1) Construct a combinational circuit that has 2 sets of 4-bit integer inputs \((a_3a_2a_1a_0\text{ and }b_3b_2b_1b_0)\), a control input \((x)\), and a 5-bit integer output \((c_4c_3c_2c_1c_0)\). If \(x = 1\), the output \(c\) is to be the difference of the integers \(a - b\), while if \(x = 0\), the output \(c\) is to be the sum of the integers \(a + b\). You may use full adders and up to 4 additional gates. Draw the circuit diagram.

2) Consider the semaphore solution to solve the producer/consumer problem with a buffer of \(n\) elements. Write the basic code for producers and consumers. Declare and initialize all semaphores.

3) Consider the following page replacement algorithms: FIFO (first in first out), LRU (least recently used), OPT (optimal replacement), and 2\(^{nd}\) chance. Logical memory has 10 pages (pages 0 .. 9), while physical memory consists of 4 frames (frames 0 .. 3). The page reference string begins with 5, 3, 8, 4 to fill the four frames. Each part begins from this same initial point.

On your solution page, show the 2 frame traces for each part. For 2\(^{nd}\) chance, also show the reference bit values. Each reference bit value is indicated by a 1 or 0 in parentheses.

a) Continue the page reference string with at most 4 additional terms where LRU will result in strictly fewer page faults than FIFO.

\[
\begin{array}{cccccccc}
\text{ref. str.:} & 5 & 3 & 8 & 4 & \_ & \_ & \_ & \_\\
\hline
5 & 5 & 5 & 5 & LRU \\
3 & 3 & 3 \\
8 & 8 \\
4 \\
\hline
5 & 5 & 5 & 5 & FIFO \\
3 & 3 & 3 \\
8 & 8 \\
4 \\
\end{array}
\]

*** continued on the next page ***
b) Continue the page reference string with at most 3 additional terms where OPT will result in strictly fewer page faults than LRU.

ref. str.: 5 3 8 4 _ _ _

<table>
<thead>
<tr>
<th>5</th>
<th>5</th>
<th>5</th>
<th>3</th>
<th>3</th>
<th>8</th>
<th>8</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5</th>
<th>5</th>
<th>5</th>
<th>3</th>
<th>3</th>
<th>8</th>
<th>8</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


c) Continue the page reference string with at most 5 additional terms where 2\textsuperscript{nd} chance will result in strictly fewer page faults than FIFO.

ref. str.: 5 3 8 4 _ _ _ _ _

<table>
<thead>
<tr>
<th>5 (1)</th>
<th>5 (1)</th>
<th>5 (1)</th>
<th>5 (1)</th>
<th>3 (1)</th>
<th>3 (1)</th>
<th>3 (1)</th>
<th>8 (1)</th>
<th>8 (1)</th>
<th>4 (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2\textsuperscript{nd} chance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5</th>
<th>5</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIFO</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>3</th>
<th>3</th>
<th>8</th>
<th>8</th>
<th>4</th>
</tr>
</thead>
</table>

| 3 | 3 | 3 | 8 | 8 | 4 |
1) Given a (possibly empty) binary search tree of integers, write an iterative function that inserts an integer $x$ into the tree. Do not use recursion.

2) Consider the following quicksort code to sort an array of floats. The algorithm is begun by the call quicksort ($a$, 0, n-1).

```c
void quicksort(float a[], int first, int last)
// Sort a[first]..a[last];  original call: quicksort(a, 0, n-1)
{
    if (first < last) {
        int splitpt = partition(a, first, last);
        quicksort(a, first, splitpt - 1);
        quicksort(a, splitpt + 1, last);
    } //endif
}
```

Write the function partition. The function returns splitpt and modifies $a[first]$ .. $a[last]$ in the following way: Let $x$ be the float value in $a[first]$ at the time partition is called. When partition has completed,
- $a[k] \leq x$ for $k = first, \ldots, splitpt$
- $a[k] \geq x$ for $k = splitpt, \ldots, last$.

3) a) Write an algorithm that returns the median of 3 given integers $a$, $b$, and $c$. The average number of integer comparisons in your algorithm must be less than 3.
   
b) Assuming the 3 inputs are random and distinct, determine the precise average number of comparisons in your algorithm. Justify your answer.
1. Give a state diagram for a deterministic finite automaton that recognizes the following language over $\Sigma = \{0, 1\}$:

$\{w : w \text{ has an even number of occurrences of the substring 01 and } w \text{ has an odd length}\}$

2. Answer each of the following questions with only YES or NO to indicate whether or not the following languages are decidable. Do not guess if unsure, as wrong answers will lower your score!

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

   a. $\{D : D \text{ is a deterministic finite automaton and } L(D) = \emptyset\}$
   b. $\{P : P \text{ is a pushdown automaton and } L(P) = \emptyset\}$
   c. $\{M : M \text{ is a Turing machine and } L(M) = \emptyset\}$
   d. $\{D_1, D_2 : D_1 \text{ and } D_2 \text{ are deterministic finite automata and } L(D_1) = L(D_2)\}$
   e. $\{P_1, P_2 : P_1 \text{ and } P_2 \text{ are pushdown automata and } L(P_1) = L(P_2)\}$
   f. $\{M_1, M_2 : M_1 \text{ and } M_2 \text{ are Turing machines and } L(M_1) = L(M_2)\}$
   g. $\{M : M \text{ is a Turing machine that has a state named } q_{27}\}$
   h. $\{M : M \text{ is a Turing machine with a transition to } q_{27} \text{ in its delta function}\}$
   i. $\{M, w : M \text{ is a Turing machine that enters its state } q_{27} \text{ when run on input string } w\}$
   j. $\{M : M \text{ is a Turing machine that enters its state } q_{27} \text{ when run on any string}\}$

3. A vertex cover of a graph $G = (V, E)$ is $C \subseteq V$ such that every edge $e \in E$ is adjacent to at least one $c \in C$. Let $\text{VERTEX-COVER} = \{V, E, k : G = (V, E) \text{ is a graph that contains a vertex cover of size } k\}$. Show that $\text{VERTEX-COVER}$ is NP-Complete. You may assume the result of the Cook-Levin Theory.