(60 pts) 1. **Short Answers.** For each question, write a short answer, or circle the correct answer.

(a) Consider the recurrence $T(n) = 1$ for $n \leq 2$ and $T(n) = 2T(\sqrt{n}) + 1$ for $n > 2$. This solves to

$$T(n) = \Theta(\text{_______________})$$

(b) Donald Trump claims to have found each of the three-node patterns below in a splay tree. The numbers shown are ranks, and the individual weight of each node is 1. (These patterns are possibly imbedded in a larger tree.) Circle the patterns that are definitely lies.

```
          8      8      8     1     1     1     2
        / \    / \    / \    / \    / \    / \    / \    / \
 8  8 | 7  1 | 6  6 | 8  7 | 9  7 | 0  1 | 1  1 | 2
      \   \  \   \  \   \  \   \  \   \  \   \  \   \
        0  0  1  2
```

(c) Suppose you have the option of running one of three algorithms to solve a given problem. On an input of size $n$ (assume $n$ is a power of 3 and $T(1) = 1$):

- Algorithm $A$ breaks it into 3 pieces of size $n/3$, recursively solves each piece, and then combines the solutions in time $n^2$.
- Algorithm $B$ breaks it into 4 pieces of size $n/3$, recursively solves each piece, and then combines the solutions in time $n$.
- Algorithm $C$ breaks it into 9 pieces of size $n/3$, recursively solves each piece, and then combines the solutions in time $O(n^2)$.

The fastest algorithm is algorithm _____ with running time $\Theta(\text{__________})$.

The slowest algorithm is algorithm _____ with running time $\Theta(\text{__________})$. 
(d) The splay tree below contains keys \{1, 2, 3, 4, 5\} in in-order. Inside each node write the key that is stored there.

Also, draw the tree that results from splaying the node labeled \(x\).

(e) A splay tree \(T\) contains \(n\) keys numbered 1, \ldots, \(n\). A new node containing key \(n + 1\) is added by making it the root of the tree. Its left child is the old tree \(T\), and it has no right child. All the keys have weight 1. What is the amortized cost of this update to the tree.

\[\Theta(1) \quad \Theta(\log \log n) \quad \Theta(\log n) \quad \Theta(\sqrt{n}) \quad \Theta(n) \quad [\text{Can’t tell from the info. given.}]\]

(f) You are given an algorithm \(A\) where the cost of operations is equal 1, 2, 1, 4, 1, 2, 1, 8, \ldots — i.e., the cost of the \(i^{th}\) operation is the largest power of 2 that divides \(i\). What is the amortized cost of each operation performed by \(A\)?

\[\Theta(1) \quad \Theta(\log n) \quad \Theta(\sqrt{n}) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2)\]

(g) For computing a maximum matching in a bipartite graph using network flow, the length of the augmenting paths is always odd.

\(T\) \quad \(F\)

(h) Let \(G\) be a bipartite graph with \(n\) vertices and \(m\) edges, which has a maximum matching of size \(k\). Set this up as a flow problem in the usual way. Now run Dinic’s algorithm on it. The first blocking flow it finds will have flow at least \(\underline{\hspace{2cm}}\), and the algorithm will take \(O(\underline{\hspace{2cm}})\) time to find it. (The latter blank should contain the tightest bound that is correct.)

(i) Circle the algorithms that are polynomial time algorithms for network flow:

\[\text{Dinic} \quad \text{Ford-Fulkerson} \quad \text{simplex} \quad \text{ellipsoid}\]

\[\text{interior point} \quad \text{Seidel} \quad \text{Edmonds-Karp #2}\]
(j) In the flow network below, all the edges are directed from left to right, and all the capacities are one.

![Flow Network Image]

What is the maximum flow in this network? 

Highlight the edges of some blocking flow in this network that is not a maximum flow.

(k) Let \( P(n, d) \) be the number of properly parenthesized expressions of length \( n \), where the maximum depth of nesting is at most \( d \). You will write a dynamic program to compute this number.

**Example:** The string \( ()()()() \) has length 8 and depth 1, \( ()((()))() \) has length 10 depth 2, and \( ()(((())())) \) has length 16 and depth 4.

