Programming Assignment 1

Getting started

Complete the reading and practice assignments posted on the course schedule. Review class handouts and examples. Please remember that you can collaborate on the practice problems, but no collaboration is allowed on the Programming Projects. **The only help you are allowed to get on a Programming Project, must come from a CIS Sandbox tutor or the course instructor.**

This week's project requires knowledge of only some very basic operations on numeric values. However, it is not trivial due to its logic. In particular, it requires the use of integer division and the remainder (a.k.a. the mod) operator.

When working on a project always make sure you understand the program requirements first, then think about the algorithm that you will use and write a sketch of it in English. Only after you have thought through the details of the algorithm and verified it on a few test cases, start working on its implementation in Python. Don't forget to test your program using multiple different test cases.

When working on the program it is important to learn **to develop it gradually by implementing one logical step of the algorithm at a time and testing the program after implementing each step.**

Programming Project: SchoolBus **worth: 10 points**

When will the bus be at our stop?

Imagine that you are the owner of a company that provides transportation to the students of the local schools. Your customers have requested that the bus schedule and pricing information be available online. You have already hired an experienced app developer to provide an app front-end for this application, but now you need to implement the logic of determining the pricing and the arrival and departure time of the school bus to a specified location. The rules of the bus operation and the requirements on the program are described below.

The bus arrives at the first stop exactly 45 minutes prior to the time when the school starts. The bus spends 2 minutes at each stop waiting for the students to get on and 3 minutes traveling between any two consecutive stops. All school bus stops are numbered and each customer knows their stop number.

The cost of the school bus ticket equals $1 plus 15 cents for each full 4 minute interval spent traveling from the stop to the school. Note that the waiting time at the stop where the student gets in, is not charged, so, for instance the price of the ticket from stop number 1 equals (43//4)*15 + 100 = 260 cents, or 2.60. Here, 100 is the base price of the ticket in cents, and (43//4) denotes integer division, thus producing 10 (and not 10.75).

At first, the user will be prompted to enter

1. the time when the school starts in hours and minutes, based on the 24-hour clock, as two separate numbers: hours and minutes.
2. the stop number (between 1 and 9).

The program will then compute and output

1. The arrival and departure time for the stop. The time should be specified in hours and minutes based on the 24 hour clock.
2. The length of the trip (not including the wait time at the stop from which the student departs to school) in minutes.
3. The cost of the ticket in dollars.

Here's what the interaction should look like for the above example (user input appears in **boldface**)

```
Please enter the hour when school starts:  10
Please enter the minute when school starts: 30
Please enter your stop number: 5
The bus will be at stop number 5 between 10:05 and 10:07
The length of the trip from stop number 5 is 23 minutes
The cost of the ticket from stop number 5 is $1.75
```

Hint: The easiest way of dealing with time and time intervals is to convert everything to minutes and work with this representation. For example, the school start time of 10:30 can be converted to minutes as follows 10*60+30 = 630 minutes from the start of the day (midnight).

In the example above, the arrival time at the 5-th stop can be computed as follows. First, the bus arrives at the first stop at 630-45=585 (i.e. 9:45 in minutes). The bus arrives at the 5-th stop 4*(3+2)=20 minutes later (accounting for the past 4 stops). So, the arrival time to the 5-th stop in minutes is 585+20 = 605 minutes. To convert the minute representation back to hours and minutes, use the integer division and the remainder operators (// and %): 605//60=10, and 605%60=5.

Here is another sample interaction:

```
Please enter the hour when school starts: 8
Please enter the minute when school starts: 14
Please enter your stop number 7
The bus will be at stop number 7 between 7:59 and 8:01
The length of the trip from stop number 7 is 13 minutes
The cost of the trip from stop number 7 is $1.45
```

Important Note

Your program will be tested by a computer program before I evaluate it. The tester program is not intelligent enough to interpret the output line and deduce which part of it represents the times and which represents the cost. **Therefore the input and output of your program should appear in exactly the order that is shown and in the format that is shown in the sample interaction.**

Other Requirements: You must use named constants to represent

- the 45 minute length of the whole trip,
- the 3 minute length of the drive between two stops,
- the 2 minute length of each stop,

- the $1 base cost of the ticket, and
- the 4 minute increments by which the cost of trip is charged,
- the 15 cents, which is the cost for each 4-minutes of driving from the stop to the school.

Use these constants in the calculations. Then, if the duration of the drive, or costs of a ticket change, the only change required to make your program produce accurate values, would be to modify the values of these constants.

In this project you should not be using any loops, lists or any data structures not covered by the first week's material. Likewise, you should not worry about the user entering invalid data (for instance, invalid time or the stop number that exceeds 9). Your program will be tested with valid input only. Note, however, that your program must print the leading zero when specifying the minutes less than 10 (i.e. print 12:05 instead of 12:5). Also, make sure not to leave spaces between the numbers and the : character.

Remember that the best way to develop a program is by working on it incrementally and periodically verifying the correctness of each developed part. For instance, after developing the code that just gets all input information, implement and test the computation of the arrival time, then the departure time, the length of the trip to school and, finally, the cost.

You can verify that the intermediate values computed by your program (e.g. the number of full 4-minute intervals of the trip to school) are correct by printing them out. This technique is called *debugging output*. The extra printing statements must later be removed when the final version of the program is produced.

Grading

The grading schema for this project is roughly as follows:

Your program should compile without syntax errors to receive any credit. If a part of your program is working, you will receive partial credit, but only if the program compiles without syntax errors.

- 2 points will be awarded for correctly handling input and output
- 3 points will be awarded for the correct computation of the arrival and departure time,
- 2 points will be awarded for correctly computing the length of the drive to school, and
- 1 point will be awarded for correctly computing the price of the ticket.
- 2 points will be awarded for good programming style, as defined in handout 1.