1) Consider the implementation of a closed hash table $a[0]..a[n-1]$ to store distinct positive integers, using quadratic probing to resolve collisions. A value of 0 indicates that a hash table location is currently unused. The hash function is $h(x) = x \% n$.

Write a function that searches the table for a given integer $x$. If found, the function returns the index of where $x$ exists in the table. Return -1 if $x$ is not found in the table. The average runtime of your routine should be according to the usual hashing standards.

2) Consider an ordered linked list with $n$ entries in ascending order. Each entry has 2 components: a key component of type int and the usual next link component.
   a) Write a function to insert a new entry with key $x$ into its proper place. Note that a key may be added as the new first or last entry in the list, and so there are $n+1$ locations where $x$ could be inserted.
   b) Assume that each of the $n+1$ possibilities is equally likely. Determine the average number of times ints are compared in the above insertion algorithm. Your answer should be a precise function of $n$. An asymptotic answer (such as one that uses big-oh, big-theta, etc.) is not acceptable.
3) For each function with input argument \( n \), determine the asymptotic number of “fundamental operations” that will be executed. Note that \( f_d \) is recursive. Choose each answer from among the following. You do not need to explain your choices.

\[
\theta(1) \quad \theta(\log n) \quad \theta(n) \quad \theta(n \log n) \quad \theta(n^2) \quad \theta(n^2 \log n) \quad \theta(n^3) \quad \theta(2^n) \quad \theta(n!)
\]

a) 
```c
c
void fa(int n) {
   for(k = 1; k < n; k++)
      for(i = k+1; i <= n; i++)
         for (j = k+1; j <= n; j++)
            Perform 1 fundamental operation;
      //endfor j
   //endfor i
   //endfor k
}
c
```

b) 
```c
c
void fb(int n) {
   for(i = 1; i <= n; i = 2*i)
      Perform 1 fundamental operation;
   //endfor i
}
c
```

c) 
```c
c
void fc(int n) {
   for(i = n; i > 0; i = i-2)
      Perform 1 fundamental operation;
   //endfor i
}
c
```

d) 
```c
c
void fd(int n) {
   if (n > 1) {
      fd(n/3);
      fd(n/3);
      fd(n/3);
      Perform n fundamental operations;
   }//endif
}
c
```