THE UNIVERSITY OF THE WEST INDIES
Department of Computing
COMP1127–Introduction to Computing II

Lab 6

The following Lab Exercise SHOULD BE DONE in IDLE or another Integrated Development Environment (IDE).
The Exercise SHOULD NOT BE DONE on Hackerrank.

Opening Date: **Friday, November 25, 2022**

Lab Date (Week 4): **Monday, November 28 – Saturday, December 3, 2022**
Due Date: **11:45 pm, Sunday, December 4, 2022 (on OurVLE)**

The code for most functions for the Student Database has been provided in **lab6_code.py**, **lab6_adt.py** and **lab6_util.py**. Download these 3 files and add your code for the problems before uploading the 3 files on OurVLE. Lab6_code.py is for the main program code (lab6_adt and lab6_util are imported within lab6_code); lab6_adt.py contains the student ADT; lab6_util.py contains user-defined higher order procedures such `my_map`, `my_filter`, and `foldr`.

A student record contains a tuple of a tag 'student' and the student data as a list. The student data contains the id number, name, date of birth, number of courses being done by the student, and a list of the student's courses. **The id number is an integer.** The name is a list of a first name and a last name; to create a name, one must call the `make-name` procedure with the first name and last name. The date of birth is a tuple of year, month and day created with the `make-date` constructor. The list of student courses includes tuples of course and grade.

In this lab, `make_student` is to be defined within Problem 1. The other functions have been provided:

Type	Name	Description
Constructor	**`make_student()`**	**Creates a student as a tuple of a tag 'student' and a student record. The student record is a list with an id, name, date of birth, number of courses, and a list of course-grade pairs.**
Constructor	`make_name()`	Creates a name as a list with first name and last name.
Constructor	`make_date()`	Creates a date as a tuple of year, month, day.
Constructor	`make_course()`	Creates a student with an id, first name and last name as a list and the course codes and grades as a list of tuples.
Selector	`stud_data()`	Returns a list with the full student record
Selector	`student_id()`	Returns the id value from the given student record.
Selector	`student_name()`	Returns student name as a list of first and last names from the student record

Selector	`student_dob`	Returns student date of birth as a tuple of birth year, month, day from the student record
Selector	`student_no_of_courses()`	Returns number of courses being taken by a student
Selector	`student_courses()`	Returns a list of tuples where first part of the tuple is the course and the second part is the corresponding grade.
Selector	`student_fname()`	Returns student's first name as a string given the student name list.
Selector	`student_lname()`	Returns student's last name as a string given the student name list.
Selector	`get_ccode()`	Returns the course code from a tuple of code and grade.
Selector	`get_grade()`	Returns the grade from a tuple of course code and grade.
Selector	`date_yyyy()`	Returns the year from a tuple of year, month, day
Selector	`date_mm()`	Returns the month from a tuple of year, month, day
Selector	`date_dd()`	Returns the day from a tuple of year, month, day

In **lab6_code.py**, already defined as shown below (and awaiting the definition of `make_student` in Problem 1), there is a list `slist` of 10 students `st1, st2, st3, st4, st5, st6, st7, st8, st9, st10` with ID numbers `62000050, 62000001, 62000035, 62000021, 62000034, 85000050, 90000001, 95000035, 99000021, 92000034` and courses `COMP1126, COMP1127, COMP1210, COMP1220, MATHS1142, MATHS1152, PHYS1421.`

```
st1 = make_student(62000050,"Jane","Doe",1995,12,25)
st2 = make_student(62000001,"John","Brown",1990,7,6)
st3 = make_student(62000035,"Jack","Green",1999,1,3)
st4 = make_student(62000021,"Chris","Brown",1992,7,10)
st5 = make_student(62000034,"Joe","White",2000,4,20)
st6 = make_student(85000050,"Janet","Dopa",1965,12,25)
st7 = make_student(90000001,"Johnathan","Browning",1970,7,6)
st8 = make_student(95000035,"Jackie","Greene",1979,1,3)
st9 = make_student(99000021,"Christine","Black",1982,7,10)
st10 = make_student(92000034,"Joette","Whiteley",2000,4,20)
slist= [st1,st2,st3,st4,st5,st6,st7,st8,st9,st10]
add_course(slist,62000050,'COMP1126','B+')
add_course(slist,62000050,'COMP1127','A')
add_course(slist,62000050,'COMP1210','C+')
add_course(slist,62000050,'COMP1220','A-')
add_course(slist,62000001,'COMP1126','B')
add_course(slist,62000001,'COMP1127','A+')
add_course(slist,62000001,'COMP1210','C')
add_course(slist,62000001,'COMP1220','B-')
```

```
add_course(slist,62000001,'MATH1142','A-')
add_course(slist,62000035,'COMP1126','A+')
add_course(slist,62000035,'COMP1127','A')
add_course(slist,62000035,'MATH1142','C+')
add_course(slist,62000035,'MATH1152','F3')
add_course(slist,62000021,'PHYS1421','F1')
add_course(slist,62000021,'COMP1127','A-')
add_course(slist,62000021,'COMP1210','C')
add_course(slist,62000021,'COMP1220','B+')
add_course(slist,62000021,'MATH1142','A-')
add_course(slist,62000034,'COMP1126','A-')
add_course(slist,62000034,'COMP1127','C')
add_course(slist,62000034,'COMP1220','B+')
add_course(slist,90000001,'COMP1126','B')
add_course(slist,90000001,'COMP1127','A+')
add_course(slist,95000035,'COMP1210','C')
add_course(slist,92000034,'COMP1220','B-')
add_course(slist,92000034,'MATH1142','F2')
add_course(slist,92000034,'COMP1127','F3')
```

