Homework 5
Adding Functions to the Mix

Due Date: Wednesday, October 21st, 2020 by 11:59:59 PM
Value: 50 points

This assignment falls under the standard cmsc201 academic integrity policy. This means you should not discuss/show/copy/distribute your solutions, your code or your main ideas for the solutions to any other student. Also, you should not post these problems on any forum, internet solutions website, etc.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
"""
File:    FILENAME.py
Author:  YOUR NAME
Date:    THE DATE
Section: YOUR DISCUSSION SECTION NUMBER
E-mail:  YOUR_EMAIL@umbc.edu
Description:
  DESCRIPTION OF WHAT THE PROGRAM DOES
"""
```

I've rated the problems based on what I think the expected difficulty is. This is not a guarantee about what you'll find hard or easy, but it's basically just my expectation.

Submission Details

Submit the files under the following titles:
(These are case sensitive as usual.)
submit cmsc201 HW5 {files go here}

Problem 1 - Genetic Extraction	hw5_part1.py
Problem 2 - Pascal Levels	hw5_part2.py
Problem 3 - Robo Vac	hw5_part3.py
Problem 4 - not to be done	
Problem 5 - Reverse Selection	hw5_part5.py

Coding Standards

Coding standards can be found here.

We will be looking for:

1. At least one inline comment per program explaining something about your code.
2. Constants above your function definitions, outside of the "if __name__ == '__main__':" block.
    a. A magic value is a string which is outside of a print or input statement, but is used to check a variable, so for instance:
        i. `print(first_creature_name, 'has died in the fight. ')` does not involve magic values.
        ii. However, `if my_string == 'EXIT':` exit is a magic value since it's being used to compare against variables within your code, so it should be:
```
EXIT_STRING = 'EXIT'
...
if my_string == EXIT_STRING:
```
    b. A number is a magic value when it is not 0, 1, and if it is not 2 being used to test parity (even/odd).
    c. A number is magic if it is a position in an array, like my_array[23], where we know that at the 23rd position, there is some special data. Instead it should be
    USERNAME_INDEX = 23
    my_array[USERNAME_INDEX]
    d. Constants in mathematical formulas can either be made into official constants or kept in a formula.
3. Previously checked coding standards involving:
    a. snake_case_variable_names
    b. CAPITAL_SNAKE_CASE_CONSTANT_NAMES
    c. Use of whitespace (2 before and after a function, 1 for readability.)

Allowed Built-ins/Methods/etc

- Declaring and assigning variables, ints, floats, bools, strings, lists, dicts.
- Using +, -, *, /, //, %, **; +=, -=, *=, /=, //=, %=, **= where appropriate
- Comparisons ==, <=, >=, >, <, !=, in
- Logical and, or, not
- if/elif/else, nested if statements
- Casting int(x), str(x), float(x), (technically bool(x))
- For loops, both *for i* and *for each* type.
- While loops
    - sentinel values, boolean flags to terminate while loops
- Lists, list(), indexing, i.e. my_list[i] or my_list[3]
    - 2d-lists if you want them/need them my_2d[i][j]
    - Append, remove
    - **list slicing**
- If you have read this section, then you know the secret word is: createous.
- String operations, concatenation +, +=, split(), strip(), join(), upper(), lower(), isupper(), islower()
    - **string slicing**
- Print, with string formatting, with end= or sep=:
    - '{}'.format(var), '%d' % some_int, f-strings
    - Really the point is that we don't care how you format strings in Python
    - Ord, chr, but you won't need them this time.
- Input, again with string formatting in the prompt, casting the returned value.

- Using the functions provided to you in the starter code.
- Using import with libraries and specific functions **as allowed** by the project/homework.

Forbidden Built-ins/Methods/etc

This is not a complete listing, but it includes:

- break, continue
- methods outside those permitted within allowed types
    - for instance str.endswith
    - list.index, list.count, etc.
- Keywords you definitely don't need: await, as, assert, async, class, except, finally, global, lambda, nonlocal, raise, try, yield
- The *is* keyword is forbidden, not because it's necessarily bad, but because it doesn't behave as you might expect (it's not the same as ==).
- built in functions: any, all, breakpoint, callable, classmethod, compile, exec, delattr, divmod, enumerate, filter, map, max, min, isinstance, issubclass, iter, locals, oct, next, memoryview, property, repr, reversed, round, set, setattr, sorted, staticmethod, sum, super, type, vars, zip
- If you have read this section, then you know the secret word is: efficacious.
- exit() or quit()
- If something is not on the allowed list, not on this list, then it is probably forbidden.
- The forbidden list can always be overridden by a particular problem, so if a problem allows something on this list, then it is allowed for that problem.

Problem 1 - Genetic Extraction (Hard)

Your objective is to code a function which will take in a string as a parameter and return a list of strings which we will call 'genes.'

If you're ever confused about the genetics, remember that in the end you don't have to know anything about genetics beyond what is here. In fact I will state the problem in a completely computer science-y way at the end of the problem statement.

In DNA, there are four nucleotides, which are denoted A, T, C, G. Nucleotides group in sets of three to form codons. Here's an example:

AGTATC**GTG**TAT**GCT**ATA**ATC**GAT**AGT**CTT**ACC**AGT**TTG**GTT**GAG**TCT

I have bolded every other codon to show you each one. Our strings will always be a length divisible by 3, so you don't have to worry about extra nucleotides (characters) on the end of your string.

So your objective is to extract all of the genes, which in our definition start with a start codon:

The only start codon is: ATG
The three stop codons are: TAG, TAA, and TGA

These are all considered magic, so you'll need to make constants out of them in your program.

Let me show you an example of gene extraction:

If our sequence is:
ATGCTC**GCT**TAG
then it is a gene, so we return
['**ATG**CTC**GCT**TAG']

On the other hand, if we have two start codons:
ATGCTC**ATG**CTC**GCT**TAG
then we return:
['**ATG**CTC**ATG**CTC**GCT**TAG', '**ATG**CTC**GCT**TAG']
This is because we have two ATG start codons and one stop codon.

Whenever you see a start codon, you should remember where it was. Whenever you then find a stop codon, you should then stop all of the genes. Never read past a stop codon.

In this example there is no stop codon, so no genes.
ATGCTC**ATG**CTC
Return an empty list. []

In this example there are no start codons, but there are stop codons, so no genes.
TAACAG**CCC**CTA**ATT**TAG**TTC**TGT**CTG**TCC**GAC**GTT
Return an empty list. []

For example, the sequence:
ATGGCT**ATG**GCC**GAA**AGT**ATG**TGG**TGA**AAC**ATG**CAC
should return:
['ATGGCTATGGCCGAAAGTATGTGGTGA',
'ATGGCCGAAAGTATGTGGTGA', 'ATGTGGTGA']

Your function should begin:

