1. Give the major advantage of any hashed data structure over array-based and linked list structures?

*The address is computed from the key. This can greatly reduce the number of memory access operations.*

2. What is the potential downside to hashed data structures compared to array-based and linked list structures?

*There can be a lot of overhead.*

3. What characteristic common to all hashing access algorithms makes them fast in the key field mode?

*The address is calculated from the data.*

4. For a hashed data structure implemented in Java, give the two alternatives for what is stored in the primary storage area array?

*The first uses the Java API Hashtable class. This class defines a map between a set of keys and a set of data. The data are stored in the main array. The user does specify the hash function used. The structure is dynamic and unencapsulated. It resolves collisions dynamically by expanding memory. The nodes are stored in the primary array.*

*The Second uses what is essentially an array of linked-lists. The calculation of the address leads to the head of a linked list. Collisions extend the lists. Headers to linked lists are stored in the primary array.*

5. Define loading factor.

*The loading factor describes the fraction of memory which is actually in use. For some hashing algorithms, having a smaller loading factor can decrease the possibility of collisions.*

6. Define the term collision in the context of data structures.

*In hashed structures, collisions occur when the first calculation of the address from the key leads to an address which contains the wrong node.*

7. True or false, collisions improve the performance of a hashed data structure?

False, *Collisions always slow down a process since we have to wait for memory-access time. What happens subsequent to a collision is often a sequential search of a few more nodes.*

8. Keys are numeric values between 0 and 10,000. Assuming 200 nodes are to be stored in a hashed data structure that uses the direct hashing function, how many elements will be in the primary storage area array?

*There are 200 elements in the primary array.*

9. Assuming the Direct Hashing function and the Subtraction preprocessing algorithm is used to map keys into indices, give the index it maps the key 2000 into, assuming:

a) The range of the keys is 0 to 999,999.

2000

b) The range of the keys is 100 to 999,999.

1900

10. Give the Division Hashing function and the index it maps the key 2000 into, assuming a primary storage area array size of 61 elements and:

a) The range of the keys is 0 to 999,999.

48 = 2000 mod(61)

b) The range of the keys is 100 to 999,999.

*One could choose 48 as above or 9 = 1900 mod 61. It doesn't actually matter, as long as one is consistent about how much is subtracted in the hashing algorithm. The number of potential keys is so much larger than the prime 61 that there will be many collisions if there are many nodes. It would be better to choose a much larger prime. Of course that needs to be balanced against how many nodes one expects.*

11. Nodes are to be stored in a hashed data structure that utilizes the direct hashing function. Assuming the key field was an integer ranging from 2000 to 100,000 and the structure will store a maximum of 60,000 nodes:

*N = 98001, n = 60000.*

a) Compute the loading factor of the structure.

Lf*= n/N = 0.61*

b) Compute the density of the structure assuming a node width of 100 bytes. w=100

D = 1/(1+4/(Lfw)) = 0.938

c) Give the node width that results in a density of 0.7.

Inverting the formula gives: w = 4D/Lf/(1-D) = 15.2

12. A key maps into index 200 of a Direct Hashed data structure's primary storage array. A Fetch operation is to be performed. How will we tell if the node is not in the structure?

*The contents of the primary storage array at 200 will be null.*

13. What would be the density and maximum loading factor of the structure described in Exercise 11 if 9000 nodes were to be stored in the structure?

*The maximum loading factor is 0.092 and the Density is 0.697.*

14. Which data structure is faster: a perfect or a nonperfect hashed structure?

*Perfect hashed structures don't have to provide for collisions and therefore tend to be faster.*

15. Define search length.

*The search length is the number of memory accesses required to find the node.*

16. What is the average search length of a perfect hashed data structure?

*The perfect hashed data structure always has a search length of 1.0.*

17. Considering density and speed, what is the optimum loading factor for a hashed data structure that uses a nonperfect hashing function, and what is the average search length at that loading factor?

*According to the text. The optimal Loading factor is 0.75 which has an average search length of 4-5. This is a complicated calculation.*

18. Which of the following integers are 4k + 3 primes?

a) 4726 Clearly not prime.

b) 2003 This is a prime. 2003=4\*500+3

c) 9109 Not = 4n+3. (It is prime, however.)