Lab 5 –Heap & Hash

Problem 1

Given the following class:

```
class Heap {
public:
	Heap(int maxsize);
	~Heap();
	void buildHeap(int *arrIn, int arrInSize);
	bool heapInsert(int dataIn);
	bool heapDelete(int &dataOut);
	int getSize();
	bool isFull();
	bool isEmpty();
	void print(int rootLoc, int level);
private:
	void reheapUp(int childLoc);
	void reheapDown(int rootLoc);

	int* arr;
	int last;
	int maxSize;
};
```

Implement its member functions using the following guideline:

`Heap(int maxsize);`	Create a dynamic array with size maxsize Set last to -1.
`~Heap();`	Clear the array.
`void reheapUp(int childLoc);`	Reheap up operator.
`void reheapDown(int rootLoc);`	Reheap down operator.
`void buildHeap(int *arrIn, int arrInSize);`	Create a new heap from the array arrIn.
`bool heapInsert(int dataIn);`	Insert a new data node into the heap. Return false if heap is full, true otherwise.
`bool heapDelete(int &dataOut);`	Remove the root node of the heap. Return false if heap is empty, true otherwise.
`int getSize();`	Return the size (not max size) of the heap.
`bool isFull();`	Return true if the heap is full, false otherwise.
`bool isEmpty();`	Return true if the heap is empty, false otherwise.
`void print(int rootLoc, int level);`	Print the heap in RNL order.

Test it with the following codes:

```
Heap heap = Heap(256);
int arr[7] = { 9, 3, 55, 16, 32, 88, 1 };
heap.buildHeap(arr, 7);
heap.print(0, 0);
```

Problem 2

Create a priority queue for a heap by using the deleteHeap function. Each time you remove the root, add it to the priority queue. Repeat this until there is no node left. (This whole process is actually the heap sort algorithm).

Problem 3*

Write a function to check if an array with known size is:

a) A min heap using recursion.
b) A min heap using iteration.

Problem 4*

Implement the following hash functions:

$f_1(key) = (7\times digit[0]^0 + 7\times digit[1]^1 + \cdots 7\times digit[end]^{end})\%7719$

$f_2(key) = \left(7\times digit[0]^{digit[0]} + 7\times digit[1]^{digit[1]} + \cdots 7\times digit[end]^{digit[end]}\right)\%7719$

$f_3(key) = digit[end]\ digit[end-1]\ digit[end-2]$

$f_4(key) = fold\ shift\ of\ the\ first\ 9\ digits$

Examples:

$f_1(325) = (7\times 3^0 + 7\times 2^1 + 7\times 5^2)\%7719$

$f_2(325) = (7\times 3^3 + 7\times 2^2 + 7\times 5^5)\%7719$

$f_3(325) = 523$

$f_3(12) = 210$

$f_3(325146) = 641$

$f_4(123456789) = (123 + 456 + 789)\%1000 = 368$

$f_4(325) = (000 + 000 + 325)\%1000 = 325$