```
def extract_genes(sequence):
    """
    This function should return a list of genes that start with the
    start codon and end with one of the three stop codons.
    :param sequence: a string of nucleotides
    :return: a list of "genes"
    """
    # your code goes here
```

Test Driver and Starter Code

```
The starter code is at:
/afs/umbc.edu/users/e/r/eric8/pub/cs201/fall20/hw5_part1_starter.py
```

You should copy it to your hw5 directory using the cp command:

```
cp
/afs/umbc.edu/users/e/r/eric8/pub/cs201/fall20/hw5_part1_starter.py
hw5_part1.py
```

Do not forget the second part of the line (on the next line). That will copy the file into the new name.

You should only write additional global constants, and code within the extract_genes function. Feel free to modify the driver code, or just use it as is. The only part of your code that will be tested is the extract_genes function.

Hints:

1. Consider using a list to keep track of the start codon positions.

2. You are permitted and encouraged to use slices on this question.

Sample Output

```
linux5[201]% python3 hw5_part1.py
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)seq
Enter your own sequence: ATGATGATGTAA
ATGATGATGTAA
['ATGATGATGTAA', 'ATGATGTAA', 'ATGTAA']
How many codons do you want to create? (or stop to
end, seq to enter your own
sequence)ATGTAAATGTAAATGTTAA
You entered a non-STOP non-integer, try again.
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)seq
Enter your own sequence: ATGTAAATGTAAATGTAG
ATGTAAATGTAAATGTAG
['ATGTAA', 'ATGTAA', 'ATGTAG']
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)seq
Enter your own sequence: ATGCTCATGTAGATGCCACTGTGAACC
ATGCTCATGTAGATGCCACTGTGAACC
['ATGCTCATGTAG', 'ATGTAG', 'ATGCCACTGTGA']
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)stop
```

