก่สแตก s ให้เป็นสแตกว่าง
   1. วาดรูปสแตก s หลังทำโอเปอร์เรชั่น s.pop() ไป 3 ครั้งจนของหมอ พร้อมทั้งแสดงค่า top ใหม่



   2. การให้ค่าตั้งต้นของออปเจคใน C++ ทำได้โดยฟังก์ชั่นที่เรียกว่า _____[20] ซึ่งเป็นฟังก์ชั่นชื่อเหมือนกับชื่อคลาสและไม่มี return ไทป์
   3. จากรูปของสแตกว่าง ค่าของ top = _____[21]
   4. ส่วน array ไม่มีของ จึงไม่ต้องให้ค่าตั้งต้น ดังนั้นเมื่อดีฟายด์ออปเจคใหม่ array จะมีค่าขยะเก่าค้างอยู่ แต่ไม่เป็นไร สำคัญที่ค่า top
   5. constructor ให้ค่าตั้งต้นออปเจคที่ดีฟายด์ เปลี่ยน state ของออปเจค จึงไม่ใช่ const ฟังก์ชั่น
   6. เขียน constructor ของคลาส โดยใช้ข้อมูลในข้อข้างต้น เนื่องจากฟังก์ชั่นสั้นมาก เราอาจทำให้เป็น inline ฟังก์ชั่น ซึ่งทำได้ 2 แบบ คือ เขียน function prototype และ function definition แบบเดิม แต่เติมคีย์เวิร์ดคำว่า inline หน้า function definition หรือแบบที่ 2 เขียนตัวฟังก์ชั่นเข้าไปในคลาสเดฟฟินิชั่นเลยจะทำให้เป็น inline โดยอัตโนมัติ ข้างล่างเป็นแบบแรก

---

[16] ต้องการเอา item ที่ top ออกมาเป็นค่า return กลับ แต่จริงๆ แล้วไม่จำเป็นต้องลบค่าที่อยู่ที่ top ก็ได้ เพียงแต่เปลี่ยนค่า top ให้ลดลงไป 1 ก็พอ

[17] เปลี่ยนค่า top ลดลง 1 top- -

[18] `T pop();`

[19] `T stack<T, MAX>::pop(){ return item[top--];}`

[20] constructor

[21] -1

```
template <class T,int MAX = 10>
class stack {
      int top;
      T items[MAX];
public :
      ____________________________________[22] //constructor prototype
};
template <class T,int MAX = 10>
stack<T, MAX>::stack(){//constructor definition
   ____________________________________________________[23]
}
```

7. เราสามารถใช้กลไกของ **initializer list** ให้ค่าตั้งต้นของ data ของออปเจคโดยใช้ชื่อฟิลด์และวงเล็บค่าที่ต้องการให้ไว้ภายใน ในที่นี้ต้องการให้ค่า top ค่าเดียว ดังแสดงข้างล่าง ใช้การเขียนตัวบอดี้ของฟังก์ชั่นในคลาสเดฟฟินิชั่นเป็นการแสดง inline ฟังก์ชั่น

```
template <class T,int MAX = 10>
class stack {
      int top;
      T items[MAX];
public :
      stack():top(-1){} //constructor init empty stack
};
```

```
template <class T,int MAX = 10>
class stack {
      int top;
      T items[MAX];
public :
      stack():top(-1){}                  //constructor init empty stack
      ____________________________________ //top prototype
      ____________________________________ //inline empty function
      ____________________________________ // inline full function

};
template <class T,int MAX = 10>
const T& stack<T, MAX>::top() const {//top definition
   ____________________________________________________
}
```

4. i = top(s) สำหรับตรวจสอบค่าข้อมูลที่ทอปของสแตกโดยไม่นำข้อมูลออกจากสแตก ค่าที่ได้เก็บไว้ที่ i

   1. พิจารณา พารามิเตอร์ ค่ารีเทอร์นกลับ วิธีการพาสค่าเข้า/ออก เขียน top[24]

