

**FACULTY OF ENGINEERING**

**COMPUTER SYSTEMS ENGINEERING DEPARTMENT**

**Data Structures & Algorithms**

**HOMEWORK #4**

**Prepared by:**

**Rana AbuMazen 1070437**

**Instructor: Dr. Diab Abu Aidah**

**Sample Data Structures Questions
Searching**

* **Short Answers:**
1. Draw a hash table with open addressing and a size of 9. Use the hash function "k%9". Insert the keys: 5, 29, 20, 0, 27 and 18 into your table (in that order).

|  |
| --- |
| 0 0 |
| 1  27 |
| 2  29 |
| 3 20 |
| 4 18 |
| 5 5 |
| 6 |
| 7 |
| 8 |
| 9 |

1. Suppose you are building an open address hash table with double hashing. The hash table capacity is n, so that the valid hash table indexes range from 0 to n. Fill in the blanks:
	1. In order to ensure that every array position is examined, the value returned by the second hash function must be 1+ (K mod n – 1) with respect to n.
	2. One way to ensure this good behavior is to make n be prime, and have the return value of the second hash function range from 1 to 1+ (n mod n-1) (including the end points).
2. You are writing code for the remove member function of a chained hash table. Fill in the blanks in this pseudocode with the two missing statements. You may use pseudocode yourself, or write actual C++ code:

 void Table::remove(int key)

 {

 Node cursor;

 size\_t i;

 1. i = hash(key);

 2. Make cursor point to the node that contains an item with the

 given key(or set it to NULL if there is no such node).

 3. if (cursor != NULL)

 {

 (Define a new node p = NULL)

 3a. p = cursor;

 3b. cursor = cursor -> next;

1. Draw a hash table with chaining and a size of 9. Use the hash function "k%9" to insert the keys 5, 29, 20, 0, and 18 into your table.

|  |
| --- |
| 0  000 |
| 1   272727 |
| 2   29 |
| 3  18 |
| 4  20 |
| 5  |
| 6 5 |
| 7 |
| 8 |
| 9 |

1. Suppose that I have the following record\_type definition for a record in a hash table:

 struct record\_type

 {

 int key;

 ... other stuff may also appear here ...

 };

The hash table uses open addressing with linear probing. The table size is a global constant called CAPACITY. Locations of the table that have NEVER been used will contain the key -1. Locations of the table that were once used but are now vacant will contain the key -2. All valid keys will be non-negative, and the hash function is:

size\_t hash(int key)

{

 return (key % CAPACITY);

 }

Complete the implementation of the following function. There is no need to check the precondition, but your code must be as efficient as possible.

// Precondition: data[0]...data[CAPACITY-1] is an open address hash table

// as described above.

// Postcondition: If search\_key occurs as a key of a record in the table, then

// the function returns true; otherwise the function returns false.

bool key\_occurs (const record\_type data [ ], int search\_key){

 boolean [ ] ex = new boolean[CAPACITY];

 int hash = (int) Math.abs(key % CAPACITY);

 for (; ex[hash]; hash = (hash + 1) % CAPACIITY)

 if (search\_key [hash] == key)

 return true;

 return false;

}

1. Suppose that an open-address hash table has a capacity of 811 and it contains 81 elements. What is the table's load factor? (An appoximation is fine.)
* Load Factor A = 81 / 811 = 0.0999
1. I plan to put 1000 items in a hash table, and I want the average number of accesses in a successful search to be about 2.0.
	1. About how big should the array be if I use open addressing with linear probing? NOTE: For a load factor of A, the average number of accesses is generally ½(1+ 1/ (1-A)).

2 = 0.5 (1 + 1/1-A) 🡪 A = 2/3

A = 1000 / size 🡪 Size = 1500

* 1. About how big should the array be if I use chained hashing? NOTE: For a load factor of A, the average number of accesses is generally (1+A/2).

2 = 1 + A/2 🡪 A = 2

A = 1000 / size 🡪 Size = 500

* **Multiple Choice:**
1. What is the worst-case time for serial search finding a single item in an array?
	* A. Constant time
	* B. Logarithmic time
	* C. Linear time
	* D. Quadratic time
2. What is the worst-case time for binary search finding a single item in an array?
	* A. Constant time
	* B. Logarithmic time
	* C. Linear time
	* D. Quadratic time
3. What additional requirement is placed on an array, so that binary search may be used to locate an entry?
	* A. The array elements must form a heap.
	* B. The array must have at least 2 entries.
	* C. The array must be sorted.
	* D. The array's size must be a power of two.
4. What is the best definition of a collision in a hash table?
	* A. Two entries are identical except for their keys.
	* B. Two entries with different data have the exact same key.
	* C. Two entries with different keys have the same exact hash value.
	* D. Two entries with the exact same key have different hash values.
5. Which guideline is NOT suggested from from empirical or theoretical studies of hash tables:
	* A. Hash table size should be the product of two primes.
	* B. Hash table size should be the upper of a pair of twin primes.
	* C. Hash table size should have the form 4K+3 for some K.
	* D. Hash table size should not be too near a power of two.
6. In an open-address hash table there is a difference between those spots which have never been used and those spots which have previously been used but no longer contain an item. Which function has a better implementation because of this difference?
	* A. insert
	* B. is\_present
	* C. remove
	* D. size
	* E. Two or more of the above functions a
7. What kind of initialization needs to be done for an open-address hash table?
	* A. None.
	* B. The key at each array location must be initialized.
	* C. The head pointer of each chain must be set to NULL.
	* D. Both B and C must be carried out.
8. What kind of initialization needs to be done for a chained hash table?
	* A. None.
	* B. The key at each array location must be initialized.
	* C. The head pointer of each chain must be set to NULL.
	* D. Both B and C must be carried out.
9. A chained hash table has an array size of 512. What is the maximum number of entries that can be placed in the table?
	* A. 256
	* B. 511
	* C. 512
	* D. 1024
	* E. There is no maximum.
10. Suppose you place m items in a hash table with an array size of s. What is the correct formula for the load factor?
	* A. s + m
	* B. s - m
	* C. m - s
	* D. m \* s
	* E. m / s