```
linux5[202]% python3 hw5_part1.py yellow
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)36
ACGGCAGTAGTCTACTTATCCACGGCAAGAGTGTAACGCGTCAATCGGGTCGA
AGCCGTCGCTAATTTAACCCACCATAACCACCTTCCACGTAGACGGAGGGTCT
GT
[]
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)36
TATTGACTCTTTGCATTTGGCTAATAACAGCCATAGGAGAAAGCACTGTTAGT
CGTATCGATCTTCTCGTGGACCCGGACTATCATACCTACCTGCCGGGCGGATT
CC
[]
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)36
TCTGGGGACCTATTCCTTTTGGATTATGCTACCACAGCAGCATCCCAATCATG
CTGGGAAGGACCTATGGCCTGCATCCTTGTAAGTGGTGAGTATCCCACCATGA
GA
[]
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)36
CATCAATGCCCTTCGAATAAGAACGCTATGCATAATGCAATCGACCTATATTA
AGTTTTACGTCACCGAGCCACAAGTTTACACGCTCTAAAAGGTATGCTTGTGT
TA
['ATGCATAATGCAATCGACCTATATTAA']
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)stop
```

Note here that with randomly generated codons, it's rather unlikely that you're going to encounter a gene in such a short span. Using lengths of 1000 are about what you need to produce likely genes.

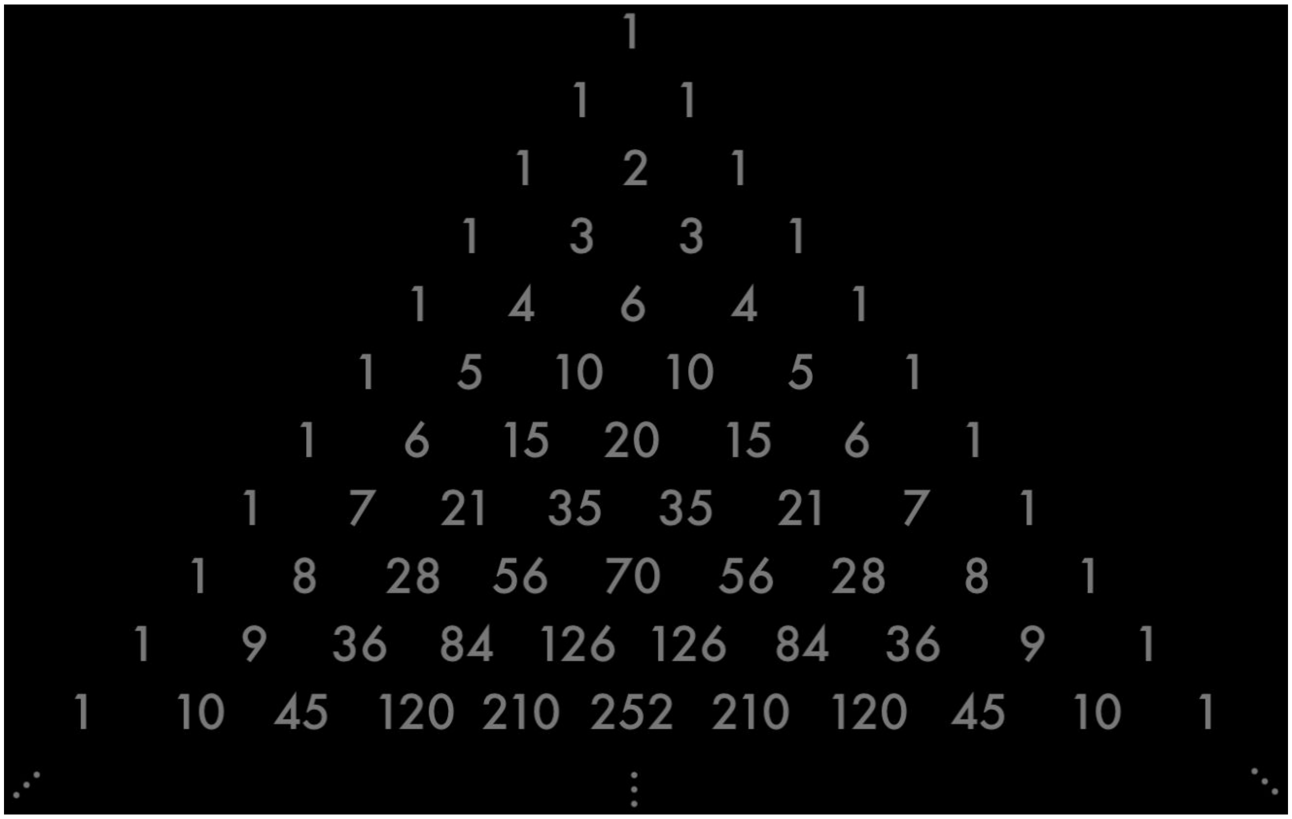
```
linux5[202]% python3 hw5_part1.py robotical
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)40
CGAAGAGTTTGATGCCGGTTGTTACCAATGCCTATGTGATCGATGCGGGTATT
TGCGGATGCCCAAATAGTCTTCTGAAGATATCAACAATACTACATTTAGTCTA
GTAGATCTACCCCT
['ATGCCTATGTGA', 'ATGTGA',
'ATGCGGGTATTTGCGGATGCCCAAATAGTCTTCTGA']
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)40
GCGTTCAAGGACTGTCAGCTTGTCAGGGTGATATCAAAAGTGAGGGTTGCGAG
GTACCCGTTTTGACCACGATATTTGGGTCCGTCGAGGGCCGGAATGCGATCAG
TGGTATTCATGGGG
[]
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)40
TGGCGAAGAGTGATCATGCGTCTTGATCTATTTCTGCCCGTGGTTCCTAGTAC
GCGGGGACCCTCCATGGTCTCTATCGTGGACCCTCGATCATAGTCTCGTGAAT
AGCGCTTTACACGT
['ATGCGTCTTGATCTATTTCTGCCCGTGGTTCCTAGTACGCGGGGACCCTCC
ATGGTCTCTATCGTGGACCCTCGATCATAG',
'ATGGTCTCTATCGTGGACCCTCGATCATAG']
How many codons do you want to create? (or stop to
end, seq to enter your own sequence)stop
```

Here are some better results, robotical is apparently a good random seed.

Problem 2 - Pascal Levels (Medium)

For this problem you're going to take a list, and compute the next level of Pascal's triangle based on the previous level (if the previous list isn't in Pascal's triangle, do it anyway).

To compute the next level in Pascal's triangle, it's important to know what Pascal's Triangle is.



Note here that if you look at any entry, and then at the two entries at a diagonal above it, you'll see that the entry is always equal to the sum of the two diagonal entries above it.

So for instance given the list [1, 4, 6, 4, 1] your objective is to return the list [1, 5, 10, 10, 5, 1].

You'll probably have to add an extra 1 on the end of your list in order to maintain the Pascal Triangle property.

Test Driver and Starter Code

