ผลรวมย่อยติดกันมากสุด (Maximum Contiguous Sum)

กำหนดให้มีแถวลำดับ (array) A ของตัวเลขจำนวนเต็มจำนวน n ตัว เราจะเรียก ลำดับย่อยที่อยู่ติดกัน (contiguous subsequence) คือสมาชิกใน A จำนวนไม่ต่ำกว่าหนึ่งตัวที่อยู่ติดกัน ตัวอย่างเช่น สำหรับ A = (a1,a2,a3,...,an) นั้น ลำดับย่อย (a2,a3,a4) จะเป็นลำดับย่อยที่อยู่ติดกัน แต่ (a2,a4) หรือ (a1,a3,a4,a5) จะไม่เป็นลำดับย่อยที่อยู่ติดกัน

จงเขียนโปรแกรมย่อยภาษา C หรือ C++ เพื่อหาว่าลำดับย่อยที่อยู่ติดกันที่ผลรวมของสมาชิกทั้งหมดในลำดับย่อยนั้นมากที่สุด มีผลรวมเป็นเท่าใด โดยให้เขียนคำตอบลงในโครงฟังก์ชันดังต่อไปนี้ และให้ตอบคำตอบโดยการคืนค่า (return) ค่าผลรวมมากที่สุด

```
int MCS(int A[], int n) {
  // A คืออาเรย์ที่มีขนาด n ตัวเลขจำนวนเต็ม
  // ฟังก์ชันนี้ต้องคืนค่าผลรวมของลำดับย่อยที่มากที่สุด
}
```

โปรแกรมของคุณควรจะมี Time Complexity เป็น O(n lg n) (หมายเหตุ: ปัญหาข้อนี้มีขั้นตอนวิธีที่ใช้เวลาเป็น O(n) และนิสิตสามารถใช้วิธีการดังกล่าวได้)

ตัวอย่าง

ตัวอย่างที่ 1

ในกรณีที่ A มีค่าเป็น (-1,2,-3,-4,5,6,-7) และ n มีค่าเป็น 7 นั้น ฟังก์ชันของคุณควรจะคืนค่า 11

ตัวอย่างที่ 2

ในกรณีที่ A มีค่าเป็น (-5,-3,-1,-7,-8,-10) และ n มีค่าเป็น 6 นั้น ฟังก์ชันของคุณควรจะคืนค่า -1

แฟ้มโครงคำตอบ

สำหรับโจทย์ข้อนี้ แฟ้มโครงของคำตอบจะอยู่ในชื่อ MCS.cpp นิสิตสามารถใช้แฟ้มดังกล่าวเป็นจุดเริ่มต้นในการตอบปัญหาข้อนี้ได้ ในแฟ้มดังกล่าวจะมีส่วนของโปรแกรม (main) ที่ทำหน้าที่รับข้อมูลจากแป้นพิมพ์และส่งข้อมูลออกทางจอภาพอยู่แล้ว โดยส่วนของโปรแกรมดังกล่าวจะเรียกใช้ฟังก์ชัน MCS ที่คุณเขียนขึ้น คุณไม่ควรแก้ไขส่วนของโปรแกรมดังกล่าว

ในโปรแกรมย่อย MCS ที่มีให้ในแฟ้มโครงของคำตอบนั้นจะทำการคืนค่า 0 เสมอ นิสิตจะต้องแก้ไขโปรแกรมย่อย findMax ดังกล่าวให้ทำงานตามที่กำหนดไว้ในโจทย์

MCS.cpp

```
#include <stdio.h>
#include <stdlib.h>

int MCS(int A[],int n) {
  return 0;
}


int main(int argc, char **argv) {
  int *A;
  int n;
  scanf("%d",&n);

  // malloc A
  A = (int*)malloc(sizeof(int) * n);

  // read input
  for (int i = 0;i < n;i++) {
    scanf("%d",&A[i]);
  }

  printf("%d\n",MCS(A,n));
}
```

ชอบเขตข้อมูลทดสอบ

40% ของข้อมูลทดสอบนั้น $3 \le n \le 5,000$ (สำหรับข้อมูลชุดนี้ โปรแกรมควรจะใช้เวลาการทำงานไม่เกินกว่า $O(n^2)$)

60% ของข้อมูลทดสอบนั้น $3 \le n \le 50,000$ (สำหรับข้อมูลชุดนี้ โปรแกรมควรจะใช้เวลาการทำงานไม่เกินกว่า $O(n \log n)$)