   2. เนื่องจาก top ไม่ได้เปลี่ยนค่าใดๆ ของออปเจค เพียงแต่ดูค่าเฉยๆ จึงควรให้ top เป็น **const function**

---

[22] `stack();`

[23] `stack<T, MAX>::stack(){top = -1;}`

[24] `T stack<T, MAX>::top(){ return item[top];}`

3. เหตุที่ return type เป็น const & ผิดกับ pop ที่มี return type เป็น by value เนื่องจาก logic ของ top ต้องการเพียงดูค่าที่ top เฉยๆ ไม่ต้องการนำไปทำอย่างอื่น จึงไม่พาส by value ซึ่งเรียก copy constructor มาคัดลอกออปเจคบนสแตกออกมา ทำให้เสียเวลา แต่ pop เอาออปเจคออกไป อาจต้องนำไปทำอะไร จึงต้องคัดลอกมา

5. empty(s) สำหรับตรวจสอบว่าสแตก s ว่าง (ไม่มีของใดๆ อยู่ในสแตก) หรือไม่

   คอนดิชั่นใดที่ทำให้สแตกว่าง _____[25] เขียนโอเปอร์เรชั่น empty ให้เป็น inline function[26]

6. full(s) สำหรับตรวจสอบว่าสแตก s เต็มหรือไม่ (ถ้าใส่เข้าไปจะเกิน range ของ array)

   คอนดิชั่นใดที่ทำให้สแตกเต็ม _____[27] เขียนโอเปอร์เรชั่น full ให้เป็น inline function[28]

7. ต่างจากสแตกจริง สแตกที่อิมพลิเมนต์มีหน่วยความจำจำกัด จึงต้องตรวจการเกิน range ของ array push ทำไม่ได้เมื่อสแตกเต็ม (full) เรียกสแตกโอเวอร์โฟล์ (**stack overflow**) pop ทำไม่ได้เมื่อสแตกว่าง (empty) เรียกสแตกอันเดอร์โฟล์ (**stack underflow**) เพิ่มการตรวจสอบใน push[29] และ pop ที่เขียนไปแล้ว

8. เขียนฟังก์ชั่น print เพื่อพิมพ์ของในสแตก และ ฟังก์ชั่น make_empty เพื่อทำให้สแตกกลายเป็นสแตกว่าง

9. พิจารณาว่าต้องเขียน copy constructor operator= และ destructor หรือไม่? ถ้าไม่เขียนและใช้ copy copy constructor operator= และ destructor ตัวที่คอมไพเลอร์สร้างให้แล้ว มันทำอะไรให้ในแต่ละฟังก์ชั่น?[30]

   copy constructor_____

   _____

   operator= _____

   _____

   destructor _____

   _____

<u>การทดลองที่ 2</u> จะเขียนโปรแกรมเรียกใช้คลาส stack ที่ implement เพื่อทดสอบให้แน่ใจว่าใช้งานได้ถูกทุกกรณีได้อย่างไร? อาจลองใช้ตัวทดสอบในหน้าถัดไป

---

[25] ค่า top เป็น -1

[26] `bool empty(){return top == -1;}`

[27] ค่า top กท่ากับ MAX -1

[28] `bool full(){return top == MAX - 1;}`

[29] `if (top == MAX - 1) throw overflow_error("stack overflow"); else items[++top] = i;`

[30] ไม่ต้องเขียนเลย เพราะstack ไม่มีdata เป็น pointer จึงไม่มีการ allocate memory แบบ dynamic ดังนั้น การ copy ของ copy constructor และ operator= ไม่ต้อง allocate memory แบบ dynamic เพื่อไม่ให้ data pointer ชี้ไปที่เดียวกัน ส่วน destructor ไม่ต้องเก็บกวาดอะไร จึงสามารถใช้ version ที่คอมไพเลอร์ synthesize ให้ได้ทั้ง 3 routines โดย copy constructor และ operator= ของคอมไพเลอร์ทำการคัดลอก byte by byte ส่วน destructor ของคอมไพเลอร์ไม่ได้ทำอะไร

```
void main(){
// 1. testing stack
   const int MAXSTACK = 2;
   cout << "======== stack testing part 1 ========\n";
   stack<char,MAXSTACK> s;
   try {
      s.print();
      s.push('A');      s.push('B');          s.print();
      s.push('C');      s.print();
      s.pop();          s.print();
   } catch (exception &e) {
      cerr << "---------In part 1-----------\n";
      cerr << "Caught: " << e.what( ) << "\n";
      cerr << "-------------------------------";
}
```

