Algorithm Review Questions: Fall 2023 Midterm

1. Prove by induction or other means

   (a) Prove that the sum of the first n odd numbers is equal to 2^n. For example; $1+3=4=2^2$, $1+3+5=8-2^3$.

   (b) Prove that if a number is divisible by three then the sum of its digits are also divisible by three (For example, 123 is divisible by three and $1+2+3$ is also divisible by three.

   (c) For all integers greater than 3; $2^n < n!$

   (d) For the time equation $T(n) = n^2 + 2n + 1$, the complexity is $O(n^2)$ with $n0 = 3$ and $C = 2$.

2. Compute Complexity from Pseudocode. Assume n is an integer and $n > 0$, $n > k$;

   (a)
```
int sum(int n)
 { int sum=2^k;
for (int i=0; i<n;i++)
 {
   sum=sum/2

   if(sum==1) {break;}
}
return sum; }
```

   (b)
```
int sum(int n)
 { int sum=0;
for (int i=0; i<n;i=i+k)
 {
   for(int j=0; j<i;j++)
   { sum=sum+1; }
}
return sum; }
```

   (c)
```
int sum(int n)
 { int sum=sum(n-1)+1;
   if(n==0)
    {return sum;}
}
```

3. Compute Complexity from formula of T(n).

   (a) $T(n) = T(n/2) + n$

   (b) $T(n) = 4T(n/2) + n^2\sqrt(n)$

   (c) $T(n) = T(n-1) + lgn$

   (d) $T(n) = T(\sqrt(n)) + n^2$

4. **Priority Queues**

   (a) Given the following lists of numbers create a binomial queue for each of them.
   List 1: 7, 10, 8, 35, 34 40
   List 2: 8, 11, 14, 23, 30, 55, 60
   Merge these two queues into a third binomial queue
   Take the combined set of numbers from list 1 and list 2 and create a heap.
   Give one point of difference between a heap and a binomial queue.

   (b) Given a set of integers that are unsorted and coming one after another, at any given point find the median of the set of numbers currently seen. If the data has odd set of values, the median will be the number in the middle of the sorted integers. If the data has even set of numbers, the median will be the average of the two numbers in the middle of the sorted integers.
   For example if the stream of integers is 5, 15, 1, 3; the medians would be;
   Time 1: 5
   Time 2: (5+15)/2=10
   Time 3: 5; middle number of (1 5 15)
   Time 4: 4=(3+5)/2; middle numbers of (1 3 5 15)

   If you use heaps to solve this problem, the complexity will be $O(nlogn)$. [HINT: Check out Computing the Running Median Problem]

5. **Trees**

   (a) Insert the following numbers in order to a binary tree, while keeping it balanced; 1 2 3 4 10 9 8 7 6 5 11 12. Show all the steps for each operation.
   Delete 9 from the tree you created. Show all the steps

   (b) Create a binary tree where the numbers {1,2,3,4,5,6,7,8} are inserted in order.

   (c) What is the minimum number of times that will 8 have to be accessed to make it to the root, using the Splay tree method. Show each step of moving 8 higher up the tree.

   (d) Can a black node have one red child and one black child ? If we relax the condition such that the children of a red node can be either red or black, what are operations necessary to keep the tree balanced ?

   (e) Show the red-black tree created after inserting nodes 1,2,3,4,5,6 into an initially empty tree.

   (f) Create a Red-Black Tree, by inserting the numbers in the given sequence {8,7,5,3,2,6,4,1}.

6. **Hashing**

(a) Use the following hashing functions to place the given list of elements. You will have three hash tables, one for each function.
List = 81,161,6,16,121,56
Hashing Functions: (i is the number of conflicts)
  i. $h(x) = x mod 10 + i$ (Linear Probing)
  ii. $h(x) = x mod 10 + i^2$ (Quadratic Probing)
  iii. $h(x) = x mod 7 + i$ (Linear Probing)
Give the number of conflicts for each function
Create a hashing function for the given set of numbers for which there will be no conflicts

7. **Divide and Conquer**

(a) The following code computes the minimum of value from an array of integers using divide and conquer

```
private static int findmin(int [] a, int left, int right)
 {
 if(left==right)
  return a[left];

//Divide the array into smaller sections
int center = (left+right)/2;
int lsum=findmin(a, left, center);
int rsum=findmin(a, center+1, right)

//Combine the answers
if(lmin<rmin)
return lmin;
else
return rmin;

}
```

  i. Show how this code would execute on the following list of numbers: 4 8 -3 2 -1 -8 6 7.
  ii. Give the formula for T(n), time for this algorithm to run with n elements. You can assume that $n = 2^k$
  iii. Show that $T(n) = O(n)$ (HINT: You might have to use the geometric series)
  iv. Given the k, m and n values find the probability of false positive in a Bloom filter.

(b) Give three examples of recursion that cannot be solved using the Masters theorem formulas, and explain why they cannot be solved. Solve the recursion for **any one** of them

(c) Develop a divide and conquer algorithm to find the most frequently occuring number (mode) in a set of integers. Give the steps and compute the complexity of your methods