ASSIGNMENT A1
SECTION E

Read the instructions below carefully. The instructions must be followed. This assignment is worth 5% of your grade. The assignment is due on Sunday 3rd of October Midnight. No late assignments will be accepted.

The goal of this assignment is to learn and practice (via programming) the concepts that we have learned so far: numbers, algebraic expressions, boolean expressions, strings and operations on strings, variables, Python's builtin functions, designing your own functions, and the use of Python's input and output functions. Before you can start this assignment, you need to know how to use a (IDLE's) text editor to edit python modules (files) as well as how to use python shell interactively to test your code. If you have no idea how to do these things watching video of the 3rd lecture, for example, will help. Submit your assignment by the deadline via Blackboard Learn (as instructed in the first lab.)
Each question asks you to design one function. Put all these functions (for all the questions) in ONE file only, called a1_xxxxxx.py (where xxxxxx is replaced with your student number). Within this file, a1_xxxxxx.py, separate your answers (i.e. code) to each question with a comment that looks like this:

```
##########################################################################
  # Question X
  ##########################################################################
```

In addition to a1_xxxxxx.py you have to submit a separate file called a1_xxxxxx.txt. What should be in that file is explained below. Thus for assignment 1 you have to submit **two files**: a1_xxxxxx.py and a1_xxxxxx.txt. Submit your assignment by the deadline via Blackboard Learn (as instructed in the first lab.)

Your program must run without syntax errors. The questions that have syntax errors will receive zero points. More specifically, for each of the functions below, I have provided one or two tests to test your functions with. For example, you should test question 1 by making function call `lbs2kg(1)` in the Python shell. To obtain a partial mark your function may not necessarily give the correct answer on these tests. But if your function gives any kind of python error when run on the tests provided below, that question will be marked with zero points.

The purpose of this assignment is to practice concepts that you have seen in the first 2 weeks of class.
Thus this assignment does not require use of loops, if and other branching statements, lists ... etc, except for question 10 (and Question 8 if you want to be creative). Thus

you must solve the questions below without loops, if and other branching statements, lists … unless explicitly stated otherwise in the question.

After reading each of these questions once, go to the "Testing your code" section below and see what the output of your function should give. In that section, you can find a couple of function calls and the required results for each question. Studying these example function calls will help you a lot to understand what each individual question requires, or rather what its function should do.
To determine your grade, your functions will be tested both with examples provided in Section "Testing your code" and with some other examples. Thus you two should test your functions with more example that what I provided below.

Section: Assignment 1 questions

1 (2 points) Write a function lbs2kg(w) that returns the given weight, w, expressed in pounds as weight in kilograms

2 (2 points) Write a function id_formater(fn, ln, appelation, city, year) that returns a string of the form "appelation. ln, fn (city, year)"

3 (2 points) Write a personalized limerick maker in the form of a function called limerick_maker(). This function prompts the user for the name and city of birth and then prints a limerick that includes that name and city of birth. You can make up your own limerick or adapt one from http://www.loonylimericks.com/. (Note, though, that many of these are not suitable for this question since they use the name or city of birth as part of the rhyme.)

4 (2 points) Write a function id_formater_display() that prompts the user for the first name, last name, appellation, place of birth, and year of birth. The function should then print the ID of the user using the same format as specified in question 2. (To do that your solution must use a function call to id_formater function from question 2).

5 (2 points) Write a function l2loz(w) that takes a non-negative number w as input and returns a pair of numbers (l,o) such that w = l + o/16 and l is an integer and o is a non-negative number smaller than 16.

6 (2 points) Write

6. (2 points) write function convert_days_to_YY_MM_DD(day), which get number of days as input and return the following string:
Year: YY, Month: MM, Day: DD
Where YY, MM and DD is the number of years, months and days converted by the function.

Note: Assume that the year = 360 days and the month = 30 days.