การทดลองที่ 3

1. ใช้สแตกที่สร้างขึ้นแก้ปัญหา parenthesis matching โดยสร้างโปรแกรมรับสตริงอินพุทแล้วตรวจสอบว่าสตริงนั้นมีวงเล็บเข้าคู่กันอย่างถูกต้องหรือไม่ (น.ศ.อาจฝึกออกแบบเองโดยใช้ Warnier-Orr Diagram Design Tool หากทำไม่ได้ค่อยไปดูเฉลยที่เรียนไปแล้วในภาคทฤษฎี)
2. เติม data structure เพื่อเช็คสอบเพิ่มเติม เช่น ผิดที่ตำแหน่ง (char) ตัวใด ผิดเพราะเหตุใด เช่น วงเล็บเปิดปิดไม่เข้าคู่กัน ขาดวงเล็บปิด วงเล็บเปิดเกิน
3. หากเริ่มต้นไม่ได้ อาจเริ่มต้นด้งนี้

```
//2. try parenthesis matching problem
   char c;
   char open;
   int err_pos = 0;  //error position
   int error = 0; //err.type(not match, LackOpenParen, LackCloseParen)
   bool b;

   cout << "\n\n========= part 2 =========\n";
   s.make_empty();
   s.print();
   cout << "Enter expression to test parenthesis matching problem : ";
   try {
      while (cin >> c && !error) {
         //your code
      }
      if (error == 1)
         //your code
   } catch (exception &e) {
      cerr << "------In part 2 : parenthesis matching--------\n";
      cerr << "Caught: " << e.what( );
      cerr << "at position: " << err_pos << "of input." << "\n";
   }//catch
```

การทดลองที่ 4 การทดลองนี้สำคัญที่ต้องคิด algorithm เพื่อใช้แก้ปัญหาโดยใช้ stack
ใช้สแตกที่สร้างขึ้นแก้ปัญหาโปรแกรมที่จำลองที่จอดรถ (parking lot) ซึ่งสามารถจอดรถได้เรียงเดี่ยว โดยมีที่สำหรับให้จอดรถได้มากที่สุด 10 คัน ทุกครั้งที่มีคนนำรถมาจอด จะมีการบันทึกเลขทะเบียนและออกใบรับรถคันนั้น เมื่อเจ้าของรถมารับรถเจ้าหน้าที่จะเลื่อนรถคันอื่น ๆ ที่จอดซ้อนท้ายออกเพื่อนำรถมาให้ จากนั้นก็จะเลื่อนรถคันที่ถูกเลื่อนออกมากลับเข้าที่ตามลำดับเดิม (ให้ใช้สแตกแก้ปัญหาคือการเอารถเข้า/ออกต้องใช้ push/pop เท่านั้น ห้ามเอารถออกโดยเอาออกจาก array แล้วเลื่อนคันอื่นๆ ลงมา เพราะเรากำลังศึกษาเรื่องสแตกอยู่ ในการแก้ปัญหา อาจใช้สแตกมากกว่า 1 สแตกได้)
การจำลองโดยใช้โปรแกรมให้ทำดังนี้

