(i) NO, the two functions will not produce the same result. In update_listA, the function directly iterates over the elements of the list. This approach would lead to an infinite loop for any list containing an even element. This happens because the function inserts an element with twice the value at each even index, causing the list to expand continuously. As a result, the iteration repeats indefinitely due to the added elements, resulting in an unending loop.

(ii) Tup1 = ((), {}, [], 5)
1) Error
2) ((), {1: [5]}, [], 5)
3) Tup1(0) is a tuple. Append does not work for tuple.
4) ((), {}, [5], 5)
5) ()
6) It shows an error as a tuple doesn't allow assignment operation.

(iii) B and D will produce the same output because both refer to the last element of the list.

(iv)

```
def rotate_list(lst, k):
    if not lst:
        return lst
    k = k % len(lst)
    return lst[k:] + lst[:k]
```

a. It has too many unnecessary checks for k
b. Modulus operation has been used incorrectly

(v)
Output:
a) [[[[[]]]]]
b) ((((()))))
NO, the output will not remain the same if you change the second parameter. The second parameter, k, controls how often the data is wrapped inside lists or tuples. A different value of k will result in a different nesting level of lists or tuples.

(vi)
{1: True, 2: False, 3: 'b'}
{2: False, 3: 'b', 1: True}

(vii)
a)assert(sortList([6, 8, 2,1])) == [1, 2, 6, 8]
b)assert (sortList([1, 2, 3, 4, 5,6,7])) == [1, 2, 3, 4, 5,6,7]
c) in line 8 while j >= 0 and key < a[j]:
(viii)
a) assert expr_conv("a+b*c-d^e") == "abc*+de^-"
b) assert expr_conv("(a+b)*c-(d/(e+f)^g)") == "ab+c*def+g^/-"
c) assert expr_conv("a*(b+c)-(d/e^f)+g") == "abc+*de/f^*-g+"