



Informatics II

Exercise 10

April 29, 2020

Hash Tables, Hash Functions

Task 1. The hash table $T[0 \dots 10]$ stores integers and uses chaining to resolve conflicts. Illustrate the hash tables after the values 5, 19, 27, 15, 30, 34, 26, 12, and 21 have been inserted (in that order) using the following hash functions:

1. $h_1(k) = (k + 7) \bmod 11$ (**Division Method**)
2. $h_2(k) = \lfloor 8(k \cdot 0.618 \bmod 1) \rfloor$ (**Multiplication Method**)

Task 2. The hash table $T[0 \dots 10]$ may contain up to 11 elements and uses open addressing. Draw an image of the hash table after the keys 5, 19, 27, 15, 30, 34, 26, 12, and 21 have been inserted (in that order) using the following hash functions:

1. $h_1(k, i) = (k + i) \bmod 11$
2. $h_2(k, i) = (k \bmod 11 + i(k \bmod 7)) \bmod 11$

For each table indicate the longest probe sequence with the corresponding key.

Task 3. Write a C-program where a hash table consisting of m slots of positive integers is implemented. **Double hashing** should be used to resolve conflicts. The hash-function to be used is:

$$\begin{aligned}h(k, i) &= (h_1(k) + i * h_2(k)) \bmod m \\h_1(k) &= (k \bmod m) + 1 \\h_2(k) &= m' - (k \bmod m') \\m' &= m - 1\end{aligned}$$

Your program should include the following:

- The constant m corresponding to the size of the hash table.
- The function `void init(int A[])` that fills all slots of the hash table A with the value 0.



- The hashing function *int h(int k, int i)* that receives the key *k* and the probe number *i* and returns the hashed key.
- The function *void insert(int A[], int key)* that inserts *key* into the hash table *A*.
- The function *int search(int A[], int key)* that returns -1 if the requested *key* was not found in the hash table *A*. Otherwise, it should return the index of the requested hash key in the table.
- The function *void printHash(int A[])* that prints the table size and all non-empty slots of the hash table *A* accompanied with their index and the key.

Hashtable		Output printHash
0	1315	Table size: 7
1	2002	i: 0 key: 1315
2	2001	i: 1 key: 2002
3	2000	i: 2 key: 2001
4		i: 3 key: 2000
5	1313	i: 5 key: 1313
6	1314	i: 6 key: 1314

Mathematical functions available in the library `math.h` can be used and don't need to be redefined. For testing purposes, use a table size of 7, add the values 1313, 1314, 1315, 2000, 2001 and 2002 and print your hashtable. Search for the values 2000, 10, 1314, 1313 and 337 and print the results.

Task 4. Write a C-program where a hash table consisting of *m* slots of positive integers is implemented. **Chaining** should be used for collision resolution. The hash function to be used is $h(k) = \lfloor m(kA \bmod 1) \rfloor$ where $A = (\sqrt{7} - 1)/2$ and $kA \bmod 1$ returns the fractional part of kA . i.e. $kA - \lfloor kA \rfloor$. Each slot is implemented as a linked list to resolve conflicts. A node of a hash table slot is defined as follows:

```
1 struct element {
2     int val;
3     struct element *next;
4 };
```

The global hash table *H* is defined, as follows:

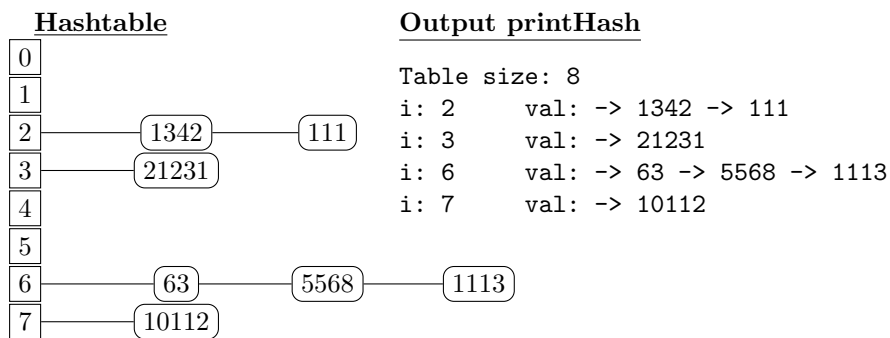
```
1 struct element* H[m];
```

Implement the following functions:

- *void init()*, which initializes hash table *H*.



- *int h(int val)*, which gets a value *val* as input and returns the hashed key.
- *struct element* search(int val)*, which returns NULL if the requested *val* is not found in hash table *H*. Otherwise, it returns a pointer to the node with the requested *val*.
- *void insert(int val)*, which inserts value *val* into hash table *H*, if it does not already exist in the table (no duplicates should exist).
- *void printHash()*, which prints the values of all non-empty slots of the hash table *H*.



Mathematical functions that are available via library `math.h` can be used and don't need to be redefined. For testing purposes, use a hash table of size 8, add the values 111, 10112, 1113, 5568, 63, 1342 and 21231, and print your hash table. Search for values 1, 10112, 1113, 5568, 337 and print the results.