COMP 1900 - Fall 2021

Lab 3: Conditionals

Total Points: 32

Due: Fri., Oct. 1, by 2359 CDT
Please carefully read the submission instructions at the end of the assignment. As with all other assignments, this should be done *individually*.

Grader: The lab TA who supervises your registered lab section will grade your submission. Grades will generally be posted within 1-2 weeks of the assignment due date. Questions about grading? Please contact them first.

1. (6 points) The first three digits of a Tennessee ZIP code range from 370 to 385, inclusive. These three digits, called the **prefix**, determine which city handles the mail for that ZIP code. The Tennessee prefixes are divided like this:[1]

Prefix	Handled By
370, 371, 372	Nashville
373, 374	Chattanooga
375, 380, 381	Memphis
376	Johnson City
377, 378, 379	Knoxville
382	McKenzie
383	Jackson
384	Columbia
385	Cookeville

Note that this table does *not* say that all ZIP codes with those prefixes are geographically located within the indicated cities. The table expresses which city's post office is responsible for *handling* mail with that prefix.

Write a program named `tn_zip_codes.py` that reads user input for a 5-digit ZIP code, as an integer. You may assume that the user will always enter a 5-digit integer. Your program should

[1]`https://en.wikipedia.org/wiki/List_of_ZIP_Code_prefixes`

determine the three-digit prefix of the user's input and print which city handles mail for that ZIP code. If the prefix is something besides the values in the table above, print a message saying that it's outside of Tennessee.

Example program run (underlined parts indicate what the user enters)

```
Enter 5-digit ZIP code: 38103
Mail with ZIP code 38103 is handled by Memphis.
```

Example program run (underlined parts indicate what the user enters)

```
Enter 5-digit ZIP code: 37308
Mail with ZIP code 37308 is handled by Chattanooga.
```

Example program run (underlined parts indicate what the user enters)

```
Enter 5-digit ZIP code: 30332
30332 seems to be outside the state of Tennessee.
```

2. (8 points) **Zeller's congruence** (developed by German mathematician Christian Zeller in the late 1800s) gives a way to determine the day of the week of any calendar date. The formula states:

$$h = \left(q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + \left\lfloor \frac{J}{4} \right\rfloor + 5J \right) \bmod 7$$

where h is the day of the week (with 0 = Saturday, 1 = Sunday, 2 = Monday, etc.), q is the day of the month (1-31), m is the month (with 3 = March, 4 = April, 5 = May, ... , 12 = December, 13 = January, 14 = February), K is the year of the century (e.g., $K = 21$ for 2021), and J is the century (e.g., $J = 20$ for 2021).

Note that the brackets in this expression mean "floor" — i.e., round *down* the result of the division to the next integer. The tricky part about this formula is that January and February are treated as the 13th and 14th months of the *previous* year. Here are two examples of how to apply Zeller's congruence:

Example 1: July 20, 1969

$q = 20$, $m = 7$, $K = 69$, $J = 19$

Using these values, Zeller's congruence yields $h = 1$, which corresponds to Sunday.

Example 2: February 9, 2000

February is treated as the 14th month of the previous year, 1999.

$q = 9$, $m = 14$, $K = 99$, $J = 19$

Using these values, Zeller's congruence yields $h = 4$, which corresponds to Wednesday.

Write a program named `zeller.py` that allows the user to enter the month (1-12), day (1-31), and year. You may assume that the user will enter valid input. The program should then use Zeller's congruence to determine and display the corresponding day of the week.

Hint: Check the user's input to see if you need to make changes to the entered values for month and year. If the month is 1 or 2, you should replace it with 13 or 14 and also decrease the year by one.

Example program run (underlined parts indicate what the user enters)

```
Enter the month (1-12): 2
Enter the day of the month (1-31): 9
Enter the year: 2000

2/9/2000 is a Wednesday.
```

3. In the U.S., all earned income is subject to federal income tax. The tax system is divided into **brackets** so that higher earners pay a higher percentage of their income in tax. In the 2020 tax year (which is what you filed taxes for in 2021), the brackets were set up like this for a single filer:

Income	Tax Rate
Up to \$9,875	10%
Over \$9,875, up to \$40,125	12%
Over \$40,125, up to \$85,525	22%
Over \$85,525, up to \$163,300	24%
Over \$163,300, up to \$207,350	32%
Over \$207,350, up to \$518,400	35%
Over \$518,400	37%

A common misconception is that if you're near the border between two brackets, it's advantageous to refuse a small raise because you would end up with less after-tax pay if you get bumped into the next higher bracket. *This is totally false.* The tax rates in the table above are **marginal**, which means they apply only to the *portions* of income in each range, rather than all of the income.

