Concordia University
Department of Computer Science and Software Engineering
COMP 348: Principles of Programming Languages
Fall 2017
Assignment 3
Evaluation: 100 points
(5% of your final grade)
Due date and time: Monday December 4th,2017 at 23:59

I. Object Programming with Ruby

Question 1- (10 pts)

Write a Ruby method *double* that given an array of integers *int_values*, uses the `Array.each`, `Array.sort` method and code blocks to print each array value and its corresponding double value, in sorted ascending order. For instance, given the array [125, 200, 25, 150, 50], your code should print out the following:

```
25    50
50    100
125   250
150   300
200   400
```

Question 2 (15 pts)

a) Write a Ruby method for extracting the digits from Quebec Heath Insurance Card ID called *HID_num* that given a string *str*, uses regular expressions and back references to find the first valid card number in the string. A person valid health care number has 4 uppercase letters, followed by 8 digits number (e.g.: RENL18347692).The method should return the individual number as an integer value, or *nil* if no individual valid card number is found.

Examples:

```
HID_num ("SALZ18347692 SALZ62487931") # returns 18347692
HID_num ("HOuN12345678 HOUN24681357") # returns 24681357
HID_num ("ChAP62148730 CHA62148730") # returns nil
```

b) Write a Ruby program that reads the name of a text file from the command line, opens the file, reads every line of text in the file, that is a string pass it the method *HID_num* and prints only the lines that return an accepted number. If your text file contains the three lines above it should only give the first two returns

Question 3 (25 pts)

Write a Ruby program in `inventory.rb` that reads an inventory database file where each line contains an item ID, name, category, and expired date. Your program must process this file and display two lists. The first list displays each category (sorted by category name). For each category, display on a separate line the name of the category, the number of items in the category, and each item name in the category (this list is sorted by category name). The second list displays each item (sorted by item name). For each item, display on a separate line the item's ID, item name, and expiry date (sorted by item name). Your program will be called with the name of the database file as an argument. For example, running your program on a file called data.txt ("`ruby inventory.rb data.txt`") could generate:

```
% data.txt
0002: Bread,CERL, 28NOV17
0004: Milk, DAIR,07DEC18
0003: CheeseMozar, DAIR, 01SEP18
0008: PotatoeRed,VEGT,23NOV17
0004: YOGORT, DAR,03JAN18
0008: PotatoeRed,VEGT,12DEC17
0002: Bread, REFF, 28JUN19
0005: Bacon, REFF,10JUL20
0002: Bread, REFF
```

```
%ruby inventenry.rb data.txt

ERROR 0004: YOGORT, DAR,03JAN18
ERROR 0002: Bread, REFF
CATEGORIES
CERL,1, Bread
DAIR,2, CheeseMozar,Milk
REFF,2, Bacon, Bread
VEGT,1 PotatoeRed
ITEMS
0005: Bacon, 10JUL20
0002: Bread, 28NOV17, 28JUN19
0003: CheeseMozar, 01SEP18
0004: Milk, 07DEC18
0008: PotatoeRed, 23NOV17
```

Item IDs must be coded on 4 digit starting from 0000 to 9999, and are separated from item names by a colon and a space. Item names must be composed of 1 or more lowercase and uppercase characters. Category names must be exactly four uppercase letters. Expiry date must be composed of two valid digits for the day of the month and three valid upper case letter of the month name and the last two digits of the year. Category names and expiry date are separated by commas. Lines that do not follow this format should produce an error message.

Items may have an arbitrary number of expiry dates. Each item may only be counted once for each category with the closest expiry date, even if they have multiple expiry date within the category (e.g., `PotatoeRed` only counts as 1 item for `VEGT`).You may assume each item has a single unique ID.

While reading in the file, for each invalid line found, your program should output ERROR followed by the invalid line. Next, it should output "CATEGORIES", followed by the information for each category (in sorted order by category name). Finally, it should output "ITEMS", followed by the information for each item (in sorted order by item name).

Note: You can make use of (a) Code block, (b) pattern matching, and (c) the sort build-in function of Ruby.

II. Procedural programming with C

Question 4 (10 pts)

Write a C function *twoStrCompr* that takes two arguments of type string. The function *twoStrgCompr* should return "True" if the first string is alphabetically smaller than the second string argument, "False" otherwise. You may assume that the two strings arguments contains only upper case letters, or lower case letter but not both, and no blanks or other no alphabetical characters is allowed. Test your function with a suitable main program. Once working properly, convert the function *twoStrgCompr* to pointer arithmetic syntax, and check that it still behaves in the same way.

Question 5 (20 pts)

a) Write a C program that: creates and prints out a linked list of strings. Similar to java your program should include the node struct. Test your program for different lists of string.

b) Add the following functions to the program above:

- `void add_after(node_ptr &list, char a_word[], char word_after[])`

which inserts a node containing "`word_after`" in the linked list "`list`", after the first occurrence of a node containing "`a_word`". If "`list`" does not contain such a node, the function leaves it unchanged.

- `void delete_node(node_ptr &a_list, char a_word[])`

which deletes the first node in "a_list" which contains "a_word".

- `void list_selection_sort(node_ptr &a_list)`

which sorts a list alphabetically (use your answer to Question 4 to help).

sample input/output might be:

```
Enter first word (or '.' to end list): though
Enter next word (or '.' to end list): actions
Enter next word (or '.' to end list): speak
Enter next word (or '.' to end list): louder
Enter next word (or '.' to end list): than
Enter next word (or '.' to end list): words
Enter next word (or '.' to end list): .

THE LIST IS NOW:
though actions speak louder than words

AFTER WHICH WORD WOULD YOU LIKE TO ADD AN EXTRA WORD? though
WHICH WORD WOULD YOU LIKE TO ADD? many

THE LIST IS NOW:
though many actions speak louder than words

WHICH WORD WOULD YOU LIKE TO DELETE? than

THE LIST IS NOW:
though many actions speak louder words

AFTER SORTING, THE LIST IS:
actions louder many speak though words
```

Question 6 (20 pts)

a) Similar to assignment 2, but now using C program, compute series for functions ln (1+ *x*) and e^x depending on a switch variable *s*. Write a function that takes in three arguments e.g. function *compute-trig*(*x*, *s*, *n*), where

$$\boldsymbol{compute-trig(x,s,n)}\begin{cases} e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots \frac{x^n}{n!} & \boldsymbol{s=1} \\ \ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots \frac{x^n}{n} & -1 < x < 1, \quad \boldsymbol{s=2} \end{cases}$$

s is the switch variable that decides which function to evaluate, *n* is the precision variable and *x* is the parameter variable. Parameters *s* and *x* should be defined as keyword parameters and *n* should be an optional parameter with default value of *n*=5. You are not allowed to use any inbuilt functions except functions to check value type. Your program should print an apt error for string and decimal values for *s* and *n*. Additionally, there is a limit for *x* with *s*=2.

b) Compare between Lisp and C paradigms to compute the functions above, which one you will recommend to learn first for a freshman with no computing background? why?

Submission:

- **Assignment must be done individually (no groups are permitted).**
  - Create one zip file, **containing all the source files for each question of the assignment.**
  - Name your zip file this way:*a1_studentID*, where *studentID* is your ID number, for the second assignment, student 123456 would submit a zip file named a3_123456.zip
- Assignments must be submitted through the course web page(EAS) or Moodle site of the course (please check your section**).**

Note: **Assignment not submitted by the due date and in the correct format will not be graded – NO EXCEPTIONS!!!!**