Assignment 4.1 String

- **Topics** การสร้างและตรวจสอบรูปแบบคำสั่ง
- **Learning outcomes** สามารถเขียนโปรแกรมวิเคราะห์ความถูกต้องของสตริงในรูปแบบที่กำหนด
- **โจทย์ปัญหา** สมมุติให้มีคำสั่งที่ใช้งานได้ ให้ตรวจสอบว่าคำสั่งที่ป้อนถูกต้องหรือไม่
  - ลักษณะคำสั่ง (case insensitive) แบ่งออกเป็น 3 กลุ่ม

  1. คำสั่งที่มีเพียง token เดียว ห้ามมิให้มีคำอื่นปนในบรรทัด
     list, end, sort, pop, help, sqrt , rec , neg , pow , + , - , * , /
  2. คำสั่งที่ต้องตามด้วยตัวเลข 1 ตัว ต้องมีตัวเลข 1 ตัวตามหลังคำสั่ง(ใช้เว้นวรรคคั่น)
     delete <n>, search <n>, peek <n> , push <n>
  3. คำสั่งที่ต้องตามด้วยตัวเลข 1 ตัว ต้องมีตัวเลขตามหลังอย่างน้อย 1 ตัว
     add <n1> [n2 n3 n4 ...] , insert <n1> [n2 n3 n4 ...]

- **ขั้นตอน** เขียนโปรแกรมวนรอบอ่านค่าจนกว่าจะเจอคำสั่ง end
  - ถ้าป้อน syntax ผิดรูปแบบจาก 1, 2, 3 ให้แจ้ง Error
  - ถ้าป้อนรูปแบบ 2 หรือ 3 ให้แยกคำสั่งออกจาก ตัวเลข
    - ถ้าในส่วนของตัวเลขมีการพิมพ์ผิด/พิมพ์เกิน ให้แจ้ง Error

Assignment 4.1 String

- ตัวอย่างฟังก์ชัน
  - อ่านข้อมูล(gets) สร้างฟังก์ชันแบ่งคำ(strtok) ใส่ในอาร์เรย์(cmd[][]) จำนวน count
  - เช็คข้อมูลตัวแรกของอาร์เรย์(cmd[0]) สร้างฟังก์ชัน แยกกลุ่มคำสั่ง

    1 = "end","sort","pop","sqrt","rec","neg" ,"+","-","*","/","pow"

    2 = "peek","search","delete","push"

    3 = "insert","add"

    0 = "syntax error"

    คำสั่งกลุ่มที่ 0 ให้แสดงผล syntax error
- การตรวจสอบพารามิเตอร์ Parameter error
  - ตรวจสอบพารามิเตอร์ทุกตัวว่าเป็นตัวเลข
  - คำสั่งกลุ่มที่ 1 (ไม่ต้องการ พารามิเตอร์เช่น end, sort, pop, sqrt, nec , ...)
    - group = 1 && count != 1
  - คำสั่งกลุ่มที่ 2 (ต้องการพารามิเตอร์ 1 ตัว เช่น push 10, delete 20 , ...)
    - group = 2 && count != 2
  - คำสั่งกลุ่มที่ 3 (ต้องการพารามิเตอร์อย่างน้อย 1 ตัว เช่น add 10 20 30)
    - group = 2 && count <2

Assignment 4.1 (Test Case)

```
command> [list] [sort] [pop]          // หมายถึงทดสอบ 3 ครั้ง(ไม่ต้องพิมพ์ []) ครั้งแรกพิมพ์ list ครั้งที่สอง sort ....
answer> OK
command> [sqrt ] , [neg] , [rec] , [*] [+] , [-] , [*] , [/] // คำสั่งถูกมีพารามิเตอร์ 1 ตัว
answer> OK
command> [Delete 1] , [search 2] , [PUSH 3] , [peek 4] // คำสั่งถูกมีพารามิเตอร์ 1 ตัว
answer> OK
command> [add 1] , [insert 1 2 3] // คำสั่งถูกมีพารามิเตอร์ ได้มากกว่า 1 ตัว
answer> OK
command> [push pop]  [end list sort]  // ครั้งสั่งผิด พิมพ์มากกว่า 1 คำสั่งในบรรทัดเดียว
answer> parameter error
command> [pip] [delete0]             // พิมพ์คำสั่งผิด
answer> syntax  error
command> delete x      // พารามิเตอร์ตามหลังไม่ใช่ตัวเลข
answer> parameter error
command> delete 1   2 // พารามิเตอร์เกิน
answer> parameter error
command> add                          // พารามิเตอร์ไม่ครบ
answer> parameter  error
command> add  123     // คำสั่งถูก
answer> OK
command> add  1   12   345           // คำสั่งถูก
answer> OK
command> add  1   12   34X           // มีพารามิเตอร์ที่ผิดบางตัว
answer> parameter  error
command> end                          // คำสั่งถูก จบโปรแกรม
answer> OK
End program
Program writen by  62070501xx xxxxxxxx  xxxxxxxxxxxxxxx
```