Thus, earning more income will generally result in more after-tax pay. There *are* some exceptions, such as an income increase disqualifying a taxpayer for the Earned Income Tax Credit, but we won't consider those special cases here. (If you're interested in U.S. tax laws, see `https://www.irs.gov/credits-deductions/individuals/earned-income-tax-credit` for more info about the EITC. It's good bedtime reading.)

Note that federal tax is not the only thing withheld from your paycheck. There are also deductions for Social Security and Medicare (known as **FICA taxes**), state income tax (thankfully zero if you live in Tennessee), retirement contributions, insurance premiums, and so on. But for this problem, we'll focus on just the federal tax.

Here are two examples of how the federal tax is computed.

- Alice earns \$100,000 as a software engineer. Her income can be broken down into four parts:
  - The first \$9,875 incurs 10% of tax: \$9,875 × 0.1 = \$987.50
  - The part between \$9,875 and \$40,125 incurs 12% of tax: (\$40,125 - \$9,875) × 0.12 = \$3,630
  - The part between \$40,125 and \$85,525 incurs 22% of tax: (\$85,525 - \$40,125) × 0.22 = \$9,988
  - Finally, the part over \$85,525 incurs 24% of tax: (\$100,000 - \$85,525) × 0.24 = \$3,474

  Alice's total tax owed is thus \$987.50 + \$3,630 + \$9,988 + \$3,474 = \$18,079.50. Her **effective** tax rate is \$18,079.50/\$100,000 or about 18.1%.
- Bob earns \$40,125 as a sloth trainer. His income can be broken down into two parts:
  - The first \$9,875 incurs 10% of tax: \$9,875 × 0.1 = \$987.50
  - The part between \$9,875 and \$40,125 incurs 12% of tax: (\$40,125 - \$9,875) × 0.12 = \$3,630

  Bob's total tax owed is thus \$987.50 + \$3,630 = \$4,617.50. His effective tax rate is \$4,617.50/\$40,125 or about 11.5%.

Your task in this problem is to write a program that computes the federal tax owed for any salary. After you're done, maybe you can start your own company to compete with H&R Block!

(a) (6 points) *Before* writing a program, it can be helpful to work out some **test cases**: program inputs with known outputs. In addition to producing concrete test data to use later, the process of working out the test cases often provides insight into how to solve the problem. This technique is called **test-driven development**.

Neatly write or type how the tax owed *and* effective tax rate would be computed for three different incomes. Clearly show all of your calculations. You can pick your own numbers for the incomes, but they must be in three different tax brackets, and they must not repeat the Alice and Bob examples provided earlier.

(b) (12 points) Write a program named `tax_calculator.py` that takes user input for 2020 income earned and computes the tax owed and effective tax rate. Your program should also provide a bracket-by-bracket breakdown of how the tax owed was computed. Round off all monetary amounts to two decimal places, and round off the effective tax rate to one decimal place. Verify that your program works correctly using the earlier examples, as well as your own test cases.

Example program run (underlined parts indicate what the user enters)

```
What was your 2020 income? 100000

    First $  9875:      $    987.50
$  9875 - $ 40125:      $   3630.00
$ 40125 - $ 85525:      $   9988.00
$ 85525 - $163300:      $   3474.00

Total tax owed:         $  18079.50
Effective tax rate:          18.1%
```

Need Help?

- Attend your weekly lab meeting. Your lab TA can give you live help during this session.
- Contact your lecture instructor, and/or attend office hours.
- The UofM offers free online tutoring through the Educational Support Program (ESP): `https://www.memphis.edu/esp/onlinetutoring.php`

Submission Instructions

- Put your test cases from the income tax problem into a *single PDF file.* If you wrote this on paper, please scan them into a single PDF file. There are a number of free phone apps that you can use for this. I've had good experiences with CamScanner on Android; the software is available for iOS as well. *Points may be deducted for submitting in a format other than PDF.*
- Create a zip file containing your PDF file from above and all of your Python source files. You can use any Python development environment you like, as long as you submit the source files.
- Upload your zip file to the appropriate dropbox folder on eCourseware. The dropbox *will* cut off submissions at precisely the stated deadline, so please submit with some time to spare. Late submissions are not accepted. You may submit as many times as you want up to the deadline. Each submission overwrites the previous one.
- Contact your lab TA to schedule a one-on-one code review via Zoom. During this short meeting (15-30 minutes), your TA will ask you to run your code. They may also ask you to explain your code and/or run different test cases. The code review must be completed for you to get a grade for the lab.

  Note that since the code review happens after your submission, its purpose is *not* for you to get help with writing your code! Getting help should be done during the scheduled lab sessions, or by reaching out to your lecture instructor/online tutors *before* the due date.