Assignment 5

1. We wish to implement a dictionary by using direct addressing on a huge array. At the start, the array entries may contain garbage, and initializing the entire array is impractical because of its size. Describe a scheme for implementing a direct-address dictionary on a huge array. Each stored object should use O(1) space; the operations SEARCH, INSERT, and DELETE should take O(1) time each; and the initialization of the data structure should take O(1) time. (Hint: Use an additional stack, whose size is the number of keys actually stored in the dictionary, to help determine whether a given entry in the huge array is valid or not.)

Initialization: Huge array = A,
stack = S, and stack is empty
k = key which contains the index of the element, A[k]
size of stack = number of keys actually stored in dictionary
Valid entry: S[1……S.top].,
set S.top=0

If key k is stored in array A then A[k] contains the index i, which is a valid entry in S and S[i]=k has the value k. So the record is valid when 0 < A[k] <= S.top and S[A[k]] = k.

Search: searching for element x with a key (k) in the dictionary using direct addressing.
If (0 < A[k] <= S.top and S[A[k]] = k)
Condition True, element is present in the dictionary.

Else element is not found.

Insertion: Insert an element **x** with key k.
Assumption: x is not in the dictionary.
Insert x as the top element of the stack.
S.top++; //
S.top = x; // push the element to the top of the stack.
A[k] = S.top;
This operation insert the element in O(1).

Delete: Delete an element **x** with key k.
Searching for the element x in the array A with index k.
If found delete the element (x) and fix the gap in the A by arranging the other elements.

To delete an element first copy the element on the top of the stack
A[k] = x; //location of element to be deleted
S.top = A[k]; // copy the element to the top of the stack
S.top = NIL; // set the value of the top as NULL
A[k] = NIL; // deleting the element from original array A
S.top -- ; // Decrement the top.

2. Consider a hash table of size m = 1000 and a corresponding hash function:

$$h(k) = \lfloor m(kA \bmod 1) \rfloor, \quad A = \frac{\sqrt{5}-1}{2}$$

Compute the locations to which the keys 61, 62, 63, 64, and 65 are mapped. **(30 pts)**

h(61) = floor (1000 * (61* (sqrt 5 -1/2) mod 1))
= floor (1000 * ((61*0.6180339887) mod 1))
= floor (1000 * (37.7 mod 1))
= floor (1000 * (.7000))
= floor (700)
= 700

h(62) = floor (1000 * (62* (sqrt 5 -1/2) mod 1))
= floor (1000 * ((62*0.6180339887) mod 1))
= floor (1000 * (38.318 mod 1))
= floor (1000 * (.3181))
= floor (318.1)
= 318

h(63) = floor (1000 * (63* (sqrt 5 -1/2) mod 1))
= floor (1000 * ((63*0.6180339887) mod 1))
= floor (1000 * (38.936 mod 1))
= floor (1000 * (.93614))
= floor (936.14)
= 936

h(64) = floor (1000 * (64* (sqrt 5 -1/2) mod 1))
= floor (1000 * ((64*0.6180339887) mod 1))
= floor (1000 * (39.554 mod 1))
= floor (1000 * (.55417))
= floor (554.17)
= 554

h(65) = floor (1000 * (65* (sqrt 5 -1/2) mod 1))
= floor (1000 * ((65*0.6180339887) mod 1))
= floor (1000 * (40.1722 mod 1))
= floor (1000 * (.172209))

$= \text{floor}\ (172.209)$
$= 172$