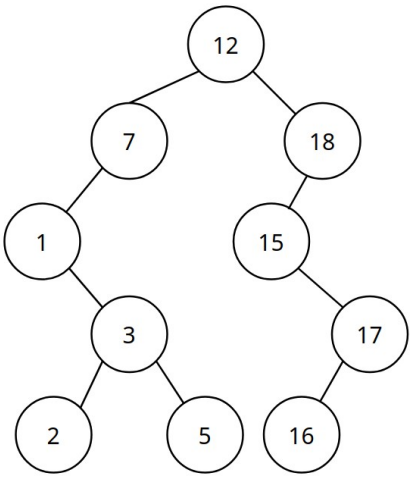
BST Parity Score

(1 sec, 512mb)

ให้เขียนโปรแกรมเพื่อเพิ่มบริการ int CP::map_bst::parity_score(int d) ให้กับ CP::map_bst โดยฟังก์ชันนี้จะทำหน้าที่คืนผลต่างระหว่างจำนวนปมที่มี Key เป็นเลขคี่และจำนวนปมที่มี Key เป็นเลขคู่ และมีความลึกนับจากรากของต้นไม้ไม่เกิน d (ให้รากของต้นไม้มีความลึกเริ่มที่ 0)

ยกตัวอย่างต้นไม้ binary search tree ด้านขวา เลขที่แสดงในแต่ละปมคือ Key ของปมนั้นๆ



- ถ้าเราเรียกใช้ parity_score(2) เราจะต้องได้คำตอบเป็น 1
- ถ้าเราเรียกใช้ parity_score(3) เราจะต้องได้คำตอบเป็น 3
- ถ้าเราเรียกใช้ parity_score(4) เราจะต้องได้คำตอบเป็น 2

รับประกันว่าค่าของทุกปมในข้อนี้จะเป็นข้อมูลประเภท int ที่มีค่าไม่ต่ำกว่า 0 เสมอ

ข้อบังคับ

- โจทย์ข้อนี้จะมีไฟล์ตั้งต้นมาให้ ซึ่งประกอบด้วยไฟล์ map_bst.h, main.cpp และ student.h อยู่ ให้นิสิตเขียน code เพิ่มเติมลงในไฟล์ student.h เท่านั้น และการส่งไฟล์เข้าสู่ระบบ grader ให้ส่งเฉพาะไฟล์ student.h เท่านั้น
- ไฟล์ student.h จะต้องไม่ทำการอ่านเขียนข้อมูลใดๆ ไปยังหน้าจอหรือคีย์บอร์ดหรือไฟล์ใดๆ
- หากใช้ VS Code ให้ทำการ compile ที่ไฟล์ main.cpp

**** main ที่ใช้จริงใน grader นั้นจะต่างจาก main ที่ได้รับในไฟล์โปรเจ็กต์เริ่มต้น แต่จะทำการทดสอบในลักษณะเดียวกัน ****

คำอธิบายฟังก์ชัน main

main() จะอ่านข้อมูล 2 บรรทัดคือ

- บรรทัดแรก ประกอบด้วยจำนวนเต็มสองตัวคือ N และ d แทนจำนวนปมในต้นไม้ และความลึกสูงสุดที่จะพิจารณาในฟังก์ชัน parity_score
  (1 <= N <= 500,000, 0 <= d <= ความลึกของต้นไม้)
- บรรทัดที่สอง ประกอบด้วยจำนวนเต็ม N ตัว แทน Key ของแต่ละปมที่ insert เข้าไปในต้นไม้ตามลำดับ

คำแนะนำ

นิสิตสามารถเขียนโปรแกรมในฟังก์ชัน parity_score_helper(node *current_node, int depth, int &parity_score) ที่ให้ไว้ใน student.h ได้ (ไม่บังคับว่าต้องใช้)

(มีตัวอย่างข้อมูลทดสอบอยู่หน้าถัดไป)

ข้อมูลชุดทดสอบ

- 10% d = 1
- 10% N = 2
- 20% แต่ละปมในต้นไม้จะไม่มีลูกด้านขวา (ปมลูกจะอยู่ด้านซ้ายของปมแม่เสมอ)
- 25% d มีค่าเท่ากับความลึกของปมที่ลึกที่สุดของต้นไม้
- 35% ไม่มีข้อกำหนดพิเศษอื่นใด

ตัวอย่างการทำงานของ main

ข้อมูลนำเข้า	ข้อมูลส่งออก
10 0 12 7 18 1 15 3 17 2 5 16	1
10 3 12 7 18 1 15 3 17 2 5 16	3
10 2 7 4 9 2 5 8 10 1 3 6	1