Fill in the blank (1 point each blank):

1. __________________ probing is simple to implement, but typically results in primary ______________________, which causes performance to degrade rapidly.

2. __________________ probing still causes secondary clustering.

3. Record deletion is straightforward when ____________________________ is used to resolve collisions. _______________ deletion must be used when using a method based on open addressing.

4. When the load factor associated with a hash table reaches its critical value, ________________ is used to build a new and bigger table.

Multiple Choice (3 points each)

1. A d-ary heap is like a binary heap except for each node has d children. The following is a d-ary min heap where d=4. Assuming node 2 is at location zero, what is the formula for the subscript of the first child of a node:

(a) 4i+1       (b) 4(i-1)     (c) 4i    (d) 4(i+1)     (e) None of the previous

2. One reason a min binary heap (stored as an array) implementation of a priority queue is preferable to an AVL implementation is that the binary heap implementation provides a better worst-case run time guarantee for which of the priority queue operations?

a. insert       
b. deleteMin    
c. findMin      
d. none of the above
3. What is the primary reason to use a binomial queue instead of a binary heap?
   a. Faster average-case delete max
   b. Faster insert
   c. Less space usage
   d. Faster merge
   e. Faster worst-case delete max

4. What is the primary reason to use double hashing instead of linear probing?
   a. Faster insert when the hash table is getting full
   b. Find is more accurate
   c. Easier to support delete
   d. Easier to expand the hash table when it gets full

**Short Answer**

1. (5 points) Here is an array of ten integers:
   
   5 3 8 9 1 7 0 2 6 4
   
   Draw this array after the FIRST iteration of the large loop in an insertion sort (sorting from smallest to largest).

2. (5 points) Give a concise accurate description of a good way for quicksort to choose a pivot element. Your approach should be better than "use the entry at location [low]".

3. (5 points) Show the steps to sort 9, 8, 7, 6, 5, 4, 3, 2, 1 using Shellsort with increments {7, 3, 1}. 
4. (10 points) We want to do double hashing where the first hash function is \((5 \times x) \mod 13\). The personalized step function (or second hash function) is \(1 + (x \mod 11)\). Some items have already been added to the hash table. Show what happens if you add (in order) 2, 4, and 1. If there are collisions, show the entire probe sequence tried.

5. (6 points) Recall that when merging binomial queues, there is not a unique correct answer. Showing all your work, give one of the possible queues which result from performing deleteMin on the following priority queue:
6. (6 points) Show the results of merging the following two leftist heaps

```
3
  13
   14
   15
  6
   19
  22
  55
```

```
8
  11
   20
```

7. (6 points) Show the results of merging the following two skew heaps.

```
2
  6
  5
```

```
3
  4
  7
  8
  9
```
8. (10 points) Match each of the given sorting algorithms with one or more of the given characteristics. Write as many letters as apply in the blank to the left of the name of the sorting method.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>insertion sort</td>
<td>A. linear extra space</td>
</tr>
<tr>
<td>shellsort</td>
<td>B. optimal average running time</td>
</tr>
<tr>
<td>heapsort</td>
<td>C. linear best-case running time</td>
</tr>
<tr>
<td>quicksort</td>
<td>D. quadratic worst-case running time</td>
</tr>
<tr>
<td>mergesort</td>
<td>E. linear for nearly-ordered files</td>
</tr>
<tr>
<td></td>
<td>F. stable</td>
</tr>
<tr>
<td></td>
<td>G. adaptive</td>
</tr>
</tbody>
</table>

4. (Minimum Average - 15 points).
Write a recursive method that has three parameters – a reference to an array of type double, and the two indices lo and hi that indicate what subarray of that array we are dealing with in this method call. You can assume the subarray is of size greater than or equal to 3. You want to return the minimum average of three consecutive values in the array – that is, calculate the average of every three consecutive values, and return the minimum of those averages.

```java
double minAverage(double[] arr, int lo, int hi) {
    // your code goes here
```
5. (14 points) You have two binomial queues, each stored in an array (like we did for the last programming assignment). Write mergeSame(Node * q1, Node * q2) which merges two trees of the same size.

Class Node
{
    int val;
    Node * leftChild;
    Node * rightSib;
}

Class Queue
{
    Node * forrest[];
    int maxSubUsed;  // largest subscript of forest which has been used.
    ...
}

Node * mergeSame(Node * q1, Node * q2){