The following programs are due at the beginning of class on **Thursday, November 14**. You can submit your programs online via NYU Classes. Please submit a separate .py file for each program, and put your name and the problem/assignment number in a comment at the top of the program.

1. Rewrite your program from homework #2, problem #1 so that it properly handles any ValueError, TypeError, and ZeroDivisionError exceptions.

```python
# SOLUTIONS: Homework 8 − Problem 1
try:
    print("This program calculates the quotient and remainder of two numbers.")
    divisor = int(input("Please enter the divisor (the whole number by which you will be dividing). "))
    dividend = int(input("Please enter the dividend. "))
    quotient = dividend // divisor
    remainder = dividend % divisor
    print("The quotient is", quotient)
    print("The remainder is", remainder)
except ValueError as err:
    print("The value you entered was not an integer.")
except TypeError as err:
    print("Please make sure the value you enter is an integer.")
except ZeroDivisionError as err:
    print("The divisor you entered was 0, but you cannot divide by 0.")
```

2. You’ve been asked to write a program that grades a students’ test. Each question on the test is multiple choice with 4 possible answers (A, B, C, and D,) and there are 20 questions total. The answer key is:

The test takers store their answers in a text file "answers.txt". Your program should read this file, compare these answers with the answer key, and then tell the user the percentage of the questions they answered correctly. If the file "answers.txt" does not exist, it should give the student a 0.

# SOLUTIONS: Homework 8 - Problem 2

```python
answer_key = ['B', 'A', 'D', 'D', 'C',
'B', '7', 'A', 'C', 'C',
'D', 'B', 'A', 'B', 'A',
'C', 'B', 'D', 'A', 'C']
student_answers = []

try:
    infile = open('answers.txt', 'r')
    answer = infile.readline()
    answer = answer.strip('
')
    print(answer)
    while answer != '':
        student_answers.append(answer)
        answer = infile.readline()
        answer = answer.strip('
')
    infile.close()
    num_correct = 0
    for i in range(20):  #20 problems on the test
        if student_answers[i] == answer_key[i]:
            num_correct += 1
    print("This student’s score was a", 100*num_correct/20, "out of 100.")
except IOError:
    print("This student’s score was a 0 out of 100.")
```

3. You are writing a video game that stores the players’ scores as a nested list. So far you have the following scores (which you can hardwire into your program.)

<table>
<thead>
<tr>
<th>Player</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winner95</td>
<td>200</td>
</tr>
<tr>
<td>Selena</td>
<td>1000</td>
</tr>
<tr>
<td>PlayaHata</td>
<td>400</td>
</tr>
<tr>
<td>Jon</td>
<td>500</td>
</tr>
<tr>
<td>Arifur</td>
<td>300</td>
</tr>
</tbody>
</table>

Write a function `print_top_scores` that takes a nested list of scores and an integer n as arguments, and returns the top n scores in the list. For example, if you called
list_of_scores = [['Hamilton', 3000], ['Yulesi24', 1500], ['Jamel', 2000]]
print_top_scores(list_of_scores, 2)

then the program would print

Hamilton 3000
Jamel 2000

Use your function to write a program that asks the user for a number n, and prints the top n scores from the above table.

# SOLUTIONS: Homework 8 - Problem 3

def get_top_scores(list_of_scores, n):
    for score in list_of_scores:
        score.reverse()
    list_of_scores.sort()
    for score in list_of_scores:
        score.reverse()
    list_of_scores.reverse()
    return list_of_scores[:n]

player_scores = [['Winner95', 200], ['Selena', 1000], ['PlayaHata', 400], ['Jon', 500], ['Arifur', 300]]

top_scores = get_top_scores(player_scores, 3)
for score in top_scores:
    for info in score:
        print(info, end=' ')
    print()

4. Every list object in Python has a built-in sort method. There are many different algorithms for sorting a list. In class, we discussed one of the most common called bubble sort. Another common algorithm is selection sort, which works as follows:

1. Find the minimum value in the list.

2. Swap this value with the first value in the list. The first value is now considered sorted.

3. Repeat these steps for the remaining unsorted items in the list.

For example, the following diagram demonstrates selection sort being used to sort the list [7, 4, 5, 9, 8, 2, 1].
Write a function `selection_sort` that takes a list as an argument, sorts the list using selection sort, and returns the sorted list. Use your function to write a program that sorts the following list and prints the output to the screen: [4, 8, 1, 3, 2, 9, 5, 7, 6]. You can hardwire this list into your program if you want.

```python
# SOLUTIONS: Homework 8 – Problem 4

def selection_sort(some_list):
    for n in range(len(some_list)):
        smallest = min(some_list[n:]):
        i = some_list.index(smallest)
        some_list[i] = some_list[n]
        some_list[n] = smallest
    return some_list

eexample_list = [4, 8, 1, 3, 2, 9, 5, 7, 6]
print("This selection sorts the list", example_list)
sorted_list = selection_sort(example_list)
print("The sorted list is", sorted_list)
```