Assignment 4.2 (Linked List)

- สร้าง Linked List ของตัวเลขจำนวนจริง โดยใช้ภาษาซี วนรอบทำงานตามคำสั่งที่กำหนด
- โปรแกรมทำงานในลักษณะ line command ประกอบด้วย syntax คือ

  [คำสั่งเดี่ยว] [คำสั่ง เว้นวรรค ตัวเลข] [คำสั่ง เว้นวรรค ชุดตัวเลข]

  - **add list** เพิ่มชุดตัวเลข เข้าไปใน linked list ตามลำดับ ใส่ต่อจากตัวสุดท้าย
  - **insert list** เพิ่มชุดตัวเลขลงในลิงค์ลิสต์ คำสั่งนี้ใช้ได้เมื่อข้อมูลเรียงลำดับอยู่เท่านั้น
  - **push n** เพิ่มตัวเลข เข้าไปเป็นตัวแรกของ linked list
  - **peek n** เรียกดูข้อมูลในตำแหน่งที่ n ของ linked list (n เริ่มจาก 0,1,2...)
    - ถ้า n เป็น 0 ให้แสดงข้อมูลตัวแรก ถ้าเป็น -1 ให้แสดงข้อมูลตัวสุดท้าย
    - ถ้า n เกินจำนวนที่มีอยู่ ให้แสดงเตือนจำนวนข้อมูลสูงสุดที่มีอยู่
  - **delete n** ค้นหาเพื่อลบตัวเลขที่มีค่า n ออกจาก linked list โดยมีการถามยืนยันก่อนการลบ
    - ถ้าไม่เจอ ให้ฟ้อง not found
    - ถ้ามีข้อมูลซ้ำกันเกินกว่า 1 ตัว ให้วนถามเพื่อลบไปเรื่อยๆ จนหมด
  - **search n** ค้นหาตัวเลขที่มีค่า n ที่อยู่ใน linked list แสดงผลตำแหน่งที่เจอ
    - ถ้าไม่เจอ ให้ฟ้อง not found
  - [sqrt] [rec] [neg] ดึงข้อมูลตัวแรกมากระทำ เช่น $\sqrt{x}$, $1/x$, $-x$ แล้วใส่กลับ [+] [-] [*] [/] [pow] ดึงข้อมูล 2 ตัวแรกมากระทำเช่น $x2/x1$, $x2^{x1}$ แล้วใส่กลับ
  - **list** แสดงข้อมูลทั้งหมดใน linked list
  - **sort** เรียงลำดับข้อมูลใน linked list
  - **pop** ดึง(ลบ) ข้อมูลตัวแรก
  - **help** แสดงชุดคำสั่งที่โปรแกรมสามารถทำงานได้
  - **end** สั่งให้จบโปรแกรม

Assignment 4.2 (Linked List)

- ถ้าพิมพ์คำสั่งผิด หรือ พารามิเตอร์ผิด/ไม่ครบ ให้แสดง **Command error** แล้วรอรับคำสั่งใหม่
- ถ้าจำนวนข้อมูลไม่เพียงพอที่จะทำงานตามคำสั่งให้แสดงผล **Can't operation** แต่ถ้าข้อมูลหมดให้แสดงผล **No data**
- ถ้าคำสั่งทำงานได้สำเร็จ จะแสดงผลหรือไม่ก็ได้
- การแสดงผลหน้าจอโต้ตอบ จะมีการวนรอบแสดงผล 3 ข้อความ คือ
  - **list>** ใช้สำหรับแสดงข้อมูลที่มีอยู่ทั้งหมด ให้แสดงทุกครั้งก่อนรับคำสั่ง
  - **command>** ใช้สำหรับสถานะรอรับคำสั่ง
  - **answer>** ใช้สำหรับแสดงผลผลลัพธ์ของการทำงานนั้น หรือความผิดพลาดที่เกิดขึ้น

```
list> NULL                  //แสดงข้อมูลที่เก็บอยู่ ถ้าไม่มีให้แสดงผล NULL
command>                    //รอรับคำสั่ง
```

```
list> NULL
command>  Add xyz           //ถ้าพิมพ์ หรือ ใช้คำสั่งผิด
answer> Command error       //แสดงผลเมื่อป้อนคำสั่งผิด
```

```
list> NULL
command>  list              //ถ้าป้อนคำสั่งถูก แต่ไม่มีข้อมูล
answer> No data             //แสดงผลเมื่อไม่มีข้อมูล
```

```
list> NULL
command> Add  1             //ถ้าป้อนคำสั่งที่ทำงานได้
answer> success             //ถ้าทำคำสั่งสำเร็จจะมีบรรทัดนี้หรือไม่ก็ได้
```

```
list> 1
command>  +                 //ถ้าป้อนคำสั่งถูก แต่ข้อมูลไม่พอ (การบวก ต้องมีข้อมูล 2 ตัว)
answer> Can't operation     //แสดงผลเมื่อข้อมูลไม่พอ
```

```
list> 1                     //แสดงข้อมูลเก็บอยู่
command> delete 100         //ถ้าป้อนคำสั่งที่ทำงานไม่ได้ (หาข้อมูลที่ลบไม่เจอ)
answer> 100 Not found       //แสดงผลเมื่อคำสั่งทำงานไม่ได้
```

