Nathaniel Daniel

Nicolescu

CS 326

16 November 2020

HW 6

1. You cannot build a macro to do this as C macros are not recursive.
2.
    a. The stack is not cleared between function calls. As a result, the old value of y is used as the y from the first call to p and the y from the second call to p happen to share the same memory location.
    b. The value will not be retained if another function overwrites the value. For example:

```
void p (){
        int y;
        printf ("%d ", y);
        y = 2;
}
void q(){
        unsigned long long data = 12345; // y's memory is overwritten here.
        printf ("%llu ", data);
}
void main (){
        p();
```

```
        q();
        p();
    }
```

3.

a. In a language with only pass by value semantics, a swap routine can only ever get a copy of its arguments and cannot mutate what is passed to it. As a result, it cannot perform the necessary mutations needed for a swap operation.
b. Languages with only pass by name are also unable to implement a generic swap routine. Some parameters might depend on each other, like swap(i, a[i]). This would mutate i while the procedure is running, leading to an incorrect array access and final result.

4.

a. No mutations take place, the values specified in main are printed.

   1

   1

b. In p, x and y are both references to a[0]. a[0] is incremented by 2.

   3

   1

c. The order of which parameters are updated from the function call is ambiguous. x++ and y++ are also both ambiguous as the correct value to use for x and y is unspecified.

   3

   1

5. A program typically does not run faster if optional parameters are not specified as most languages use a fallback value.