So \( P(2, 1) = 1 \) since only \( () \) has length 2 and depth \( \leq 1 \). \( P(4, 1) = 1 \) since only \( ()() \) has length 4 and depth \( \leq 1 \). \( P(4, 2) = 2 \) since \( ()() \) and \( ((()) \) have length 4 and depth \( \leq 2 \). \( P(6, 2) = 4 \) since \( ()()(), ()((())) \), \( ((())(()) \) and \( ()()()) \) have length 6 and depth \( \leq 2 \).

i. Fill in the blanks in the following recurrence for \( P(n, d) \) (where \( n \geq 0 \) and \( d \geq 0 \)):

\[
P(n, d) = \begin{cases} 
0 & \text{if } n \text{ is odd} \\
0 & \text{if } n > 0 \text{ and } d = 0 \\
\sum & \text{if } n = 0 \\
\text{otherwise} 
\end{cases}
\]

ii. The runtime to compute \( P(n, d) \) using dynamic programming is \( O(\_\_\_\_\_\_) \).
(l) Let $Y$ be a random variable on $[0,1]$ with probability density function (PDF) $f(y) = 4y^3$. Write a function $g(x)$ such that if $x$ is sampled randomly from Uniform$[0,1]$ then $g(x)$ will generate a correctly distributed sample from $Y$.

$$g(x) = \ldots$$

(m) Let $X$ and $Y$ be two IID random variables with PDF 
$$f(x) = \beta e^{-\beta x}$$

where $\beta$ is some positive constant.  
Let $Z = \|X - Y\|$. Write the PDF for $Z$.

What is the expected value of $Z$?

(n) State the strong diameter property of the ExpDelay($G, \beta$) ball growing algorithm.

(o)
(20 pts) 2. **Counting Modulo** \( m \)

A counter starts at 0, and keeps track of the number of increments done to it modulo \( m \). Let \( x \) represent the value of the counter.

\[ \text{inc}(): \text{This replaces } x \text{ by } (x + 1) \mod m. \text{ The cost is the absolute value of the difference between the old value of } x \text{ and the new value of } x. \]

(a) Let \( \Phi(x) = x \) be a potential function. Prove using this potential function that the amortized cost of \( \text{inc}() \) is at most 2.

(b) Now, continuing with part (a) and using the same potential function, prove that, starting from \( x = 0 \), the total cost of \( k \) \( \text{inc}() \) operations is at most \( 2k \).

It turns out that the bound of 2 from part (a) is not optimal. In the next two parts you will find and prove the minimum amortized cost of \( \text{inc}() \).

(c) Fill in the blank below defining a new potential function:

\[ \Phi(x) = \text{__________} \]

(d) With this potential, the amortized cost of an \( \text{inc}() \) is at most

\[ \text{__________}. \]

Prove this in the space below.
3. **Mildly Disordered**

A sorted array $A[1], A[2], \ldots, A[n]$ ($n \geq 4$) of distinct elements is randomly permuted in the following way.

For each element $i$ (in increasing order from 1 up to $n - 3$) $A[i]$ is swapped with $A[i + r]$, where $r$ is randomly chosen (on each step) from $\{0, 1, 2, 3\}$.

Your job is to develop an efficient comparison-based sorting algorithm for such arrays.

(a) As a function of $n$, how many possible distinct permutations of $A[]$ could be generated? (This is not a big-oh question.)

(b) Prove your answer to part (a)

(c) Your solution from part (a) implies a lower bound on the number of comparisons required in the worst cast to sort $A[]$. What is that lower bound? (This is not a big-oh question.)

(d) Give a comparison-based algorithm that sorts such permuted arrays using at most $3(n - 3)$ comparisons.
(20 pts) 3. **Electric Charging Stations.**

You have to pick a subset $S$ of $\{0, 1, \ldots, n\}$ such that

(a) Both $0 \in S$ and $n \in S$, and

(b) Two consecutive points in the set differ by at most $D$.

Such a subset is called $(n, D)$-good. So, for example, if $n = 10$ and $D = 5$ then $\{0, 5, 10\}$ is $(10, 5)$-good, and so is $\{0, 3, 7, 10\}$ but $\{0, 6, 10\}$ is not.

Given a vector of costs $c_1, \ldots, c_n$, let the cost of a subset be

$$ C(S) = \sum_{i \in S} c_i. $$

Given $n$, $D$ and the cost vector $c_1, \ldots, c_n$, the goal is to find the cost of the $(n, D)$-good subset with minimum cost.

