COP 3502 Recitation Sheet: Heaps, Hash Tables

Directions: Each of these questions is from either a past Foundation Exam or one of my past exams. They were created to be written on paper. More involved exercises involving coding can be done with these topics. To view these, try to edit the sample code posted for the week.

1) Show the state of a binary heap (min heap) after the insertion of the following items, in this order, into an initially empty binary heap:

13, 2, 19, 16, 14, 3, 9, 12, 6, 2, 18, 11 and 8

2) Show the array representation of the final state of the heap from the previous question.

Index	1	2	3	4	5	6	7	8	9	10	11	12	13
Value													

3) Delete the minimum value from the heap in question 8 and show a picture of the resulting heap.

4) Use the following hash function to insert the given elements into the hash table below. Use **quadratic probing** to resolve any collisions. You may assume that the correct table size (in this case, 10) is always passed to the function with the key that is being hashed.

```
int hash(int key, int table_size)
{
     int a = (key % 100) / 10;
     int b = key % 10;
     return (a + b) % table_size;
}
```

Keys to insert (one by one, in the following order): 2555, 1523, 5893, 800, 956

0	1	2	3	4	5	6	7	8	9

5) There are two hash functions that take in strings as input shown below. Each returns an integer in between 0 and 1,000,002. (Note: 1,000,003 is a prime number.) Which of these two is a better hash function? Explain the weakness in the other function.

```
int f1(char* str) {

    int i = 0, res = 0;
    while (str[i] != '\0') {
        res = (256*res + (int)(str[i]))%1000003;
        i++;
    }
    return res;
}

int f2(char* str) {

    int i = 0, res = 0;
    while (str[i] != '\0') {
        res = (res + (int)(str[i]))%1000003;
        i++;
    }
    return res;
}
```

6) Consider using a hash table of size 16 to store integers, using the linear probing technique to resolve collisions and the following hash function (intended to take non-negative integer input):

```
int f(int n) {
    int res = 0;
    while (n > 0) {
        res = res ^ (n&15);
        n = (n >> 4);
    }
}
```

Show the state of the hash table after executing the following insertions, in this order:

182, 92, 41, 130, 111, 105, 94, and 43.