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 \( h(x) = x \mod n \).

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
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
}
```

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

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

d)```c
void fd(int n) {
    if (n > 1) {
        fd(n/3);
        fd(n/3);
        fd(n/3);
        //Perform n fundamental operations;
    }//endif
}
```
Check which problems you are submitting:

☐ #1
☐ #2
☐ #3

How many pages total? ________
Please do not write on the back of any pages.

________________________________________
(print name)
________________________________________
(signature)
________________________________________
(NetId)
Problem #1

a) (4pts) List the four conditions of deadlock:

b) (16pts) Below is a semaphore solution for the producer/consumer problem. The buffer can hold n items. Semaphores are X, Y, and Z.

// The buffer is initialized to be empty and is processed as a first in first out queue

// PRODUCER CODE

while (true)
{
    1. getItem();
    2. wait(X);
    3. wait(Z);
    4. addItemToBuffer();
    5. signal(Z);
    6. signal(Y);
}

// CONSUMER CODE

while(true)
{
    1. wait(Y);
    2. wait(Z);
    3. readItemFromBuffer();
    4. signal(Z);
    5. signal(X);
    6. processItem();
}

There is a problem with each of the semaphore initializations below. Give a sequence of statements showing how an error might occur.

For instance, can the Producer and Consumer be in their critical sections at the same time? Will deadlock occur?
Please note, for full credit, you must you must list a sequence of statements that lead to an error. You will not get credit for guessing.

1) $X = 0, Y = 0, Z = 1$
2) $X = n, Y = 0, Z = 0$
3) $X = n, Y = 0, Z = 2$
4) $X = 0, Y = n, Z = 1$

Problem #2 Resource Allocation Banker’s algorithm

(3pts) What is meant by a "safe state?"

(14pts) Show a safe state process sequence for the following:

Resources: X, Y, Z where available is $X = 12, Y = 6, Z = 7$

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allocated</td>
<td>Max</td>
<td>Available</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>P0</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>P1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>P3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

(3pts) If a request for $P2$ arrives for $(3,2,4)$ can it be granted? Why or why not?
#3 Memory Management Paging

a) (4pts) Given a 3 level page table with a Translation Lookaside Buffer (TLB) hit ratio of 95%, What is the effective access time given that a TLB access is 75ns and a memory access time is 100ns?

b) (4pts) Assume a 32 bit logical address space and 3 level paging system. The first 12 bits are for the 1st level page table, the next 8 bits are for the 2nd level page table, the next 6 bits are for the 3rd level page table and remaining 6 are for the offset. How much virtual memory can be accessed?

c) (12pts) Which page replacement strategy will work best with the following page references assuming there are 4 page frames? FIFO or LRU. Work must be shown for credit. Please show your work.

Reference sequence 1 2 3 4 1 2 5 1 2 3 4 5
Answer ANY TWO of the following three questions:

1. Give a context-free grammar generating the following language over $\Sigma = \{0, 1\}$:

   $$\{0^m1^n0^k : k \geq m; m, n, k \geq 0\}$$

2. A Hamiltonian circuit in an undirected graph is a cycle that visits each node exactly once. A cycle in a graph is a non-empty path in which the only repeated node is the first and last.

   Consider the following problem:

   $\text{HAMCIRCUIT} = \{V, E : G = (V,E) \text{ is an undirected graph containing a Hamiltonian circuit}\}$

   Show that $\text{HAMCIRCUIT} \in \text{NP}$.

3. Answer TRUE or FALSE for each of the following statements to indicate whether the conclusion is always true. If you do not know the answer, do not guess.

   **Scoring**: +2 points for correct answer; 0 point for no answer; -1 point for wrong answer.

   a. If $A \leq_p \overline{B}$ and $B \in \text{co-NP}$, then $A \in \text{NP}$.
   b. If $A \leq_p B$ and $A \in \text{NP-Complete}$, then $B \in \text{NP-hard}$.
   c. If $A \leq_p B$ and $B$ is not decidable, then $A$ is not acceptable.
   d. If $A \leq_p B$ and $B \in \text{P}$, then $A$ is acceptable
   e. If $A \leq_p B$ is $B \in \text{NP}$, then $A \in \text{EXPTIME}$.
   f. If $A \leq_p B$ and $B \in \text{NP-Complete}$, then $A \in \text{P}$.
   g. If $A \leq_p B$ is and $B$ is decidable, then $A$ is decidable.
   h. If $A \leq_p B$ and $B \in \text{NP-Complete}$, then $A \in \text{NP-Complete}$.
   i. If $A \leq B$ and $B$ is co-acceptable, then $A$ is co-acceptable.
   j. If $A \leq B$ and $A$ is not co-acceptable, then $B$ is not decidable.