1. What polynomial is computed by the following straight-line program. Rewrite the program to compute the same final result using Horner’s method. How many lines do you save?

\[
\begin{align*}
    s_1 &= x \times x \\
    s_2 &= s_1 \times x \\
    s_3 &= s_2 \times x \\
    s_4 &= 5 \times s_3 \\
    s_5 &= 2 \times s_1 \\
    s_6 &= s_5 - s_4 \\
    s_7 &= 3 \times s_2 \\
    s_8 &= 4 \times x \\
    s_9 &= s_8 - s_7 \\
    s_{10} &= s_6 - s_9
\end{align*}
\]

2. Using the master theorem, determine the complexity of the solutions to the following recurrences. Show enough work so that I can follow your reasoning.

   a) \( T(n) = 1 \) for \( n < 2 \); \( T(n) = 3T(n/3) + 5n^2 \) for \( n \geq 2 \);
   b) \( T(n) = 1 \) for \( n < 2 \); \( T(n) = 5T(n/3) + 3n \) for \( n \geq 2 \);
   c) \( T(n) = 1 \) for \( n < 2 \); \( T(n) = 4T(n/2) + 5n^2 + 3n \) for \( n \geq 2 \);

3. The polynomial

\[
p(z) = 12z^{15} - 9z^{14} + 6z^{13} - 2z^{12} + z^{11} - 4z^{10} + 7z^9 - 3z^8 + 3z^7 + 5z^6 - 7z^5 - z^4 + 2z^3 + 7z^2 - 4z + 13
\]

is evaluated at the 16th roots of unity by the Discrete Fourier Transform. Identify the polynomials of degree less than 15 that are evaluated at the complex number \( i \) in the process.

4. In the following adjacency matrices, \( a[i][j] \) is 1 when there is an edge from vertex \( v[i] \) to \( v[j] \). Identify the matrices which contain no directed cycles, and give a topological sort for those graphs. In which of these are the vertices \( v[0], \ldots, v[5] \) already topologically sorted? Explain how one could decide by inspection that an adjacency matrix represents a topologically-sorted directed graph.

\[
\begin{align*}
    a) & \begin{bmatrix}
    0 & 1 & 1 & 0 & 0 & 1 \\
    1 & 0 & 1 & 1 & 0 & 1 \\
    1 & 1 & 0 & 0 & 0 & 1 \\
    0 & 1 & 0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1 & 0 & 1 \\
    1 & 1 & 1 & 0 & 1 & 0
    \end{bmatrix} & b) & \begin{bmatrix}
    0 & 1 & 1 & 0 & 0 & 1 \\
    0 & 0 & 1 & 1 & 0 & 1 \\
    0 & 0 & 0 & 0 & 0 & 1 \\
    0 & 0 & 0 & 0 & 0 & 1 \\
    0 & 0 & 0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 & 0 & 0
    \end{bmatrix}
\end{align*}
\]
5. Consider the graph $G$ with the following weights:

Use either Prim’s algorithm or Kruskal’s algorithm to determine a minimum spanning tree for $G$. (If you use Prim’s, use $F$ as the root.) Show your work clearly.

6. Suppose you have a connected graph with 12 edges of weight 1 and 12 edges of weight 2. True or false; the tree produced by Dijkstra’s shortest path algorithm on the graph must contain at least one edge of weight 1. Explain.

7. Illustrate either the 2-3-4 tree or the red-black tree that results from using the respective methods discussed in class to produce a random tree for the input HAZYQUESTION. Use the following page to show any intermediate steps that you want considered for partial credit.

8. Consider the following graph given by adjacency lists on the handout. ($\Lambda$ represents a null pointer.)

Determine the order the nodes are visited by the Depth First Search, and Breadth First search, starting at vertex $A$. Using the relevant ordering (DFS or BFS), what is the backvalue of $B$? Identify any articulation nodes.

9. Consider the network depicted on the left with source $s = A$, sink $t = I$, and capacities of edges as labeled. At right, the values of a valid flow through the network are shown.
a) Determine the value of $z$ in the flow function on the right.
b) Demonstrate the Ford-Fulkerson algorithm by augmenting the flow at least twice.
c) Find the maximum flow through the network.

10. Consider the following algorithm to search a sorted array for a specified key. It is assumed that the array is declared as char a[N] where N=k*k (Valid indices for a[] are 0 to N-1)

```c
int seq_search(char a[], int left, int right, char key)
// a modified sequential search to check only a part of the array
{
    int j=left;
    while( (j < right) && (a[j] < key) )
        j++;
    if (a[j] == key)
        return j;
    else
        return -1;
}

int square_search(char a[], int k, char key)
{
    int i=0;
    while( (i<k) && (a[i*i] <= key) )
        i++;
    if (i>0)
        return seq_search(a, (i-1)*(i-1), i*i-1, key);
    else
        return -1;
}
```

a) Briefly explain why square_search works.
b) As a function of N, what is the worst-case comparisons using key for this algorithm.
c) Compare the complexity with that of binary search.