CS106AP Midterm Exam Solutions

Summer 2019

1. Trace: The Mystery Bill

Part 1:

The output of the function call `zoo(6, 2, 0.5)` is 2600.

Part 2:

There are two issues with the code. The first is that the `kakapo()` function makes a modification to the parameter x that is passed in, but never returns that value. Since that specific value of x only exists in the `kakapo()` function, this function has no effect on the larger outcome of the program as currently written. We need to modify the `kakapo()` function to look like this:

```
def kakapo(x, p):
    x = x - x*p
    return x
```

The second issue is that we never do anything with the return value of the `kakapo()` function in `zoo()`. To fix this, we would do the following:

```
def zoo(c,h,p):
    x=0
    x += pangolin(c)
    x += sloth(h)
    x = kakapo(x,p)
    print(x)
```

Common errors

- To receive full credit for your explanation, you need to state both that `kakapo()` needed to **return `x`** and also that the return value needed to be **assigned** to `x` inside `zoo()`.
- Be careful with variable names here. Just because a variable named `c` exists inside both `zoo()` and `sloth()`, these two variables are not actually related! Because `zoo()` passes `h` into sloth, it passes the *value* 2 into the function, which is the correct value for the number of hospital services, despite the parameter's name inside `sloth()`. `sloth()` can choose to name this variable whatever it wants (`c` could have been named `panda`, and it wouldn't matter!). The same is true for `pangolin()`. Although the variable names are confusing, they don't actually cause any bugs in the program.

2. Karel: Pharmacist Karel

```
from karel.stanfordkarel import *

# PROVIDED
def turn_right():
    turn_left()
    turn_left()
    turn_left()

# PROVIDED
def turn_around():
    turn_left()
    turn_left()

def move_until_blocked():
    while front_is_clear():
        move()

def fill_vial():
    while front_is_clear():
        put_beeper()
        move()
    put_beeper()
    turn_around()

# Not provided in the decomp
def scale_vial():
    while not right_is_clear():
        move()
    turn_right()
    move()
    turn_right()

def main():
    """
    Your code goes here!
    """
    move_until_blocked()
    turn_left()
    while front_is_clear():
        scale_vial()
        move()
        fill_vial()
        scale_vial()
        move_until_blocked()    # descend from vial
        turn_left()
        move_until_blocked()    # move to next vial
        turn_left()


####### DO NOT EDIT CODE BELOW THIS LINE ########

if __name__ == '__main__':
    execute_karel_task(main)
```

3. Images: Detecting cancerous growths

Part A

```
CANCEROUS_THRESHOLD = 50

def highlight_cancerous_growths(filename):
    image = SimpleImage(filename):
    for pixel in image:
        average = (pixel.red + pixel.blue + pixel.green) // 3
        # Float division (/ instead of //) also fine in the line above
        if average < CANCEROUS_THRESHOLD:   # Both < and <= accepted
            pixel.green = 255
            pixel.red = 0
            pixel.blue = 0
        else:
            pixel.red = average
            pixel.green = average
            pixel.blue = average
    return image
```

Part B

```
def improve_detection_accuracy(best_image, images):
    """
    best_image: an unprocessed SimpleImage object of the potentially
    affected area
    images: a list of SimpleImage objects
    """
    for y in range(best_image.height):
        for x in range(best_image.width):
            count = 0
            for image in images:
                processed_pixel = image.get_pixel(x, y)
                if processed_pixel.green == 255:
                    count += 1
            if count > len(images) / 2:
                pixel.green = 255
                pixel.red = 0
                pixel.blue = 0

    return best_image
```

4. Console Program: Online Appointments

Part A

```
def convert_to_24_hour_time(time):
    """
    Takes in a string representing a time:
      X[AM|PM]
    where X is a number between 1-12

    Returns an int between 0-24 corresponding to the time.
    """
    am_index = time.find('AM')
    pm_index = time.find('PM')
    if am_index != -1:                      # Inputted morning time
        time = int(time[:am_index])
        if time == 12:                      # Handle the edge case of 12AM
            time = 0
    else:                                   # Inputted evening time
        time = int(time[:pm_index]) + 12
        if time == 24:                      # Handle the edge case of 12PM
            time = 12
    return time

# ALTERNATE SOLUTION
def convert_to_24_hour_time(time):
    """
    Takes in a string representing a time:
      X[AM|PM]
    where X is a number between 1-12

    Returns an int between 0-24 corresponding to the time.
    """
    time_num = int(time[:len(time) - 2])  # Get just the number as an int
    if time.find('PM') != -1:              # Add 12 if PM time
        time_num += 12
    if time_num % 12 == 0:                 # Handle 12AM/PM edge cases
        time_num -= 12
    return time_num
```

Part B

```
def get_appointments(filename):
    appointments = []
    with open(filename, 'r') as f:
        for line in f:
            times = line.split()
            for time in times:
              appointments.append(convert_to_24_hour_time(time))
    return appointments
```

Part C

```
def schedule_appointment(filename):
    """
    Times are inputted in the following form:
      The user will only input times on the hour (1PM, 2PM, 8AM, etc...)
      The number indicating hour is immediately followed by AM/PM
      All PM/AM will be in uppercase letters

    You should return a time converted to a 24-hour clock
      E.g. 12AM = 0, 1AM = 1, ..., 12PM = 12, 1PM = 13, 2PM = 14, etc.
    """
    existing_appointments = get_appointments(filename)
    best_time = 25
    time = input("Please input a time when you're available or DONE when
finished: ")
    while time != 'DONE':
        time_num = convert_to_24_hour_time(time)
        if time_num < best_time and time_num not in existing_appointments:
            best_time = time_num
        time = input("Please input a time when you're available or DONE when
finished: ") # Reprompt

    print('Your appointment is scheduled for :', str(best_time), "o'clock")
```

5. Dictionaries: Patient Visit Count

```
def count_visits(filename):
    count_dict = {}
    with open(filename, 'r') as f:
         for line in f:
             name = line.split()[1]
             if name not in count_dict:
                count_dict[name] = 0
            count_dict[name] += 1
    return count_dict
```