Assignment 4.2 Test Case

```
1.list> NULL
command> [pip] [delete] [add]  // ERROR
answer> Syntax error
list> NULL
command> [pop 10] [add 10.1.1] [add 3x]
answer> Parameter error
list> NULL
command> [delete 1] [neg] [+] // NULL1
answer> no data
list> NULL
2.command>  add  30   20   10   20     // add
list> 30 20  10  20
3.command> peek 0                    // peek
answer> 30
list> 30  20 10  20
command> peek 1
answer> 20
list> 30  20 10  20
command> peek 4
answer> Maximum peek = 3
list> 30  20 10  20
command> peek -1
answer> 20
list> 30  20 10  20
4.command> search 30              //search
answer> found 30 at [0]
list> 30  20 10  20
command> search 50              //search
answer> 50 not found
list> 30  20 10  20
```

```
5.command> delete 30              //delete
answer> 30 found  enter y  to confirm  [y]
list> 20 10  20
command> delete 100
answer> 100 Not command> delete 20
answer> 20 found  enter y  to confirm  [n]
answer> 20 found  enter y  to confirm  [y]
list> 20  10
command> delete 20
20 found  enter y  to confirm  [y]
list> 10
command> delete 10
10 found  enter y  to confirm  [y]
list> NULL
command> delete 10
answer> no data
list> NULL
6.command> push 10 //push
list> 10
command> push 20
list> 20 10
7.command> pop      //pop
command> pop
answer>20
list> 10
command> pop
answer>10
list> NULL
command> pop
answer>no data
```

Assignment 4.2 Test Case

```
list> NULL
8.command> [sort][rec][sqrt] //NULL2
answer> no data
list> NULL
command> add  9.5  50.5  20  -5  -20  2
list>9.5  50.5  20  -5  -20  2
command> +                //   50.5 + 9.5
answer> 60
list>60  20  -5  -20  2
command> -                //    20-60
answer> -40
list>-40  -5  -20  2
command> *                //    -5 * -40
answer> 200
list>200  -20  2
command> /                //    -20 / 200
answer> -0.1
list> -0.1  2
command> rec              //   1/x
answer> - 10
list> -10 2
command> neg              //   -x
answer> 10
list> 10 2
command> pow              //    2^10
answer> 1024
list> 1024
command> sqrt
answer> 32
```

```
command> [+] [-] [*] [/] [pow]
answer> can't operation
list> 32
command> pop
answer> 32
list> NULL
9.command> insert 300  //  insert
list> 300
command> insert 100
list> 100  300
command> insert 800 400
list> 100  300   400  800
command> add 200
list> 100  300   400  800 200
command> insert 500
answer> can't insert please sorted before
list> 100  300   400  800 200
10.command> sort          //   sort
list>100  200  300   400  800
command> insert 500
list>100  200  300   400  500  800
11.command> help      //  แสดงชุดคำสั่ง
answer> list  of  command
    .....
    .....
list>100  200  300   400  500  800
12.command> end      //   end จบโปรแกรม
```

Assignment 4.2 (Linked List)

- ทำงานตามชุดคำสั่ง
  - คำสั่งกลุ่มที่ 1
    - กรณีคำสั่ง list, sort, pop, help, end
    - กรณี unary operator เช่น "sqrt","rec","neg"
      - pop ข้อมูลออกจาก Linked List มาเป็น a
      - คำนวณ ans = 1/a (กรณี rec (reciprocal))
      - push ans กลับเข้าไปใน Linked List
    - กรณี binary operator เช่น "+","-","*","/","pow"
      - pop ข้อมูลออกจาก Linked List มาเป็น a
      - pop ข้อมูลออกจาก Linked List มาเป็น b
      - คำนวณ ans = b - a (กรณี - (minus))
      - push ans กลับเข้าไปใน Linked List
  - คำสั่งกลุ่มที่ 2
    - ทำงานตามชุดคำสั่งกับพารามิเตอร์ 1 ตัว cmd[0] + cmd[1]
      - peek (n), search(n), delete(n), push(n),
  - คำสั่งกลุ่มที่ 3
    - วนรอบทำงานตามคำสั่ง กับพารามิเตอร์ทีละตัว cmd[0] + cmd[i]
      - add ( ) , insert( )