ชื่อ ..... รหัสนักศึกษา.....

05506003 Programming Fundamentals

Lab Week 8

วัตถุประสงค์

A. นักศึกษาสามารถประยุกต์ใช้การเขียนโค้ดเพื่อตัดสินใจและแบบวนลูปได้อย่างแม่นยำ

B. นักศึกษาสามารถเขียน method และเรียกใช้ได้

B. นักศึกษาใช้ทักษะในการเขียนโปรแกรมเพื่อแก้โจทย์ที่ซับซ้อน

In class

กิจกรรมที่ 1

Homework

กิจกรรมที่ 2 เขียน ProFun08_Q2_xxyyyy.java

Largest Sum Contiguous SubArray Problem คือ ค่าที่มากที่สุดของผลรวมของ subset ของ array

การใช้กลยุทธ์ brute force คือ หาผลรวมของ ค่า start stop ทุกค่าที่เป็นไปได้ดังนี้

```
public static void main(String[] args) {
  int [] data = {-2,-3,4,-1,-2,1,5,-3};
  q2_1_BF(data);
}
static void q2_1_BF(int ... data) {
  int max = Integer.MIN_VALUE;
  int sum;
  int start, stop;
  start = stop = 0;
  int numCases = 0;
  //all pair of i,j
  for (int i = 0; i < data.length - 1; i++) {
      for (int j = i; j < data.length; j++) {
          sum = 0;
          /* your code */

          printf("case %d for start,stop = %d,%d ->
                          sum = %d",++numCases,i,j,sum);
          if (sum > max) {
              start = i;
              stop = j;
              max = sum;
             println(" ** new max **");
          } else {
             println();
          }
      }
  }
  printf("%d %d = %d%n",start,stop,max);
```

Largest Subarray Sum Problem

-2	-3	4	-1	-2	1	5	-3
0	1	2	3	4	5	6	7

4 + (-1) + (-2) + 1 + 5 = 7

Maximum Contiguous Array Sum is 7

https://www.geeksforgeeks.org/largest-sum-contiguous-subarray/

ชื่อ ..... รหัสนักศึกษา.....

Q2_1 เติม q2_1_BF() ให้สมบูรณ์

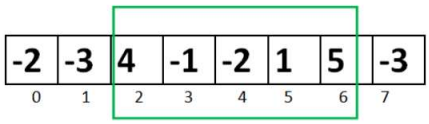
Kadane Algorithm สามารถหาค่าด้วย O(n) โดยข้าม subarray ที่มีค่า sum เป็นลบ และ forward index ทางซ้ายข้าม subarray ที่เพิ่งประมวลผลมา

Kadane's Algorithm สำหรับแก้ปัญหานี้ ระบุไว้ดังนี้

```
// Initialize:
//     max_so_far = INT_MIN
//     max_ending_here = 0

// Loop for each element of the array
//   (a) max_ending_here = max_ending_here + a[i]
//   (b) if(max_so_far < max_ending_here)
//           max_so_far = max_ending_here
//   (c) if(max_ending_here < 0)
//           max_ending_here = 0
// return max_so_far
```

Largest Subarray Sum Problem



4 + (-1) + (-2) + 1 + 5 = 7

Maximum Contiguous Array Sum is 7

https://www.geeksforgeeks.org/largest-sum-contiguous-subarray/

Q2_2 เติม q2_2_kadane() ให้สมบูรณ์

```
/ *10 */  static void q2_2_kadane(int ... data) {
/* 20 */    int start, stop;
/* 30 */    start = stop = 0;
/* 40 */    int max_so_far = Integer.MIN_VALUE;
/* 50 */    int max_ends_here = 0;

/* 60 */    for (int i = 0; i < data.length; i++) {
/* 70 */        max_ends_here += data[i];
/* 80 */        if (      ) {
/* 90 */            /* your code */
/*100 */            /* your code */
/*110 */            printf("%d %d is new max =
                          %d%n",start,stop,max_so_far);
/*100 */        }
/*110 */        if (max_ends_here < 0) {
                    // fast forword i to skip all
                    // subset of ending at j
/*120 */            /* your code */
/*130 */            /* your code */   // ignore negative sum
/*140 */        }
/*150 */    }
/*160 */    printf("%d %d = %d%n",start,stop,max_so_far);
/*170 */  }
```

ชื่อ ..... รหัสนักศึกษา.....

