```
//*****************************************************************************************/
//*** Header file for Min-max Heap in Exercise 1 by Wu, Y.H.@CYCU-ICE
//*****************************************************************************************/
#include <cmath>                            // log2, floor
typedef struct hT                           // a type for a heap node
{       int         rid;                    // a serial number as record identifier
        int       value;                    // the key for comparisons
}       heapType;
typedef enum {MIN, MAX} whichHeap;          // a type to distinguish the two parts of a min-max heap
//*****************************************************************************************/
void mmHeapInsert(heapType [], const int, const int, const int);   // add one record
int leftmostHeap(const heapType [], const int);      // locate the leftmost bottom node of a heap
//*****************************************************************************************/
void mmHeapInsert(heapType H[], const int newRid, const int newValue, const int bottom)
{    // a min-max heap, serial number of a new record, key for comparisons on a heap, the bottom node
        int            cur = bottom;                 // start at the bottom node
      int            parent = (cur - 1)/2;           // locate its parent node
      whichHeap    level = ((int)floor(log2(cur + 1)) % 2) ? MAX : MIN;     // Is it at a level of min or max?

        H[cur].rid = newRid;                         // save a new record to the bottom node
        H[cur].value = newValue;
        if (cur > 0)
        {       int grandpa;                         // trickle a new item up to its position
//
//
// Mission Three. Part I.
// Trickle up the new record if it violates the ordering rule of a min-max heap
//
//
        }       // end outer if
}       // end mmHeapInsert

int     leftmostHeap(const heapType H[], const int bottom)     // leftmost bottom node of a heap
{       // a heap, the bottom node
      int idx = 0;
//
//
// Mission Three. Part II.
// Locate the node at the leftmost bottom of a min-max heap
//
//
      return idx;
}       // end leftmostHeap
//*****************************************************************************************/
// Keep the above codes unchanged unless its correctness can be guaranteed.
//*****************************************************************************************/
```