CMPUT274—Assignment 2 (Fall 2025)

R. Hackman

Due Date: Friday October 10th, 8:00PM

Per course policy you are allowed to engage in *reasonable* collaboration with your classmates. You must include in the comments of your assignment solutions a list of any students you collaborated with on that particular question.

For each assignment you may not use any Python features or functions not discussed in class! This means, for example, you may not use loops, casting, any built-in functions or methods we have not discussed. If you are in doubt, you may ask on the discussion forum — however remember if you are including your code, or even your plan to solve a question, you must make a private post. Use of disallowed features will result in a grade of zero on the question(s) in which they were used.

All functions you write must have a complete function specification as presented in the course notes. Failure to provide function specifications for functions will lead to the loss of marks.

Unless explicitly forbidden by a question, you are always allowed to write additional helper functions that may help you solve a problem. In fact in some cases it will be *necessary* to write helper functions!

New Language Feature: The `cmput274` module has a new feature in it for your use: the function `LL`. The function `LL` takes *any* amount of parameters and produces the `LList` with those values in the order they were provided in the argument list. The `LL` function can be very helpful for writing test cases, especially for those that require the comparing of long `LLists`

1. **LList Cleaning** in this question you will write the Python function `removeOccurrences` which has two parameters. The first parameter, `l`, is an `LList` of any values. The second parameter, `toRemove`, is any value. Your function must return a `LList` which is the result of removing all occurrences of `toRemove` from `l`. For example:

   ```
   removeOccurrences(LL(1, 5, 10, 22, 10, 33), 10) -> (1, 5, 22, 33)
   ```

   You are again provided a skeleton file `q1.py` to fill in. This provides some basic test cases already written for you, as well as empty function definition. While there are two basic test cases provided, they are insufficient to effectively test your function. In addition to filling in the function definition you should write your own test cases to be confident your solution works.

   Once again, you should not edit any files in the repository itself, but instead make your own copy of it to edit. To make a new directory and copy the file for this question into that new directory you can follow the commands below:

   ```
   cd
   mkdir myA2
   cd myA2
   cp ~/F25-CMPUT274/a2/provided/q1.py ./
   ```

   Deliverables: For this question include in your final submission zip the python file `q1.py`

2. **Dictionary Lookup** In this question you will write a Python function `lookup` which has three parameters the first of which, `name` is a string, the second parameter, `names`, is a `LList` of strings, and the third parameter, `values` is a `LList` of any. Both of the parameters must have the same length, and have the following property

   - `names` and `values` *must* be the same length
   - The items in `names` are the names of some given entities
   - The items in `values` are the values of some given entities
   - The item at position `i` of the `names` `LList` has value of the item at position `i` of `values`

   Lists that have these properties are called *parallel lists* and are a rudimentary data type that is very versatile in how it can be used.

   Complete the function `lookup` so that it returns a `LList` with one item in it which is the value associated with the given `name`, or the empty `LList` if the value is not found. That is,

   - If the given `name` exists in the `names` `LList` at position `i` then return the `LList` that contains one item — the item in the `values` `LList` at position `i`.
   - If the given `name` does not exist in the `names` `LList` then return the empty `LList`

   Once again, you should not edit any files in the repository itself, but instead make your own copy of it to edit. If you have followed all instructions in the previous questions so far, then you can make a copy of this file by running the commands below:

   ```
   cd ~/myA2
   cp ~/F25-CMPUT274/a2/provided/q2.py ./
   ```

   Deliverables: For this question include in your final submission zip the python file `q2.py`

3. **Splitting Strings** In this question you will be writing a function named `strSplit` which takes two parameters, both of which are strings, and returns a `LList` of strings which is the result of "splitting" the string on all occurrences of the given separator. For this question the function specification is provided for you below.

```
def strSplit(s, sep):
  '''
  strSplit splits the string s on all occurrences of sep
           and returns a LList of the substrings of s
           that are separated by sep

  s       - a string
  sep     - a string
  returns - LList of strings

  Examples:
    strSplit("123,abcdf,hello", ",") -> ("123", "abcdf", "hello")
    strSplit("no split found", ",") -> ("no split found")
    strSplit("no split found", " ") -> ('no', 'split', 'found')
    strSplit("HelloxyFriendxHowxyAreYou?", "xy") -> ('Hello', 'FriendxHow', 'AreYou?')
  '''
  ...
```

Once again, you should not edit any files in the repository itself, but instead make your own copy of it to edit. If you have followed all instructions in the previous questions so far, then you can make a copy of this file by running the commands below:

```
cd ~/myA2
cp ~/F25-CMPUT274/a2/provided/q3.py ./
```

Deliverables: For this question include in your final submission zip the python file `q3.py`

4. **Integer detection** In this question you will write the function `isIntegerStr` which takes one string parameter and returns `True` if that string represents an integer and `False` otherwise. A string represents an integer if:

   - All the characters in the string are digit characters
   - Except, optionally, the first character which may be the `-` character instead of a digit.

   For example:

```
isIntegerStr("456") -> True
isIntegerStr("954x6") -> False
```

Hint: How can the `ord` function help you detect if a character is a digit or not?

Once again, you should not edit any files in the repository itself, but instead make your own copy of it to edit. If you have followed all instructions in the previous questions so far, then you can make a copy of this file by running the commands below:

```
cd ~/myA2
cp ~/F25-CMPUT274/a2/provided/q4.py ./
```

Deliverables: For this question include in your final submission zip the python file `q4.py`

5. **String to Integer** In this question you will write the function `strToInt` which takes one string parameter that is guaranteed to be a string that represents an integer. Your function must return the corresponding `int` that the string represents. For example:

   ```
   strToInt("-432") -> -432
   strToInt("1234") -> 1234
   ```

   Once again, you should not edit any files in the repository itself, but instead make your own copy of it to edit. If you have followed all instructions in the previous questions so far, then you can make a copy of this file by running the commands below:

   ```
   cd ~/myA2
   cp ~/F25-CMPUT274/a2/provided/q5.py ./
   ```

   Deliverables: For this question include in your final submission zip the python file `q5.py`