```
The starter code is at:
/afs/umbc.edu/users/e/r/eric8/pub/cs201/fall20/hw5_part2_starter.py
```

You should copy it to your hw5 directory using the cp command:

```
cp
/afs/umbc.edu/users/e/r/eric8/pub/cs201/fall20/hw5_part2_starter.py
hw5_part2.py
```

Do not forget the second part of the line (on the next line). That will copy the file into the new name.

If you don't, use the command:

mv hw5_part2_starter.py hw5_part2.py

Feel free to modify the driver (the name == main part) however you want to test new lists. We will only be testing your pascal_level function.

Sample Output

```
linux5[109]% python3 hw5_part2.py
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
[1, 5, 10, 10, 5, 1]
[1, 6, 15, 20, 15, 6, 1]
[1, 7, 21, 35, 35, 21, 7, 1]
[1, 8, 28, 56, 70, 56, 28, 8, 1]
[1, 9, 36, 84, 126, 126, 84, 36, 9, 1]
[1, 3, 5, 7, 9, 1]
[1, 2, 3, 5, 8, 13, 1]
[1, 2, 3, 5, 8, 13, 21, 14, 1]
```

Problem 3 - RoboVac (Hard)

For this problem you're going to write a function:

```
def robo_vac(room, starting_y, starting_x):
    """
    Compute a path through the room for the robot vacuum.

    :param room: a 2d-grid representing the room,
             '' blank strings are open spaces
    :param starting_y: starting row (first index)
    :param starting_x: starting column (second index)
    :return: the updated room with the numbers.
    """
```

The function will take a room, as a 2d-list, and two parameters, starting_y and starting_x, and return an updated 2d list with the path the vacuum will take.

The robot is very stupid, and will **not** backtrack, meaning it will not visit any position that is either forbidden or already has been visited.

The path of the robot is determined by the following pseudo-code:

1. If the robot can move in the x direction, it will move "forward" (positively) in the x direction by 1.
2. If not, but the robot can move in the y direction, it will move "down" (positively) in the y direction by 1.
3. If not, but the robot can move in the x direction "backwards" it will move in the x direction negatively by 1.
4. If not, but the robot can move in the y direction "up" it will move up (negatively) by 1.
5. If none of these, the robot will stop moving. Terminate the function and return the list.
6. Record the step at the current position, then move the robot, and go back to 1.

If a space is allowed, it will be an empty string, if it is forbidden, then it will have something in it.

Any space which has been rolled over by RoboVac becomes forbidden, because you put the number in the space.

Hints:

1. You must check that the y-position is between 0 and the number of rows in the 2d-list. Remember that between in the case of lists includes 0 but excludes the length.
2. You must check that the x-position is between 0 and the length of the row.
3. You must check that the position is not forbidden or already rolled on. The hint part of this hint is: consider whether the empty string is True or False.
4. You don't know how long robo-vac will go, so which type of loop is better for that?

Test Driver and Starter Code

```
cp
/afs/umbc.edu/users/e/r/eric8/pub/cs201/fall20/hw5_part3_starter.py
hw5_part3.py
```

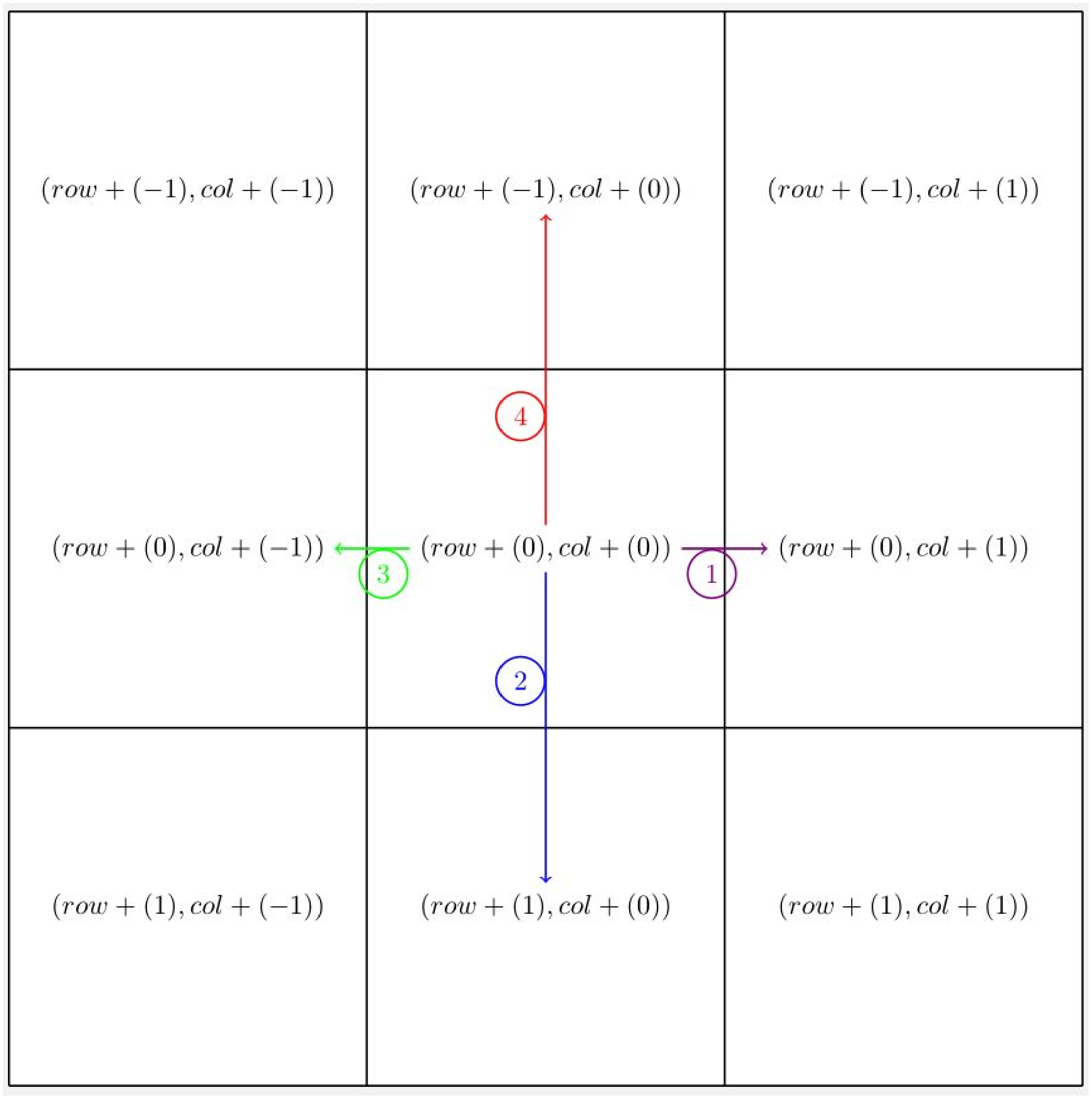
Remember to put the second part of that line (it's all one line) so that the copied file will have the right name. If not use mv:

```
mv hw5_part3_starter.py hw5_part3.py
```

Feel free to modify the driver code to create new maps and test them out.

Explanatory Picture

The numbers represent the positions where you should move in which order. So, if you are at position (r, c) you should try for (r, c+ 1) first, then (r + 1, c), then (r, c - 1) then finally (r - 1, c). So, purple, blue, green, red is the order in which to try moving to the next space.


$(row + (-1), col + (-1))$
$(row + (-1), col + (0))$
$(row + (-1), col + (1))$
4
$(row + (0), col + (-1))$
3
$(row + (0), col + (0))$
1
$(row + (0), col + (1))$
2
$(row + (1), col + (-1))$
$(row + (1), col + (0))$
$(row + (1), col + (1))$

Sample Output

```
linux5[109]% python3 hw5_part3.py
1    2    3    4    5    6
36   35   34   33   32   7
27   28   29   30   31   8
26   25   24   23   22   9
17   18   19   20   21   10
16   15   14   13   12   11

1    2    3    4    5    6
32   31   30   29   28   7
23   24   25   26   27   8
22   21   20   x    x    9
17   18   19   x    x    10
16   15   14   13   12   11

1    2    3
.    x    x
.    .    x

3    2    1
4    x    x
5    6    x

1    x    .    .    .    .    .    .
2    x    x    .    x    .    x    .
3    x    .    .    x    x    21   20
4    5    6    7    x    .    x    19
.    .    x    8    x    .    x    18
.    .    x    9    x    .    x    17
.    .    .    10   x    .    x    16
.    .    .    11   12   13   14   15
```

note: Problem 4 is not assigned to Section 60 as we have not yet covered Dictionaries

Problem 5 - Reverse Selection (Easy)

A **reverse** selection sort works in the following way:

1. For each index i in the list: [14, 2, 8, 13, 1]
    a. Find the **maximum** in the list starting at i.
    b. Swap the elements at max position and i
    c. Go back to 1a to get a new index.
2. The list should be reverse sorted, meaning that larger elements come before smaller elements.

The stages of sorting will produce:

- Unsorted: [14, 1, 8, 13, 2]
- 1st pass i = 0, [14, 1, 8, 13, 2], max was 14
- 2nd pass, i = 1, [14, 13, 8, 1, 2], max was 13
- 3rd pass, i = 2, [14, 13, 8, 1, 2], max was 8
- 4th pass, i = 3, [14, 13, 8, 2, 1], max was 2
- 5th pass, i = 4, technically unnecessary, because it's reverse sorted now.

Test Driver and Starter Code

You will find a test driver and starter code at:

```
/afs/umbc.edu/users/e/r/eric8/pub/cs201/fall20/hw5_part5_starter.py
```

You should copy it to your hw5 directory using the cp command:

```
cp
/afs/umbc.edu/users/e/r/eric8/pub/cs201/fall20/hw5_part5_starter.py
hw5_part5.py
```

Don't forget the last part in this, even though it's on the next line you need it to rename the file. If you don't you can use the command:

```
mv hw5_part5_starter.py hw5_part5.py
```

You must implement three functions:

Swap must take the indices i and j and swap the values at those indices.

```
def swap(the_list, i, j):
    """
    :param the_list: the list that we're reverse sorting
    :param i: the first position to swap
    :param j: the second position to swap
    """
    pass
```

find_max_index_after must find the **index** of the maximum element starting at (and including the index) *i*. If two elements are equal, keep the first one you find.

```
def find_max_index_after(i, the_list):
    """
    :param i: start at position i to look for the max
    :param the_list: the list to search
    :return: the INDEX of the maximum, not the maximum itself!
    """
    pass
```

reverse_selection sort must follow the pseudo-code given at the top of the problem to reverse-selection-sort the list, and then return the reverse-sorted list. You must call swap and find_max_index_after.

```
def reverse_selection_sort(the_list):
    """
    :param the_list: the list to reverse sort
    :return: the reverse sorted list
    """
    pass
```

Remember that pass is a no-op meaning it does nothing. It does act as a placeholder for code however.

Sample Output

```
linux5[281]% python3 hw5_part5.py
What length of list do you want to sort? (or STOP to end)5
The list is:  [33, 43, 19, 38, 34]
The reverse sort is:  [43, 38, 34, 33, 19]
What length of list do you want to sort? (or STOP to end)8
The list is:  [63, 62, 71, 96, 95, 94, 58, 13]
The reverse sort is:  [96, 95, 94, 71, 63, 62, 58, 13]
What length of list do you want to sort? (or STOP to end)17
The list is:  [26, 88, 11, 19, 32, 26, 1, 64, 98, 9, 63, 57,
45, 97, 27, 11, 24]
The reverse sort is:  [98, 97, 88, 64, 63, 57, 45, 32, 27, 26,
26, 24, 19, 11, 11, 9, 1]
What length of list do you want to sort? (or STOP to end)25
The list is:  [95, 29, 82, 69, 77, 80, 13, 21, 53, 64, 96, 77,
10, 39, 98, 89, 74, 100, 22, 4, 99, 48, 53, 69, 36]
The reverse sort is:  [100, 99, 98, 96, 95, 89, 82, 80, 77, 77,
74, 69, 69, 64, 53, 53, 48, 39, 36, 29, 22, 21, 13, 10, 4]
What length of list do you want to sort? (or STOP to end)STOP

linux5[282]% python3 hw5_part5.py
What length of list do you want to sort? (or STOP to end)1
The list is:  [17]
The reverse sort is:  [17]
What length of list do you want to sort? (or STOP to end)2
The list is:  [67, 5]
The reverse sort is:  [67, 5]
What length of list do you want to sort? (or STOP to end)3
The list is:  [79, 59, 46]
The reverse sort is:  [79, 59, 46]
What length of list do you want to sort? (or STOP to end)4
The list is:  [21, 26, 90, 19]
The reverse sort is:  [90, 26, 21, 19]
What length of list do you want to sort? (or STOP to end)5
The list is:  [77, 62, 89, 57, 96]
The reverse sort is:  [96, 89, 77, 62, 57]
What length of list do you want to sort? (or STOP to end)STOP
```