7 (2 points) The *median* of a group of numbers is the number from that group that has the property that at least half of the elements in the group are smaller or equal

to it and at least half of the elements in the group are bigger or equal to it. In a sorted list of numbers, the median can be found in the middle of the list (well, the middle is only well defined if the list has odd number of elements). For example, the median of this group of numbers: 10,11,13,15,16,23,26 is 15. Median of this group of numbers: 7,1,3 is 3. The median of this group of numbers: 2,2,5 is 2. Write a function median3(num1,num2,num3) that prints a message for each of the given three numbers, num1, num2 ,num3, stating if the number is a median. See the test runs below to understand in what format should the function median3 print its results.

8 (2 points) Write a function below_parabola(a,b,p,q) that returns True if the point (p,q) in the plane (i.e., the point with x-coordinate p and y-coordinate q) is **below** or **on** the graph of the parabola $y=ax^2 + b$, otherwise it returns False. You may assume that your function will be tested only with positive numbers for a (and arbitrary numbers for b).

9 (2 points) Suppose that a grading scheme for a course with 3 assignments, one midterm and one final is: each assignment is worth 5%, the midterm is worth 35% and the final is worth 50% of the grade. Write a function projected_grade(a1,A1,a2,A2,m,M) that predicts (i.e. computes) the final grade percentage as follows. The student obtained a1 points out of maximum A1 for assignment 1; a2 points out of maximum A2 for assignment 2 and m points out of maximum M for the midterm. To compute the final course grade, assign to assignment 3 the average percentage of assignment 1 and assignment 2 and assign to the final exam the same the percentage as obtained on the midterm exam. For example, if the student got 12/20 in assignment 1 and 24/24 in assignment 2, the grade in assignment 3 should be 80% since that is the average of the two percentanges. If the student obtained 9/12 in the midterm, the grade of the final exam should also be 75%. Using the given grading scheme and the predicted grades for assignment 3 and the final exam, function projected_grade should return the predicted final percentage for the course. You can assume that the function will be tested with reasonable values a1,A1,a2,A2,m, and M.

10 (2 points) Write a function called projected_grade_v2() that prompts the user to enter 6 numbers (the meaning of these six numbers is the same as a1,A1,a2,A2,m,M in the previous question). Compute the predicted percentage of the final course grade with the slight modification to the grading scheme where if the average on the midterm and the final is less than 50% the final course grade percentage is whatever is smaller of the two grading schemes. Print a meaningful message to a user about their predicted final grade. You may (or rather should) use if statements in this question.

11 (4 points) Suppose that a cashier owes a customer some change and that the cashier only has quarters, dimes, nickels, and pennies. Write a function that determines the minimum number of coins that the cashier can return. In particular, write a function change_to_coins(amount) that returns four numbers (q,d,n,p) that represent the smallest number of coins (quarters, dimes, nickels, and pennies) that add up to the given amount of dollars. You may assume that the input to function change_to_coins(amount) is a float value referred to by amount representing the

number of dollars. Before doing anything else, you should convert this number entirely to cents (that should be of type int). Once you have the total number of cents here are some hints on how to find the minimum number of coins. Hints for your solution (algorithm):

- To find the minimum number of coins the, so called, greedy strategy (i.e. greedy algorithm) works for this problem. The greedy strategy tries the maximum possible number of the biggest-valued coins first, and then the 2nd biggest and so on. For example if customer is owed 1.42 dollars, thus 142 cents, the greedy strategy will first figure out how many quarters can it give to the customer at most. In this case, it would be 5 quarters. It cannot be 6 as that equals 1.5 and the cashier would return too much money. Once the cashier returns 5 quarters, he/she still needs to return 17 cents. The next biggest coin after quarter is a dime. So the greedy strategy would try dimes next. Only one dime can fit in 17 cents, so the cashier should next return 1 dime. Then there is 7 cents left. The next biggest coin to consider is a nickel ... etc. It can be proved that this greedy strategy gives an optimal solution (i.e. the smallest number of coins) for this version of the problem. Thus for this question, you are asked to implement this strategy to find the optimal solution.
- *Side note:* in the Canada (and most other) coin systems, a greedy algorithm of picking the largest denomination of coin which is not greater than the remaining amount to be made will always produce the optimal result (i.e. give the smallest number of coins). This is not automatically the case, though: if the coin denominations were 1, 3 and 4 cents then to make 6 cents, the greedy algorithm would choose three coins: one 4-cent coin and two 1-cent coins whereas the optimal solution is two 3-cent coins.

Section: Testing your code

We would like to see evidence that you have tested your functions in python shell, like we did in class. Do this by opening your assignment with IDLE and then testing each of your functions individually. Then copy and paste the python shell output into a text file called a1_xxxxxx.txt The contents of a1_xxxxxx.txt should have something like this in it:

```
>>> # testing Question 1
>>>
>>> lbs2kg(1)
0.453592
>>>
>>> lbs2kg(130.2)
59.05767839999999
>>>
>>> lbs2kg(5)
2.26796
>>>
>>> # testing Question 2
```

```
>>>
>>> id_formater("Albert","Einstein", "Dr", "Bern", 1879)
'Dr. Einstein, Albert (Bern, 1879)'
>>>
>>> # testing Question 3
>>>
>>> limerick_maker()
Enter your name: Vida
Enter your city of birth: Mostar

Vida had funny funny hair
With tons and tons to spare
Vida's clippings made a wig
It was very big
And caused the townsfolk of Mostar to stare
>>>
>>> # testing Question 4  >>>
>>> id_formater_display()
What is your first name? Harry
What is your last name? Potter
What is your appellation? Wizard
Where were you born? Godric's Hollow
What is your year of birth? 1980

Wizard. Potter, Harry (Godric's Hollow, 1980)
>>>
>>> # testing Question 5
>>>
>>> l2loz(7.5)
(7, 8.0)
>>>
>>> l2loz(9.25)
(9, 4.0)
>>>
>>>
>>> # testing Question 6
>>>
>>> convert_days_to_YY_MM_DD(1000),
>>> Year: 2, Month: 9, Day: 10
>>>
>>> # testing Question 7
>>>
>>> median3(18, 1, 8)
18 is a median. That is  False
```

```
1 is a median. That is  False
8 is a median. That is  True
>>>
>>> median3(1, 8, 1)
1 is a median. That is  True
8 is a median. That is  False
1 is a median. That is  True
>>>
>>>
>>> # testing Question 8
>>>
>>> below_parabola(1, 0, 0, 0) True
>>> below_parabola(1, 0, 1, 1) True
>>> below_parabola(1, 0, 1, 2)
False
>>> below_parabola(2.5, 0, 1, 2)
True
>>>
>>> # testing Question 9
>>>
>>> projected_grade(10, 10, 15, 15, 30,30)
100.0
>>> projected_grade(12, 20, 24, 24, 9,12)
75.75
>>> projected_grade(18, 20, 16, 19, 60,100)
64.06578947368422
>>> projected_grade(19, 20, 16, 16, 49,100)
56.275

>>>
>>> # testing Question 10
>>>
>>> projected_grade_v2()
How many points did you get in Assignment 1? 10
What was the maximum possible number of points for Assignment 1?
10
How many points did you get in Assignment 2? 9
What was the maximum possible number of points for Assignment 2?
10
How many points did you get on the midterm? 49
What was the maximum possible number of points for the midterm?
100
Your predicted final grade is 49.0 %
>>> projected_grade_v2()
```

```
How many points did you get in Assignment 1? 10
What was the maximum possible number of points for Assignment 1?
10
How many points did you get in Assignment 2? 9
What was the maximum possible number of points for Assignment 2?
10
How many points did you get on the midterm? 0
What was the maximum possible number of points for the midterm?
100
Your predicted final grade is 0.0 %
>>> projected_grade_v2()
How many points did you get in Assignment 1? 10
What was the maximum possible number of points for Assignment 1?
10
How many points did you get in Assignment 2? 17
What was the maximum possible number of points for Assignment 2?
17
How many points did you get on the midterm? 100
What was the maximum possible number of points for the midterm?
100
Your predicted final grade is 100.0 %
>>>
>>>
>>> # testing Question 11 >>>
>>> change_to_coins(1.15) (4,
1, 1, 0)
>>> change_to_coins(2.02)
(8, 0, 0, 2)
>>>
>>> change_to_coins(0.37)
(1, 1, 0, 2)
>>>
```