1. Short Answer Questions (32 pts)

For (a) and (b), assume the base-case $T(1) = 1$ and assume $n$ is a power of 4.

(a) The recurrence $T(n) = 6T(n/4) + n^2$ solves to (circle one):

\[ \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(n^2 \log n) \quad \Theta(n \log_4 6) \]

(b) The recurrence $T(n) = 6T(n/2) + 1$ solves to (circle one):

\[ \Theta(1) \quad \Theta(n^2) \quad \Theta(n^3) \quad \Theta(n \log_2 6) \]

(c) Merging 4 sorted lists, of $n$ elements each, takes time: (pick the fastest possible using a comparison-based algorithm)

\[ \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(2^n) \]

(d) Merging $n$ sorted lists, of 4 elements each, takes time: (pick the fastest possible using a comparison-based algorithm)

\[ \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(2^n) \]

(e) The summation $1^3 + 2^3 + 3^3 + 4^3 + \ldots + n^3$ solves to

\[ \Theta(n^3) \quad \Theta(n^3 \log n) \quad \Theta(n^4) \quad \Theta(2^n) \]

(f) Suppose we have a graph with $n$ nodes and $m$ edges. If we randomly color each node red, green, or blue (each color with probability $1/3$), then the expected number of edges whose endpoints have different colors is:

\[ m/9 \quad m/3 \quad m/2 \quad 2m/3 \quad m \]
(g) Draw a treap containing the following (key priority) pairs: (a 6), (b 8), (c 2), (d 4), (e 3), (f 6).

(h) Hash functions $f_1$ and $f_2$ map elements $a, b, c, d$ to locations 0 and 1 as follows:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$f_2$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Is $\{f_1, f_2\}$ a universal family of hash functions? Briefly explain why or why not.
2. Dynamic Programming (24 pts)

(a) Car-talk chauffeur Picov Andropov hates driving on congested roads. Picov has a map of the city represented as a weighted graph. Each edge $e$ in the graph is labeled with a weight $\text{cong}(e)$ representing how congested that street is. We say that the congestion of a path is the most congested edge along that path. For example, in the graph below, the least-congested path from $A$ to $D$ is $A-C-D$ with congestion 3.

```
A 5 B
|   |
2  |
C--|---D
3  1
```

Suppose Picov wants to know for every pair of nodes $(u, v)$, the value of the least-congested path from $u$ to $v$ in the graph. Show how we can modify the Floyd-Warshall Dynamic Programming algorithm to compute this. Circle the correct choice in each of the three underlined \{ \ldots \} below.

```
//A[i][j] = congestion of best i->j path that can only go through 1..k.
Initialize: A[i][i] = 0 for all i.
           A[i][j] = cong(i,j) if there is an edge i->j
           A[i][j] = \{0, 1, infinity\} otherwise

for k = 1 to n do:
  for each i,j do:
    // you either go through k or you don't.
    A[i][j] = \{ min, max \} \{ A[i][j], \{ min, max, sum \} (A[i][k], A[k][j]) \};
```

(b) Suppose you are a consultant. Your inbox has a collection of tasks $a_1, a_2, \ldots, a_n$ to do. Each task $a_i$ has a time-to-complete $t_i$, a deadline $d_i$, and a payment $p_i$. If task $a_i$ is completed by time $d_i$, you collect $p_i$ dollars (no payment if it is completed after its deadline). Assume that all quantities are positive integers, and let us say we have sorted the tasks by deadline so that $d_1 \leq d_2 \leq \ldots \leq d_n$.

Fill in the blanks in the following dynamic-programming algorithm (given in top-down or “memoized recursion” form) to calculate the maximum profit you can make. You would call this routine with $k = n$ and $t = d_n$. Assume the matrix $\text{AlreadyComputed}$ is initialized to all $-1$’s.
maxprofit(k, t) // max profit for items 1..k, if we must end by time t.
{
    if (t < 0) return -infinity;
    if (k == 0) return 0;
    if (AlreadyComputed[k, t] != -1) return AlreadyComputed[k, t]; // done
    t = min(t, d_k); // can’t do anything after time d_k
    profit =
    max(maxprofit(k, t), // profit if don’t do task k.
        maxprofit(k, t) // profit if do task k.
    );
    AlreadyComputed[k, t] = profit;
    return profit;
}

(c) The running time of this algorithm as a function of n and d_n is Θ(______).

3. Truth or counterexample (24 pts). For each statement below, indicate whether it
is true or false. If true, give a short proof. If false, give a counterexample.

(a) The shortest path tree from a given node S cannot have more than twice the
weight (sum of edge lengths) of the minimum spanning tree.
(b) If you run Prim’s MST algorithm from some start node \( s \), for \( k \) steps, then this produces the minimum-weight \( k \)-edge tree out of all \( k \)-edge trees containing \( s \).

(c) For any given permutation \( P \) of the numbers \( \{1, 2, \ldots, n\} \), let \( T_P \) be the binary search tree you get by inserting those numbers into a BST in that order using simple BST insertion. This mapping \( P \rightarrow T_P \) is 1-1: that is, if \( P \neq P' \) then \( T_P \neq T_{P'} \).
4. Amortized analysis (20 pts).

Suppose we had code lying around for implementing a stack, and we now wanted to implement a queue. One way to do this is to use two stacks $S_1$ and $S_2$. To insert into our queue, we push into stack $S_1$. To remove from our queue we first check if $S_2$ is empty and if so we dump $S_1$ into $S_2$ (we repeatedly pop the top off $S_1$ and push it onto $S_2$ until $S_1$ is empty). Then we pop from $S_2$. Here is pseudocode:

- insert($x$): $S_1$.push($x$)
- remove(): If $S_2$ is empty then dump($S_1, S_2$). Return $S_2$.pop().
- dump($S_1, S_2$): while $S_1$ is not empty do $S_2$.push($S_1$.pop()).

Let’s say that each push costs $1$ and each pop costs $1$, and performing a dump when $S_1$ has $n$ elements costs $2n$ dollars (since we do $n$ pushes and $n$ pops).

(a) Suppose that (starting from an empty queue) we insert 3 elements, then remove 1 of them. What is the total cost of these 4 operations?

(b) Circle the smallest correct answer. Suppose we perform $n$ operations (inserts and removes) starting from an empty queue. Then the amortized cost per operation is at most:

$\$1 \quad \$2 \quad \$3 \quad \$4 \quad \$\lg(n) \quad \$n \quad \$2n$

(c) Give a proof of your answer in (b) using the bank/potential-function method. Specifically,

- For an insert, the actual cost is $1$, plus we put _______ in the bank, for a total out-of-pocket cost of _______.
- So, the out-of-pocket cost for a dump is _______.
- This means that overall, the amortized cost per insert is at most _______ and the amortized cost per remove is at most _______. The worst of these is _______, which is the answer circled above.

(d) Formally, the potential function used above is: