Carson Sklare

2. Code:

```
def all_segments(s):
    split = all_segments_r(s,0)
    if split != []:
        print(split)


def all_segments_r(s,i):
    segments = []
    if i == len(s):
        return [] #return origianl word
    for j in range(i+1,len(s)+1):
        if is_word(s[i:j]):
            split = all_segments_r(s,j)
            #print(split)
            if split != None:
                segments.append(s[i:j])
                segments += split
    return segments
```

Results:

```
>>> all_segments('hoorayitisfirday')
['ho', 'or', 'a', 'ay', 'it', 'is', 'fir', 'day', 'hooray', 'it', 'is', 'fir', '
day']
>>> all_segments('hellohowareyou')
['he', 'll', 'oh', 'ow', 'a', 're', 'you', 'are', 'you', 'oho', 'wa', 're', 'you
', 'war', 'ware', 'you', 'hell', 'oh', 'ow', 'a', 're', 'you', 'are', 'you', 'oh
o', 'wa', 're', 'you', 'war', 'ware', 'you', 'hello', 'ho', 'wa', 're', 'you', '
war', 'ware', 'you', 'how', 'a', 're', 'you', 'are', 'you']
>>> all_segments('whenlilacs')
['when', 'lilac', 'lilacs']
```

3.

A.

Code

```
#able to get down to #5 on hw3 DNA
def bf_seq(p):
    per = permutations(p)
    for perm in per:
        test = True
        for i in range(1,len(perm)):
            if not(perm[i-1][-2:] == perm[i][:2]):
                test = False
        if(test == True):
            return perm
```

B. Can go up to #5 before not working
C not working: code

```
def seq(p):
    w = p[:1]
    w2 = w[0]
    return seq_r(p[1:],w2)

##def seq_r(p,w):
##    if(p[0] == w):
##        return p
##    for i in p:
##        for j in range(1,len(i)):
##            test = True
##            if not(p[j-1][-2:] == p[j][:
##                test = False
##                p = p[:j]+p[j+1:]
##                seq_r(p)
##            if test:
##                return
##    return p

def seq_r(p,w):
    #print('p[0]: {}'.format(p[0]))
    #print('w: {}'.format(w))
    lst = []
    for i in p:
        test = True
        if not(w[-2:] == i[:2]):
            test = False
        if test:
            lst.append(w)
            #print('lst: {}'.format(lst))
            #print('p: {}'.format(p))
            se = seq_r(p[1:],i)
            if se != None or se != []:
                lst.append(se)
    return lst
```

result:

```
>>> lst = ['abcd','cder','cdtr','dfgh']
>>> seq(lst)
['abcd', [], 'abcd', []]
```

D. not entirely working, returns false if its not completely working

```
def find_seq(p):
    w = p[:1]
    w2 = w[0]
    return find_seq_r(p[1:],w2)

def find_seq_r(p,w):
        #print('p[0]: {}'.format(p[0]))
    #print('w: {}'.format(w))
    lst = []
    for i in p:
        test = True
        if not(w[-2:] == i[:2]):
            test = False
        if test:
            lst.append(w)
            #print('lst: {}'.format(lst))
            #print('p: {}'.format(p))
            se = seq_r(p[1:],i)
            if se != None or se != []:
                lst.append(se)
        else:
            return test
    return lst
```

Result:

```
def find_seq(p):
    w = p[:1]
    w2 = w[0]
    return find_seq_r(p[1:],w2)

def find_seq_r(p,w):
        #print('p[0]: {}'.format(p[0]))
    #print('w: {}'.format(w))
    lst = []
    for i in p:
        test = True
        if not(w[-2:] == i[:2]):
            test = False
        if test:
            lst.append(w)
            #print('lst: {}'.format(lst))
            #print('p: {}'.format(p))
            se = seq_r(p[1:],i)
            if se != None or se != []:
                lst.append(se)
        else:
            return test
    return lst
```