In order to do so, let $M(i) =$ the cost of the minimum cost $(i, D)$-good subset. Fill in the blanks in the following recurrence to make it correct.

$$ M(i) = \begin{cases} 
\infty & \text{if } i < 0 \\
0 & \text{if } i = 0 \\
\min_{0 \leq j \leq i} \{ M(j) + \text{[ ]} \} & \text{if } 0 < i \leq n 
\end{cases} $$

Using dynamic programming, what is the running time needed to compute $M(n)$ as a function of $n$ and $D$?

$$ O(\text{[ ]}) $$

What does this problem have to do with “Electric Charging Stations”? 
3. **Using \( k \) Agents**

Consider an \( n \) by \( n \) grid of vertices. The rows and columns are numbered 1 through \( n \). The vertex in the \( i \)th row and the \( j \)th column is labeled \((i, j)\). On every vertex \((i, j)\) there are \( M_{i,j} \geq 0 \) tokens.

There are \( k \) agents who start out at position \((1, 1)\) and walk through the grid, ending at \((n, n)\). Each step of the path moves to the next higher row or the next higher column (i.e. the move is down or to the right). An agent will collect all the tokens she finds along the path. A token can only be collected by one agent. The problem is to compute the maximum number of tokens that can be collected by all the agents.

(a) The problem with \( k \) agents can be solved using min-cost max-flow. We will give you the flow graph without capacities and costs, and you will have to fill these in to make the construction work.

For each \((i, j)\) with \( 1 \leq i, j \leq n \) we create two vertices \( a(i, j) \) and \( b(i, j) \). We put an edge from \( a(i, j) \) to \( b(i, j) \). We also add edges from \( a(i, j) \) to \( a(i + 1, j) \) and \( a(i, j + 1) \). We also add edges from \( b(i, j) \) to \( a(i + 1, j) \) and \( a(i, j + 1) \). (Vertices out of bounds, such as \( a(1, n + 1) \) don’t exist and edges to them are not created.) Finally we also have a vertex \( s \) and a vertex \( t \), and there’s an edge from \( s \) to \( a(1, 1) \) and an edge from \( a(n, n) \) and \( b(n, n) \) to \( t \). The figure below shows a \( 2 \times 2 \) grid on the left and the graph described in this paragraph on the right.

Now we find the min-cost max-flow from \( s \) to \( t \). Your problem is to figure out how much cost and capacity each of the edges we created has to have in order for this to work. Fill in the following table (next page).
<table>
<thead>
<tr>
<th>Edge</th>
<th>Capacity</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s \rightarrow a(1,1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a(i, j) \rightarrow b(i, j)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a(i, j) \rightarrow a(i+1, j)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a(i, j) \rightarrow a(i, j+1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b(i, j) \rightarrow a(i+1, j)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b(i, j) \rightarrow a(i, j+1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a(n, n) \rightarrow t$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b(n, n) \rightarrow t$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Let $C$ be the minimum cost of a maximum flow from $s$ to $t$. What is the maximum total value of all the tokens that can be collected by agents in terms of $C$?
3. (20 pts) **Not on a cycle.**

Let $G = (V, E)$ be a connected undirected graph with $n$ vertices and $m$ edges. We want to mark every vertex that is on some cycle. In other words, only vertices that are not on a cycle will not be marked.

(a) Complete the following modification of DFS to mark the vertices in such a way. Assume `Visited` is an array of length $|V|$ indexed by vertices with each entry initialized to `False` and `Depth` is an array of length $|V|$ indexed by vertices initialized to 0.

```c
int DFS(Vertex *v, int d)
{
    Depth[v] = d
    Visited[u] = True
    int Min = d+1
    for (every neighbor u of v)
    {
        if __________________________
        {
            if Depth[u] <= Depth[v]-1 && Depth[u] < Min
            {
                Min = Depth[u]
            }
        }
        else
        {
            Min = min(Min, DFS(u, d+1))
        }
    }
    if __________________________
    {
        mark v to indicate it’s on a cycle
    }
    return Min
}
```

(b) Briefly argue why your algorithm is correct and has running time $O(n + m)$.

(c) Describe another way to compute the same quantity in $O(n + m)$ time making use of an algorithm (as a black box) described in this course.