Miguel Muniz
CS 326
Hw 6
4/24/2023

1.
a)
The variable i appears to "remember" its values between the calls because the variable is not set to a value rather its left open to be incremented by the stack between function calls of foo. -

b)

```
void foo() {
    int i;
    printf("%d ", i++);
}

void bar(){
    int i = 2;
}

void main() {
    int j;
    for (j = 1; j <= 10; j++) {
    foo();
    bar();
    }
}
```

To alter the behavior I create a bar function that sets the value of i to 2. This changes the increment as its reset to 2 every time the function bar is called.
0 2 2 2 2 2 2 2 2 2

2.
It’s not possible to write a macro in c that returns a factorial of an integer as macros are not recursive. A part of that is thanks to the macro being processed during compile time.

3.
a)
In a language like C that is pass by value, the general purpose swap subroutine would be impossible to w rite
due to the fact that the subroutine would not be able to change the values of the variables/arguments pas sed in.

b)
In a language like Algo 60 that is pass by name, the general purpose swap subroutine would also be impo ssible to write
due to the fact that the subroutine would not be able as some of the parameters would rely upon each oth er.

4.
a)
Pass by value: The values aren’t changed, the values are printed in main.
Output: 1 10 11

b)
Pass by refrence: x value gets i and increments to 2. y = z which currently has the incremented i of 2 whic h then assigns a[1] as 2. z increment’s i once more to print i as 3, and a[2] is unchanged.
Output: 3 2 11

c)
Value-result: The function returns the i/x value as 2, it then returns a[1]/y as 1 as it is equal to z which whe n z is put into the argument is
one but z returns as also 1. a[2] therefore isn’t touched.
Output: 2 1 11

d)
Pass by name: the funcion returns the value of i as 2 thanks to the x variable. a[2] is equal to the variable i. and then I is once again incremented.
The potential issues with this method is that its ambiguous wheter the index of the array is evaluated befo re the function or while the funciton is execuiting.
Another issue is that the copied value of i should be recycled in the parameter such as when i = x vs i = z.

Output: 2 10 3

5. No the program will not run faster, as the compiler will just assign a default value that does not take mo re time then a user inputed value.