1. Fill in the blanks.
   (a) In ____________________ hashing, no two data have the same hash value.
   (b) Using ____________________ coding, the codons for the different symbols of a message may be written consecutively without spaces.

2. Give the asymptotic time complexity in terms of \( n \), using \( \Theta \), \( O \), or \( \Omega \), whichever is most appropriate.
   (a) \( F(n) \geq F(n - \sqrt{n}) + n^2 \)
   (b) \( H(n) < H(n/3) + H(n/4) + 2H(n/5) + n \)
   (c) \( G(n) = 3(G(2n/3) + G(n/3)) + 5n^2 \)

3. For each of these recursive subprograms, write a recurrence for the time complexity, then solve that recurrence.
   (a) void george(int n)
   
   {if (n > 0)
    {for(int i = 0; i < n; i++) cout << "hello" << endl;
     george(n/2); george(n/3); george(n/6);}}

   (b) void martha(int n)
   
   {if (n > 1)
    {martha(n-1); martha(n-2);}}
   
   Hint: Look at problem 0.3 on page 9 of your textbook.
4. Walk through Kruskal’s algorithm to find the minimum spanning tree of the weighted graph shown below. Show the evolution of the union/find structure at several intermediate steps. Whenever there is choice between two edges of equal weight, choose the edge which has the alphabetically largest vertex. Whenever there is a union of two trees of equal weight, choose the alphabetically larger root to be the root of the combined tree. Indicate path compression when it occurs.
5. A hash table with size \( m = 200 \) is used to store 400 data items, using a pseudo-random hash function. The average number of items in a cell (“bucket”) is clearly \( \frac{400}{200} = 2 \), but there could be empty cells. What is the expected number of empty cells? Approximate to the nearest integer. Hint: You may need to look in a statistics textbook, or on the internet, to figure this out.

6. Insert the letters B, A, Y, H, P, D into an empty treap, where the “random” keys are given in the following table. Show the treap after each insertion and indicate all rotations.

\[
\begin{array}{|c|c|}
\hline
\text{A} & 8 \\
\text{B} & 4 \\
\text{D} & 10 \\
\text{H} & 2 \\
\text{P} & 5 \\
\text{Y} & 9 \\
\hline
\end{array}
\]
7. Find an optimal Huffman code on the alphabet A,B,C,D,E,F where frequencies are given in the following table.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>6</td>
<td>3</td>
<td>11</td>
<td>14</td>
<td>2</td>
</tr>
</tbody>
</table>

8. A 3-dimensional $9 \times 7 \times 10$ rectangular array $A$ is stored in main memory in row major order, and its base address is 8192. Each item of $A$ takes one word of main memory, that is, one addressed location. Find the address of $A[4][5][2]$. 
9. You wish to compute all entries of Pascal’s triangle down to the 10th row, namely \( C(n, k) = \binom{n}{k} \) using dynamic program and the recurrence \( C(n, k) = C(n - 1, k - 1) + C(n - 1, k) \) for \( 0 < k < n \), and \( C(n, 0) = C(n, n) \) for all \( n \). To save space, you want to store Pascal’s triangle as a triangular array. \( C(n, k) \) will be stored as \( X[\text{location}] \), where \text{location} \) is a function of \( n \) and \( k \). Here is your code, with one line deleted:

```cpp
int const N = 10;
int X[N*(N+1)/2]; // The size of X is N+1 choose 2, which is 55 if N = 10

int location(int n, int k)
{
    assert(0 <= k and k <= n);
    return // FILL IN HERE
}

void store(int value, int n, int k)
{
    X[location(n,k)] = Value;
}

int fetch(int n, int k)
{
    return X[location(n,k)];
}

int main()
{
    for(int n = 0; n <= N; n++)
        for(int k = 0; k <= n; k++)
            if(k == 0 or k == n) store(1,n,k);
            else store(fetch(n-1,k-1)+fetch(n-1,k),n,k);
    return 1;
    // Write out the triangle
    for(int n = 0; n <= N; n++)
    {
        for(int k = 0; k <= n; k++)
            cout << " " << fetch(n,k);
        cout << endl;
    }
    return 1;
}
```

The only missing part is the return value of the function \text{location}. Fill it in.
10. Consider the following program, where n is a given constant. The purpose of the program is to find the longest increasing subsequence of a sequence.

```c
int A[n];
int B[n];
void getA()
{
    for(int i = 0; i < n; i++) cin >> A[i];
}

int main()
{
    getA();
    int s = 0;
    int i = 0;
    while(i < n) // beginning of outer loop
    {
        while(s > 0 and B[s-1] > A[i]) s--; // inner loop
        B[s] = A[i];
        s++;
        i++;
    } // end of outer loop
    for(int j = 0; j < s; j++) cout << B[j];
    cout << endl;
    return 1;
}
```

(a) If n = 10 and the input stream is 0 6 9 8 1 3 2 5 4 7 what is the output?

(b) The inner and outer loops are both linearly bounded, and thus the time complexity of the code is $O(n^2)$. But, it is not $\Theta(n^2)$. Use amortization to prove that the time complexity is $\Theta(n)$.

I haven’t gone over amortization, and besides, my program is incorrect, so this problem will not be graded.

11. You are trying to construct a cuckoo hash table of size 8, where each of the 8 names listed below has the two possible hash values indicated in the array. Using Hall’s marriage theorem, prove that you will fail to construct the table.

<table>
<thead>
<tr>
<th>Name</th>
<th>h1</th>
<th>h2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ann</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Bob</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Cal</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Dan</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Eve</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Fay</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Gus</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Hal</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>
12. Give the asymptotic complexity of $F(n)$ for each of these recurrences, using $\Theta$.

(a) $F(n) = 2F(n - 1) + 1$

(b) $F(n) = 2F(n - 1) + n$

(c) $F(n) = 2F(n - 1) + n^2$

13. Explain how to implement a very sparse one-dimensional array using a search structure.