Problem 1

In **lab6_adt.py**

a. A student record contains a tuple of a tag 'student' and the student data as a list. Define the constructor `make_student` that takes arguments student id, first name, last name, date of birth year, month, day and returns a student record. The function uses `make_name` to create the name list to be included in the record, uses `make_date` to create the date of birth tuple, inserts 0 for the number of courses, and empty list for the list of courses.

   The constructors `make_name(fname,lname)`, and `make_date(yyyy,mm,dd)` have already been defined.

In **lab6_code.py**

b. Define the following 2 dictionaries `course_list` and `grade_list` that has data concerning courses and credits, as well as each grade and its gpa, respectively. Add the code in the labeled section within *"Valid Courses and Grades"* for these dictionaries.

```
course_list={'COMP1126':3,'COMP1127':3,'COMP1161':3,
   'COMP1210':3,'COMP1220':3,'MATH1142':3,'MATH1152':3,
   'PHYS1411':3,'PHYS1412':3,'PHYS1421':3}

grade_list = {"A+":4.3,"A":4.0,"A-":3.7,"B+":3.3,"B":3.0,
   "B-":2.7,"C+":2.3,"C":2.0,"F1":1.7,"F2":1.3,"F3": 0.0}
```

Problem 2

In **lab6_code.py**

a. Write the function called `print_courses` that has no arguments. The function prompts the user to enter a valid student id number, checks if it is valid using `is_valid_student_id(slist,sid)`, gets the position of the student record using `student_direct_pos(slist,sid)`, gets all student courses using `student_courses`, then prints out these courses and grades. Add code in the labeled section within *"MAIN MENU Procedures"* for this function.

e.g. `print_courses()`

```
Enter Student ID No to Display Student Courses : 62000050
Student Id:  62000050
     Course:  ('COMP1126', 'B+')
     Course:  ('COMP1127', 'A')
     Course:  ('COMP1210', 'C+')
     Course:  ('COMP1220', 'A-')
```

b. Write the function called `early_birthdays` that takes a list of student records as an argument, uses `filter` (either python pre-defined or user-written `my_filter` in lab5_util.py), and returns a list of those records whose dates of birth falls within the first 6 months of the year. Add code in the labeled section within *"QUERIES MENU Procedures"* for this function.

e.g. `early_birthdays(slist)`

```
[[62000035, ['Jack', 'Green'], (1999, 1, 3), 4, [('COMP1126', 'A+'),
('COMP1127', 'A'), ('MATH1142', 'C+'), ('MATH1152', 'F3')]],
[62000034, ['Joe', 'White'], (2000, 4, 20), 3, [('COMP1126', 'A-'),
('COMP1127', 'C'), ('COMP1220', 'B+')]], [95000035, ['Jackie',
'Greene'], (1979, 1, 3), 1, [('COMP1210', 'C')]], [92000034,
['Joette', 'Whiteley'], (2000, 4, 20), 3, [('COMP1127', 'F3'),
('COMP1220', 'B-'), ('MATH1142', 'F2')]]]
```

c. Recall that `foldr` takes a combiner function, a base case, and a list and returns a single value.

e.g.
```
foldr (sum,0,[1,2,3,4])   => 1+ (2+ (3+ (4+ 0)))  =>   10
foldr (product,1,[2,3,4]) => 2* (3* (4* 1))       =>   24
```

Write the function `tot_no_of_courses` that takes a list of student records as an argument, uses `map` and `foldr`, and returns the summation of the number of courses for all students. Add code in the labeled section within *"QUERIES MENU Procedures"* for this function.

e.g. `tot_no_of_courses(slist)`

```
27
```

d. Write the function `largest_id_no` that takes a list of student records as an argument, uses `map, foldr, max` and returns the student id number that is the largest within the student records. Add code in the labeled section within *"QUERIES MENU Procedures"* for this function. Hint: **max(5,3) → 5, max(2, 7) → 7**

e.g. `largest_id_no(slist)`
`99000021`

e. Include a call that begins execution of the Student Database by invoking
`show_menu(slist)`
Add code in the section labeled *"LAUNCH PROGRAM"*
Test the options reflected on the Main Menu page and the Queries Menu Page.