ASSIGNMENT 23

ANSWER 1:

- Result of the code is : `1 2 8`
- When the function was defined arguments b and c were given default values as 6 and 8 respectively. In the function call `func(1, 2)`; argument `a` will be assigned 1 and b will be assigned 2 but c will be assigned its default value, which is 8. Therefore the output will be 1 2 8.

ANSWER 2:

- Result of the code is : `1 2 3`
- When the function was called `func(1, c=3, b=2)`, clearly b was assigned 2 and c was assigned value 3.
- NOTE: Note that by using the keyword arguments, we can also switch around the order of the parameters and still get the same result when we execute the function.

ANSWER 3:

- Result of the code is : `1 (2, 3)`
- The asterisk (*) is placed before the variable name that holds the values of all non-keyworded variable arguments. It is especially useful when we don't know how many values to be passed in the function. For eg., a sum function.
- NOTE: The variable that we associate with the * becomes an iterable, which means we can iterate over it.

ANSWER 4:

- Result of the code is : `1 {'c': 3, 'b': 2}`
- The double asterisk (**) is placed before the variable name that holds the values of all **keyworded** arguments in a **dictionary format**. (**) allows us to pass through any number of keyword arguments.
- NOTE: **kargs, being a dictionary maps each keyword to the value that we pass alongside it.

ANSWER 5:

- Result of the code is : `1 5 6 5`
- `func(1, *(5, 6)) ;` since two parameters in the function have default values, this function will take 2 to 4 parameters. Here only 3 parameters are given so a = 1, b = 5, c = 6 and d will take the default value which is 5.

ANSWER 6:

- Result of the code is : `(1, ['x'], {'a': 'y'})`
- When the function is called, `func(l, m, n)`, here the values of `l, m` and `n` are passed to the function. At this point the function looks like this: `func(1, [1], {'a':0})`
- When compiler went to the function

```
def func(a, b, c):
    a = 2
    b[0] = 'x'
    c['a'] = 'y'
```

  Here `b[0] = 'x'` changed the original value of list and similarly `c['a'] = 'y'` updated the dictionary entry with key `'a' = 'y'`. Thus changing the value of list `m` and dictionary `n`.
- NOTE: LISTS and DICTIONARIES are mutable.