กิจกรรมที่ 3

เขียน static void topK(int ... data) { } โดยมี intArr เป็นอาร์เรย์ 1 มิติ ขนาด 10 ช่อง มีค่าเริ่มต้นของทุกช่องเท่ากับ 0 จากนั้น ให้รับค่ามาจากคีย์บอร์ดทีละ 1 ค่า เป็นจำนวนเต็ม เพื่อใส่ลงในอาร์เรย์ intArr (ปรับเป็นรับจาก int [ ] data) ดังนี้ (ตัวอย่างใส่ 0 ปิดท้ายให้ดังนั้น สามารถตรวจได้ว่าหากพบศูนย์ให้จบการประมวลผล)

1. หากในอาร์เรย์ไม่มีตัวเลขอื่นนอกจาก 0 อยู่เลย และเลขที่รับมาเป็นจำนวนเต็มบวก ให้ใส่ค่าที่ตำแหน่งแรก

2. หากค่าที่รับมาเป็นจำนวนเต็มบวก และค่าในอาร์เรย์มีตัวเลขอื่นนอกจาก 0 ให้นำไปแทรกที่หลังตัวเลขที่น้อยที่สุด ที่มากกว่าค่าที่รับมา หากไม่มี ให้แทรกที่ตำแหน่งแรก

3 หากมีการแทรกค่า ให้เลื่อนข้อมูลในอาร์เรย์ทั้งหมด นับตั้งแต่ตำแหน่งที่ถูกแทรกไปทางขวา 1 ช่อง หากเลขที่ขยับออกไปนอกขอบเขตของอาร์เรย์ให้ถือว่าเลขนั้นหายไป

4. ให้รับค่าเรื่อย ๆ จนกว่าจะมีการใส่ตัวเลขจำนวนเต็มลบ หรือศูนย์ (ข้อมูลที่ป้อนเข้ามา จะมีกี่ตัวก็ได้ จนกว่าจะป้อนข้อมูลจำนวนเต็มลบหรือศูนย์)

5. ให้แสดงอาร์เรย์ปัจจุบัน ใน caller เมื่อมีการแทรกเลขสำเร็จทุกครั้ง และ สุดท้าย ดังตัวอย่าง

insert 5 -> [5, 0, 0, 0, 0, 0, 0, 0, 0, 0]

insert 3 -> [5, 3, 0, 0, 0, 0, 0, 0, 0, 0]

insert 8 -> [8, 5, 3, 0, 0, 0, 0, 0, 0, 0]

insert 4 -> [8, 5, 4, 3, 0, 0, 0, 0, 0, 0]

insert 10 -> [10, 8, 5, 4, 3, 0, 0, 0, 0, 0]

insert 3 -> [10, 8, 5, 4, 3, 3, 0, 0, 0, 0]

insert 1 -> [10, 8, 5, 4, 3, 3, 1, 0, 0, 0]

insert 5 -> [10, 8, 5, 5, 4, 3, 3, 1, 0, 0]

insert 9 -> [10, 9, 8, 5, 5, 4, 3, 3, 1, 0

insert 7 -> [10, 9, 8, 7, 5, 5, 4, 3, 3, 1]

insert 2 -> [10, 9, 8, 7, 5, 5, 4, 3, 3, 2]

final [10, 9, 8, 7, 5, 5, 4, 3, 3, 2]

สำหรับ PEARLS ให้เพิ่ม จำนวน input ไว้เป็นค่าแรก แทนการจบด้วยค่า <= 0 เช่น 11 5 3 8 4 10 3 1 5 9 7 2

คำสั่ง เขียน ProFun08_ xxyyyy.java (ตั้งชื่อไฟล์ให้ถูกต้อง xx = รหัสปี yyyy = รหัส 4 หลักท้าย)