- แสดงทะเบียนรถที่เข้าหรือออกจากที่จอดรถทุกครั้ง พร้อมทั้งบอกด้วยว่ามีที่ว่างเหลือสำหรับจอดรถได้อีกกี่คัน หลังจากที่นำรถเข้าหรือออกจากที่จอดแล้ว
- ทุกครั้งที่มีรถมาจอด ถ้าไม่มีที่ว่างให้บอกว่าเต็ม และไม่ให้จอด
- แสดงลำดับการเคลื่อนย้ายรถเพื่อนำรถที่เจ้าของมารับออก ทั้งลำดับการเลื่อนรถออก และลำดับการเลื่อนรถกลับเข้าที่

ตัวอย่าง input (A=arrive D=depart Q=quit)	ตัวอย่าง output	
A 10	10 arrived	spaces left = 9
A 20	20 arrived	spaces left = 8
A 30	30 arrived	spaces left = 7
A 40	40 arrived	spaces left = 6

การทดลองที่ 5 (ทำเป็นแบบผึกหัด หากต้องการ) ใช้สแตกที่สร้างขึ้นแก้ปัญหา postfix (evaluating postfix notation เช่น 1 2+4 3+*) โดยสร้างโปรแกรมรับ postfix notation แล้วเอ้าพุทคำตอบ

ตัวอย่าง การนิยามสแตกทำได้หลายลักษณะด้วยกันดังแสดงข้างล่าง
นักศึกษาอาจลองเปรียบเทียบคลาสที่เราทำขึ้นกับคลาส std::stack ซึ่งเป็นสแตกในไลบรารีมาตรฐานของภาษา C++ แล้วพิจารณาเหตุผลว่าทำไมเราจึงออกแบบคลาสต่างออกไปจากคลาสที่มีให้ใช้อยู่แล้ว?

ตัวอย่างที่ 1 ใช้ structure + function (หรือรูปแบบที่ใช้ในภาษา C แต่ดัดแปลงมาเขียนตามไวยากรณ์ภาษา C++)

```
// The following declarations are put into the header file
// and their definitions are put into some source file.
// T is a placeholder for the type of the data item in the stack.
typedef T Stack_entry;

enum Stack_error_code { // This type is for error checking and reporting.
    STACK_UNDERFLOW, STACK_OVERFLOW, // ...
};

struct Stack {
    // ... underlying data structure
};

void stack_initialize(Stack* s);
void stack_destroy(Stack* s); // declare and define only when necessary
bool stack_empty(const Stack* s);

Stack_error_code stack_get_top(const Stack* s, Stack_entry* item);
```

```
Stack_error_code stack_set_top(Stack* s, const Stack_entry* item);

Stack_error_code stack_push(Stack* s, const Stack_entry* item);
Stack_error_code stack_pop(Stack* s);
```

ตัวอย่างข้างต้น แสดงเฉพาะฟังก์ชันหลัก ๆ ที่มีใช้ตามนิยามของสแตกเท่านั้น ในการสร้างขึ้นใช้จริงอาจมีฟังก์ชันอื่น ๆ เสริมได้ เช่น ฟังก์ชันที่ใช้ในการคัดลอกข้อมูลในสแตก เพื่อให้ได้สแตก 2 ชุดที่มีข้อมูลเหมือนกัน เป็นต้น

<u>ตัวอย่างที่ 2</u> ใช้ class เพื่อนิยาม ADT (Abstract Data Types)

```
// The following class definitions is put into the header file
// and their implementation details are put into some source file.

// T is a placeholder for the type of the data item in the stack.
typedef T Stack_entry;

// Stack_underflow and Stack_overflow for error checking and reporting
class Stack_overflow: public std::exception {
    // ...
};

class Stack_underflow: public std::exception {
    // ...
};

class Stack {
public:
    Stack();

    // Copy constructor, assignment op, and destructor
    // are declare and define only when necessary.
    Stack(const Stack& rhs);
    Stack& operator=(const Stack& rhs);
    ~Stack();

    bool empty() const;

    // both top function may throw Stack_underflow
    Stack_entry& top();             // mutator (or setter) function
    const Stack_entry& top() const; // accessor (or getter) function

    void push(const Stack_entry& item); // may throw Stack_overflow
    void pop();                         // may throw Stack_underflow
private:
    // ... underlying data structure
};
```

