Heap Level

CP::priority_queue นั้นเป็นโครงสร้างข้อมูลแบบ Binary Heap เราต้องการเพิ่มฟังก์ชันให้กับ Binary Heap นี้ ได้แก่ ฟังก์ชัน std::vector<T> at_level(int k) const ซึ่งจะคืนสมาชิกทั้งหมดใน Binary Heap นี้ที่อยู่ ณ level k ใน Binary Heap เรากำหนดให้ปมใด ๆ ใน Binary Heap นั้นอยู่ ณ level k ก็ต่อเมื่อ ปมดังกล่าวอยู่ห่างจากปมรากเป็นระยะทาง k เส้นเชื่อม จากนิยามดังกล่าว ปมรากนั้นจะถือว่าอยู่ ณ level 0 และ ปมลูกซ้าย และ ลูกขวาของปมรากนั้นจะอยู่ ณ level 1

ค่าที่คืนจากฟังก์ชันนี้จะต้องเป็น std::vector<T> โดยที่ข้อมูลใน vector จะต้องเรียงลำดับตามลำดับที่ข้อมูลดังกล่าวจะถูก pop ออกมาจาก priority_queue ของเรา ตัวอย่างเช่น สำหรับ CP::priority_queue<int> นั้น vector ที่ได้คืนมาจากการเรียก at_level จะมีค่าเรียงจากตัวเลขค่ามากไปยังตัวเลขค่าน้อย

ให้สังเกตว่าฟังก์ชันนี้ถูกประกาศเป็นแบบ const ซึ่งหมายความว่าฟังก์ชันนี้จะต้องไม่ทำการเปลี่ยนแปลงค่าใด ๆ ใน priority_queue ของเรา

รับประกันว่าค่า k จะอยู่ในช่วง 0 ถึง 20 และเป็นไปได้ที่ k จะระบุถึง level ที่ไม่มีข้อมูลใน Binary Heap ของเรา ในกรณีดังกล่าว ให้คืน vector ที่ไม่มีข้อมูล

ข้อบังคับ

- โจทย์ข้อนี้จะมีไฟล์โปรเจ็คของ Code::Blocks ให้ ซึ่งในไฟล์โปรเจ็คดังกล่าวจะมีไฟล์ priority_queue.h, main.cpp และ student.h อยู่ ให้นิสิตเขียน code เพิ่มเติมลงในไฟล์ student.h เท่านั้น และการส่งไฟล์เข้าสู่ระบบ grader ให้ส่งเฉพาะไฟล์ student.h เท่านั้น
  - ในไฟล์ student.h ดังกล่าวจะต้องไม่ทำการอ่านเขียนข้อมูลใด ๆ ไปยังหน้าจอหรือคีย์บอร์ดหรือไฟล์ใด ๆ

คำอธิบายฟังก์ชัน main()

main จะอ่านข้อมูลมาสองบรรทัด ตามรูปแบบนี้

- บรรทัดแรกประกอบด้วยจำนวนเต็ม n และ k
- บรรทัดที่สองประกอบด้วยจำนวนเต็มจำนวน n ตัว ซึ่งจะถูกนำไปใส่ใน priority_queue หลังจากนั้น main จะเรียก at_level(k) แล้วทำการพิมพ์ค่าได้ได้คืนมาออกทางหน้าจอ

ข้อควรระวัง

***** main ที่ใช้จริงใน grader นั้นจะแตกต่างจาก main ที่ได้รับในไฟล์โปรเจ็คเริ่มต้น ****

- ฟังก์ชัน get_kth นั้นจะพิจารณาค่าตาม comparator ที่ใช้ใน priority_queue ด้วย

ตัวอย่างการใช้งาน at_level()

```
int main() {
  int n = 10;
  CP::priority_queue<int> pq;
  for (int i = 0;i < 10;i++) pq.push(i);
  vector<int> v1 = pq.at_level(2) << endl;  // v1 = {7,6,4,1}
  vector<int> v2 = pq.at_level(3) << endl;  // v2 = {3,2,0}
  vector<int> v3 = pq.at_level(20) << endl;  // v3 = {}
}
```