Homework 3

Benjamin Jakubowski

July 20, 2016

1. Tail recursion

a. Tail recursive versions of `F` and `G`

First, we present tail recursive versions of the methods `f` and `g`. We present the tail recursive versions in psuedocode (i.e. python).

```
def f_tc(x, accum = 0):
    if x == 0:
        return accum
    else:
        return g_tc(x - 1, accum + 1)

def g_tc(x, accum = 0):
    if x == 0:
        return accum
    else:
        return f_tc(x - 1, accum + 2)
```

b. Iterative version `FG_ITER`

Now we present an iterative version of the method. Note in addition to the integer x, the method has an additional formal parameter `func`. This parameter indicates the function to apply (`f` or `g`) and as such the function must be passed either `'f'` or `'g'`.

```
def fg_iter(x, func):
    accum = 0
    while x != 0:
        if func == 'f':
            accum = accum + 1
            func = 'g'
        elif func == 'g':
            accum = accum + 2
            func = 'f'
```

```
        else:
            raise ValueError
        x = x - 1
    return accum
```

2. Tail recursion

Consider a free list with the following block size entries: 50, 20, 100, 50, 30, 60. When allocating a memory block of size n from a larger free block of size k, assume that a block of size $k - n$ will be placed on the free list in the same cell as the original block. Further assume that if there are multiple blocks of the same size on the free list satisfying an allocation request, the first satisfying block on the list will be allocated.
Assume that allocation requests are made in the following order: 20, 20, 30, 50, 50, 45.

We show the free list under different memory allocation strategies:

a. First fit

Insert	**Free list following insertion**
NA (Initial state)	[50, 20, 100, 50, 30, 60]
20	[30, 20, 100, 50, 30, 60]
20	[10, 20, 100, 50, 30, 60]
30	[10, 20, 70, 50, 30, 60]
50	[10, 20, 20, 50, 30, 60]
50	[10, 20, 20, 30, 60]
45	[10, 20, 20, 30, 15]

b. Best fit

Insert	**Free list following insertion**
NA (Initial state)	[50, 20, 100, 50, 30, 60]
20	[50, 100, 50, 30, 60]
20	[50, 100, 50, 10, 60]
30	[20, 100, 50, 10, 60]
50	[20, 100, 10, 60]
50	[20, 100, 10, 10]
45	[20, 55, 10, 10]

c. Worst fit

Insert	Free list following insertion
NA (Initial state)	[50, 20, 100, 50, 30, 60]
20	[50, 20, 80, 50, 30, 60]
20	[50, 20, 60, 50, 30, 60]
30	[50, 20, 30, 50, 30, 60]
50	[50, 20, 30, 50, 30, 10]
50	[20, 30, 50, 30, 10]
45	[20, 30, 5, 30, 10]

d. Additional 30 byte allocation

Based on the final states of the free list after allocating 20, 20, 30, 50, 50, and 45 bytes of memory, all of the allocation strategies will support an additional 30 byte request.