เฉลยสำหรับอาจารย์คุม lab อ.กบกรุณาเก็บไว้เพื่อเฉลยให้น.ศ.ภายหลัง

```
//HOW TO USE ONLINE HELP
//just type the topic you want anywhere in the source file
//for example to understand template type template and click on
//that word you will see the topic about it on Dynamic Help window
//just click on the topic you want. This is a very good help since
//there are both explanations and examples.
#include <iostream>
#include <stdexcept>
using namespace std;
template <class T,int MAX = 10> //pass integer MAX to the template, see online help
for more information
class stack {
public:
   stack():top(-1){}           //constructor init empty the stack
   void push(const T& x);  //push item x on the top of the stack
   T pop();                          //pop the stack, return the poped item
   const T& topstack() const;//return item at the top but NOT pop it out
   bool empty(){return top == -1;}//check if the stack is empty
   bool full(){return top == MAX-1;} //check if the stack is full
   void make_empty() {top = -1;}//make the stack empty
   void print();  //print items in the stack from bottom to top
private :
   int top;
   T items[MAX];
};

template <class T,int MAX = 10>
void stack<T, MAX>::push(const T& x){
   if (top == MAX - 1)
      throw overflow_error("Stack overflow from push().");
   else items[++top] = x;
}
template <class T,int MAX = 10>
T stack<T, MAX>::pop(){
   if (top == - 1)
      throw underflow_error("Stack underflow from pop().");
   else return items[top--];
}
template <class T,int MAX = 10>
const T& stack<T,MAX>::topstack() const{
   if (top == - 1)
      throw underflow_error("Stack underflow from topstack().");
   else return items[top];
}
template <class T,int MAX = 10>
void stack<T,MAX>::print(){
   if (top == -1) cout << "nothing in the stack";
   for(int i=0; i <= top; i++)
      cout<<items[i]<<" ";
   cout<<"\n";
}
// for part 2
bool notmatch(char open, char close){
   return (!((open == '(' && close == ')') ||
             (open == '{' && close == '}') ||
             (open == '[' && close == ']')));
}
```

```
void main(){
// 1. testing stack
   const int MAXSTACK = 2;
   cout << "======== stack testing part 1 ========\n";
   stack<char,MAXSTACK> s;
   try {
      s.print();
      s.push('A');
      s.push('B');
      s.print();
      s.push('C');
      s.print();
      s.pop();
      s.print();
   } catch (exception &e) {
      cerr << "---------In part 1-----------\n";
      cerr << "Caught: " << e.what( ) << "\n";
      cerr << "-------------------------------";
}



//2. after it works try parenthesis matching problem
  char c;
   char open;
   int err_pos = 0;  //eror position
   int error = 0; //err.type(not match, LackOpenParen, LackCloseParen
   bool b;

   cout << "\n\n========= part 2 =========\n";
   s.make_empty();
   s.print();
   cout << "Enter expression to test parenthesis matching problem : ";
   try {
      while (cin >> c && !error) {
         err_pos++;
         if (c == '(' || c == '{' || c == '[') // find open paren.
            s.push(c);
         else if (c == ')' || c == '}' || c == ']')// find close paren.
            if (s.empty())
               error = 2;  //lack of open paren for the close paren.
            else if (b = notmatch(open = s.pop(), c))
               error = 1;  //open and close parens are not match.
      }
      if (error == 1)
         cout << "error at pos: " << err_pos << " close paren not match the open "
<< open <<"\n";
      else if (error == 2)
         cout << "error at pos: " << err_pos << " lack of open paren for the close
paren. "<< c <<"\n";
      else if (s.empty())
         cout << "MATCH" <<"\n";
      else  cout << " open paren. is exceed " << s.topstack() <<"\n";
   } catch (exception &e) {
      cerr << "------In part 2 : parenthesis matching--------\n";
      cerr << "Caught: " << e.what( );
      cerr << "at position: " << err_pos << "of input." << "\n";
   }
}//main()
```