เฉลย ADT

1. กำหนดให้ char และ int มีขนาด 1 และ 4 byte ตามลำดับ และสมมติว่าไม่มีการจัดระเบียบข้อมูล

```
typedef struct info {
    char name[20];
    int age;
} info_type; // 24 bytes

typedef union data {
    int myInt;
    char myString[100];
} data_type; // 100 bytes

typedef struct myStruct {
    info_type myInfo[15];
    data_type myData[20];
} type_abc; // 24*15 + 100*20 = 2,360 bytes

type_abc *p = (type_abc*)calloc(sizeof(type_abc), 50);
// มีการปรับแก้โจทย์บรรทัดประกาศ type_abc นิดหน่อย เพราะเชื่อว่าต้นฉบับน่าจะมาแบบนี้
```

หากกำหนดให้ p[0] เริ่มต้นที่ตำแหน่ง 250 จงหาตำแหน่งใน memory ของสิ่งที่กำหนด

1.1. p[13] -> myInfo[2] -> name

1.2. p[21] -> myData[12] -> myString[18]

Solution: สำหรับข้อนี้ ให้พิจารณาเนื้อที่ในหน่วยความจำดังนี้

- info_type กินเนื้อที่เท่ากับ char name[20] + int age = 20 + 4 = 24 bytes
- data_type กินพื้นที่เท่ากับ**ตัวมากสุด** เพราะประกาศแบบ union ดังนั้นกินที่เท่า char myString[100] = 100 bytes
- ดังนั้น type_abc จะกินเนื้อที่เท่ากับ info_type*15 + data_type*20 = 2,360 bytes

หาก p[0] เริ่มที่ตำแหน่ง 250 (250 ฐาน 16 = 512 + 80 = 592)

- p[13] จะเริ่มที่ 592 + 13*(2,360) = 31,272 → แปลงเป็นฐาน 16 → 7A28
  - myInfo[2] จะเริ่มที่ 31272 + 2*24 = 31,320 → แปลงเป็นฐาน 16 → 7A58
    - name จะอยู่หน้าสุด ก็คือ 7A58 เท่า myInfo[2]
- p[21] จะเริ่มที่ 592 + 21*(2,360) = 31,272 → แปลงเป็นฐาน 16 → C3E8
  - myData[12] อยู่หลัง myInfo ทั้ง 15 ตัว = 31,272 + (15*myInfo + 12*myData) = 31,272 + (15*24 + 12*100) = 32,832 → แปลงเป็นฐาน 16 → 8040
    - myString[18] ก็บวกไปอีก 18 ตัว เป็น 32,832+18 = 32,850 → แปลงเป็นฐาน 16 → 8052

2. จงเขียนโปรแกรม Tower of Hanoi โดยกำหนด function main ให้ดังนี้

```
void Hanoi(int n, char From, char To, char Mid) {
    if(n>0){
        Hanoi(n-1, From, Mid, To);
        printf("Moving from %c to %c", From, To);
        Hanoi(n-1, Mid, To, From);
    }
}
```

https://github.com/srakrn/2017-sophomore-semester-1/blob/master/ADT/hanoi_tower.c

Solution: วิธีการทำของข้อนี้ หากต้องการจำเลย ให้จำว่าเราจะเรียกฟังก์ชั่น Hanoi() สองครั้ง เรียกก่อนแสดงผลหนึ่งครั้ง และหลังแสดงผล โดยพารามิเตอร์แต่ละตัวจะสลับที่กันดังนี้

ประกาศฟังก์ชั่น	เสาต้นทาง (F)	เสากลาง (A)	เสาปลายทาง (T)
เรียกก่อน print	เสาต้นทาง (F)	เสาปลายทาง (T)	เสากลาง (A)
เรียกหลัง print	เสากลาง (A)	เสาต้นทาง (F)	เสาปลายทาง (T)

วิธีจำของเรา: ขายมัน (FAT) ที่เขตการค้าเสรี (FTA) อาเซียน (AFT)

3.1 จงอธิบายการเรียก function และการส่งผ่าน parameter แบบ pass by value

Solution: การเรียกฟังก์ชั่นโดยส่งพารามิเตอร์ผ่านค่า (pass by value) จะเป็นการส่งค่าของตัวแปรไปยังหน่อยความจำส่วนอื่น และค่าในฟังก์ชั่นและค่านอกฟังก์ชั่นจะเป็นคนละตัวแปรกัน

3.2 จงอธิบายการเรียก function และการส่งผ่าน parameter แบบ pass by reference

Solution: การเรียกฟังก์ชั่นโดยส่งพารามิเตอร์ผ่านการอ้างอิง (pass by reference) จะเป็นการส่งตำแหน่งของหน่วยความจำไปเพื่อใช้อ้างอิงค่า ทำให้เมื่อเกิดการเปลี่ยนแปลงค่าในฟังก์ชั่น จะทำให้ค่านอกฟังก์ชั่นเปลี่ยนแปลงด้วยเพราะอยู่บนหน่วยความจำตำแหน่งเดียวกัน

4. จงเขียน infix เป็น postfix และ prefix

5. จงเขียน postfix เป็น infix

6. จงเขียน prefix เป็น infix

9. จงสร้างโปรแกรมเพื่อตรวจสอบความถูกต้องของสตริง โดยมีรูปแบบดังนี้ wRxLyMz

- w,y เป็น String ที่ประกอบไปด้วย ตัวอักษร a,b,c หรือ d ความยาวเท่าไรก็ได้
- x,z เป็น String ที่เป็นส่วนกลับของ String w และ y ตามลำดับ

กำหนด function ที่ต้องใช้คือ

- void push(char item) จะทำหน้าที่ดึง item เข้า stack
- void pop(char *item) จะทำหน้าที่ดึงข้อมูลใน stack มาเก็บในบริเวณที่ item ชี้ไว้
- int stringDetect(char * string) ไว้ตรวจสอบความถูกต้องของ string ที่กำหนด

โดยที่

- return 0; เมื่อ String ถูกต้องตาม pattern ทั้งหมด
- return 1; เมื่อ String w ผิด
- return 2; เมื่อ String x ผิด
- return 3; เมื่อ String y ผิด
- return 4; เมื่อ String z ผิด
- return 5; เมื่อมีการผิดในลักษณะอื่นๆ