To keep it short, DOC Strings are removed.

Problem 4: Closest Power

```
def closest_power(base, num):
    from math import log
    exp = (int)(log(num, base))    # truncated exponent

    if (abs(base**exp - num) <= abs(base**(exp + 1) - num)):
        return (exponent)
    else:
        return (exponent + 1)
```

Problem 5: Dot Product

```
def dotProduct(listA, listB):
    return sum(list(map(product, listA, listB)))
def product(a, b):
    return a * b
```

Problem 6: Deep Reverse

```
def deep_reverse(L):
    for element in L:
        if (isinstance(element, list)):
            element.reverse()
    L.reverse()
    return L
```

Problem 7: Intersect-Difference

```
def dict_interdiff(d1, d2):
    intersection_dict = {}
    difference_dict = {}

    common_keys = []
    for key in d1.keys():
        if key in d2.keys():
            common_keys.append(key)

    for key in common_keys:
        intersection_dict[key] = f(d1[key], d2[key])

    for key in d1.keys():
        if key not in common_keys:
            difference_dict[key] = d1[key]

    for key in d2.keys():
        if key not in common_keys:
            difference_dict[key] = d2[key]

    return (intersection_dict, difference_dict)
```

Problem 8: Apply F, Filter G

```
def applyF_filterG(L, f, g):
    temp_list = L[:]
    L.clear()

    for i in temp_list:
        if (g(f(i)) == True):     # ‘== True’ is redundant
            L.append(i)

    if (len(L) == 0):
        return -1
    else:
        return max(L)
```

Problem 9: Flatten List

```
def flatten(aList):
    non_list = []         # element which isn’t list
    flatten_list = []     # element which is list, flatten it

    for element in aList:
        if (isinstance(element, list)):
            flatten_list.extend(flatten(element))
        else:
            non_list.append(element)

    return non_list + flatten_list
```