1. The *hexadecimal* system represents numbers in the base 16. The fifteen digits in this system are `'0','1',..,'9','A', 'B',..,'F'`, where `'A'` stands for the value 10, `'B'` for 11, ..., `'F'` stands for the value 15. Write a Python function `hexa` which computes the number whose hexadecimal representation is the given string (the letters in the given string are guaranteed to be one of `'0',..,'9','A',..,'F'`).

   ```
   hexa "9" = 9
   hexa "11" = 17
   hexa "1A" = 26
   hexa "90" = 144
   hexa "111" = 273
   ```

2. By adapting the binary search function described in class, write a function `bleft` which takes a value v and a sorted list l and returns the largest i such that $l[i] < v$. If there is no such i the function returns -1. Your function should only make logarithmic number of comparisons (like the binary search function).

   ```
   bleft(3,[1,3,3,3,3,4,4,8]) = 0
   bleft(3,[4,5,5,8]) = -1
   bleft(3,[3,5,5,8]) = -1
   ```

3. Write a function `quickCount` which takes a value v and a sorted list l and returns the number of occurrences of v in l. [Hint: How do you use the previous problem to propose a fast solution to this problem?]

   ```
   quickCount(3,[1,3,3,3,3,4,4,8]) = 4
   quickCount(2,[1,3,3,3,3,4,4,8]) = 0
   ```

4. In this problem the aim is to check if a given list of list of integers represents a possible solution to a sudoku puzzle.

   (a) Write a python function `validrow` that takes as input a list of integers and verifies that it is some permutation of $\{1, 2, \ldots, 9\}$.

   (b) Write a python function `transpose` that takes as input a matrix given as a list of list of integers computes the transpose of the matrix.

   (c) Write a python function `blocksTorows` that takes a list with 3 elements each of which is a list with 9 elements and does the following: Let the elements of the first list be $a_1 ... a_9$, that of the second list $b_1 ... b_9$ and the third list be $c_1 ... c_9$. It returns a list with 3 elements. The first element is $[a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3]$, the second is $[a_4, a_5, a_6, b_4, b_5, b_6, c_4, c_5, c_6]$ and the last element consists of the remaining elements.

   (d) Write a function `validsolution` which takes a matrix of integers with 9 elements each of which is a list of 9 integers and verifies that it satisfies the requirements of the solution to a Sudoku puzzle.

5. Write a function `isvalid` that takes as argument a string and returns `'Yes'` if the string is valid and returns `'No'` otherwise. A string is valid if the string has the following properties:

   - the entire string should be made of only alphabets, numbers, dots and the symbol @
   - the string contains exactly one @
   - the string contains one or more dots
   - the substring before @ should contain only alphabets and numbers (dots are not allowed)
   - the string after the last dot should be of length ≥ 1 and ≤ 3

   For example:

   ```
   >>> isvalid('xyz@abc@de.ef')
   'No'
   >>> isvalid('xyz123@abc.de.fg')
   'Yes'
   ```

6. A higher order function is one that takes another function as input. For example

   ```
   def is_zero(func, value):
       if func(value) == 0:
           return True
       else:
           return False
   ```

   Here `is_zero` takes a function `func` as argument and returns True if the result of that function application is 0.

   Write a higher order function `mymap` that takes another function `f` and a list `l`. `mymap` applies `f` to every element of `l` and returns the list of results. For example: Given the following function `double`:

   ```
   def double(i):
       return i*2
   ```

   the following is the expected output:

   ```
   >>> mymap(double, [1,2,3,4])
   [2,4,6,8]
   >>> mymap(isvalid, ['xyz@abc@de.ef','xyz123@abc.de.fg'])
   ['No','Yes']
   ```

   where `isvalid` is the function defined in the previous question.

   `map(double, [1,2,3,4])` should return `[2,4,6,8]`