ALGORITHMS & DATA STRUCTURES

Lahcen `ouarbya

TOPIC 4: HASHING
WORKING SESSION 2

In this working session you have to implement the Insert and LoadFactor functions of a hash table **using Linear Probing to deal with collisions**. The function Insert is in charge of inserting a new element in the hash table. The function LoadFactor returns the load factor of the hash table.

Learning Objectives:

1. **Implement the Insert** function of a hash table operating with Linear Probing
2. **Implement the LoadFactor** function of a hash table operating with Linear Probing

PART 1: Insert

This function allows inserting a new positive integer number into the hash table. It receives 5 input arguments: the hash table, the number to be inserted (k) and the values a, c and m corresponding to the coefficients a, c and m of the hash function:

$$h(k)=(a*k+c) \bmod m$$

Coefficients a, c and m are positive integer numbers and m is the number of buckets of the hash table. If the position where the number must be inserted is already used, the function applies **linear probing**.

PART 2: LoadFactor

This function receives as input argument the hash table and returns as a float the fraction of buckets occupied in the hash table.

Test your program with the following:

- For a=31, c=37 and m=50, what is the load factor of the hash table (initially empty) after calling the function insert 30 times? In the i-th call, the value to insert is given by $2*i^2+3i-5$. That is, the first value to insert is 0, the second value is 9 and so on.

  The answer to this question is 0.6. The content of the hash table after doing the 30 insertions described above is as follows:

```
[ -1     -1     -1     -1     -1     -1     -1     270    1320    -1
  -1     -1     225    1425   -1     -1     9      -1     -1      22
  372    114    85     1122   1764   1885   319    490    940     1219
```

```
 -1     624   774   -1    -1     -1    429   0      1029     -1
 -1     184   555   855   147   697    39    60      1534    1647]
```

Please, use this example and the next one to check that your code is working ok.

- For a=17, c=23 and m=30, and the Fibonacci series as the set of numbers to be inserted in the hash table, when does the first collision occur?

  The answer to this question is: the first collision occurs when the third element of the Fibonacci series (number 1) is inserted in the hash table. For you to check your program, the following information might be also useful: the second collision occurs when number 144 (13th element in the Fibonacci series) is inserted in the hash table.

  After making 30 insertions in this hash table, its content and load factor are the following:

```
[ 4181   34       987    2584   13    17711    89      75025    6765    8
    1     1       144     377    3     610      10946  121393   5        196418
   21    317811  1597      0    233   28657    514229    2      55       46368]
```

  Load factor: 1.0