University of Waterloo
CS240 Winter 2023
Programming Question 1

Due Date: Wednesday, February 8 at 5:00pm

Please read `https://student.cs.uwaterloo.ca/~cs240/w23/assignments.phtml#guidelines` for guidelines on submission. Submit the file CS240MaxHeap.cpp to MarkUs.
Late Policy: Assignments are due at 5:00pm, with the grace period until 11:59pm. Assignments submitted after 11:59 on the due date will not be accepted but may be reviewed (by request) for feedback purposes only.

Question 1 [20 marks]

Design and implement `CS240MaxHeap`, which is a modified version of maximum-oriented heap. Unlike the standard maxheap, each node of `CS240MaxHeap` (except possibly the last node) stores exactly k integer keys. An example is in Fig. 1. For any node v, if it has a parent, any key of the parent is larger than or equal to any key stored in node v.

Provide an implementation of the class below, with the corresponding required time complexities, where n is the total number of keys, and k is the number of keys per node. All printing should be done to the standard output.

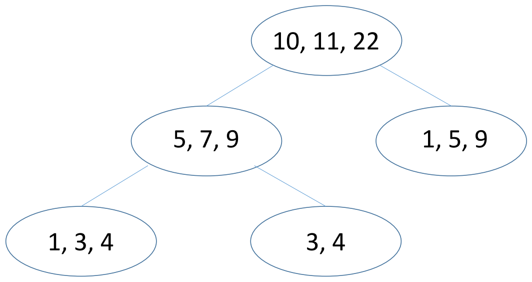

Figure 1: `CS240MaxHeap` with $k = 3$.

```
class CS240MaxHeap{
// add any fields and methods, as necessary
public:
      CS240MaxHeap(const int k); //constructor, complexity O(1)
      void insert(int key);      // inserts new key, complexity O(k log n)
      int delete_max();          // deletes and returns the maximum key
                                 // complexity O(k log n)
      int delete_max_5();  // deletes and returns the 5th maximum key
                         // complexity O(k log n)
      void print();      // prints all items in the CS240 MaxHeap,
                         // one node per line, nodes are printed in
                         // the order of levels from the leftmost to the
                         // rightmost. Keys stored at each node are printed
                         // on the same line in non-decreasing order
                         // complexity O(n)
      printLeftPath();   // prints the leftmost path, complexity O(k log n)
                         // each node is printed on separate line
                         // keys of each node are printed on the same line
                         // in non-decreasing order, complexity O(k log n)
      printRightPath();  // prints the rightmost path, complexity O(k log n)
      int crazy_clean(): // deletes all items in the heap and
                         // computes and returns  ''crazy sum'' of all keys
                         // how to compute this ''crazy sum'' is described
                         // later, complexity O(kn log n)
      int size():        // returns number of keys in the data structure
```

You may not use any pre-existing code that would trivialize implementation (i.e. heaps from STL, etc). You may use C++ vector, stack, and pair data structures. If in doubt, make a private Piazza post and ask.

Place your program in file CS240MaxHeap.cpp. We provide you with a starter code that has the main function that accepts commands from the standard input. You may assume all inputs are valid, i.e. we will never try to delete 5th largest element from a heap with less than 5 items. See the starter code for the description of the commands. You are not allowed to modify the main function.

The function `crazy_clean` works as follows. Let S be the collection of all keys in the heap. This function removes all the keys from the heap, and computes a "crazy sum" which is defined procedurally as follows. The two largest items are removed from the heap, the smallest is subtracted from the largest, and the result is added back to the heap. This continues until there is only one item left in the heap. This item is removed and returned. For the heap in Fig. 1, we first remove 22 and 11, subtract 11 from 22 and get 11, which we insert back in the heap. Then we remove 11 and 10, subtract 10 from 11 and insert 1 into the heap. After 13 iterations, we are left with only one key, namely key 0, which is then returned.

We provide several sample inputs and outputs. Note that in the main function, when printing out the heap contents, and the left or right paths, we first also print the total number of items in the heap and k, the number of items stored at each node.