UNIVERSITY OF THE WEST INDIES
CAVE HILL CAMPUS
Department of Computer Science, Mathematics & Physics

**COMP2611 - Data Structures**

**Project Assignment – Part II**

**DEADLINE DUE DATE : Sunday 15 April, 2018 – at Midnight.**

## Objective:

In this second (and final) part of your project assignment, you will now add the functionalities to your project as was defined in Part I of this project.

**Remember that your program should be created using wxWidgets and should be executed in Linux OpenSUSE!**

Replace the GUI caption from Part I ("**COMP2611 – Students Registration Database"**) with "**COMP2611 – Pelican Travels Database"** instead**.**

e.g.

| | | |
|---|---|---|
| **Ready…** | **Dr. John Charlery** | **123456789** |

## Method:

1. A text file (Clients.dat) is provided with records of clients who booked their travels with the Pelican Travel Agency. The records contain fields of:
   Client number
   First and surnames
   Type of booking (web, telephone, walk-ins, other)
   Time of booking (Spring, summer, fall, winter)
   Trip destination.

2. Using the text file as input, your program is required to read the individual records from the file and to populate the ADTs with the entire record, using the client number as the key field, in this manner:

| | | | |
|---|---|---|---|
| (i) | **Queue** | - | All records |
| (ii) | **Deque** | - | All records (alternating enqueue operations) |
| (iii) | **Priority Queue** | - | All records |
| (iv) | **Stack** | - | Spring travels |
| *(v)* | ***Binary Search Tree*** | - | *All the records* |
| (vi) | **AVL Tree** | - | Fall travels |
| (vii) | **Binary Heap Tree** | - | Summer travels |
| (viii) | **Red-Black Tree** | - | Winter travels |
| (ix) | **SetA** | - | CARICOM destination |
| (x) | **SetB** | - | Other (non-CARICOM) destinations |

   The input file should **NOT** be hard-coded into your program. Where sort order is required, it should always be noted in this assignment that sort means **Ascending Order**.

3. The **purist** definition for the Binary Search Tree is being used here; hence node duplication is not permitted and your code should reflect this.

4. In the display functions, the records should be displayed one per line.

5. The nodes in the AVL tree must contain an attribute to describe the weight of the node (i.e. negative for **left-heavy**, positive for **right-heavy** and zero for **balanced**), which must be displayed in brackets at the end of the line when the AVL tree is traversed.

6. The nodes of the Red-Black tree must contain an attribute to describe the node's colour, which must also be displayed when the Red-Black tree is traversed.

7. For all the **tree ADTs**, when their nodes are displayed, the client number of the left and right children should also be indicated in square brackets as [***left, right***]. If there is no left and/or right child, then ***NULL*** should be used in the appropriate space. You should use the display output provided in the BST code as you guide.

8. The output results of all the menu functions should be displayed in the main text box within your GUI.

9. When the file is opened, its contents should be **immediately** displayed in that main textbox as well as when the menu option for **Display File** is clicked.

10. The **File** menu item should have the sub-menus of **Open File**, **Save File**, **Save As** and **Exit**. When **Open File** is clicked, the **system fileOpen dialog** should be opened with the option to display files of type: **Data (*.dat), Text (*.txt)** and **All (*.*).** The user must also be able to type in the file name in the filename textbox. Once the contents of a file are displayed or the result of some processing is displayed, the menu selections of **Save File** and **Save As** should open the corresponding dialogs to perform the desired task. **Save As** should allow the user to specify a file name and file type into which the contents can be saved. These two functions should only save the contents of the main textbox.

11. When the **Display File** menu item is clicked, the entire content of the text file should be displayed in the display textbox if a file has been previously opened. If no file is open, then a dialog box with an appropriate error message should be activated.

12. Before each display operation is carried out, the display textbox should be cleared.

13. In the **Set** menu item, the **Create Set** should allow for the creation two sets – one set with bookings made online and the other set for all other bookings. The other sub-menu items for **Set** and all the ADTs are self-explanatory.

14. The operation indicated by the sub-menu item should then be carried out on that particular ADT **ONLY**. The other ADTs should NOT be affected by the operations on another ADT.

15. The **Exit** menu item is to close the program

16. The **About** menu item should produce a dialog box with suitable information about the programmer, the program, and the architecture of the machine the program is running

on.

17. When the cursor is placed on a sub-menu option, a description of the menu option should be displayed in the first partition of the status bar. At all other times, the string, "**Ready…**" should be displayed.

## SUBMIT:

Your project should contain a separate header file for each of the ADTs (i.e. Queue.h, PQueue.h Deque.h, Stack.h, BSTree.h, AVLTree.h, RBTree.h, SplayTree.h, HeapTree.h and Set.h) as well as a source code file (**.cpp**) which contains your progam. All the files should be submitted as a **ZIP** file with your ID number as the name (e.g. **123456789.zip**) through the portal on this course's page on eLearning. You may also use the portal to continually save your zipped project as you are developing your code. Previous versions of your submissions will be overwritten and only the latest version will be kept.

You may use any version of Linux to create your code but **Double-check** that your program code can be compiled in **OpenSUSE on the machines in CSL1**.

**DEADLINE DUE DATE :**
**Sunday 15 April, 2018 - No Later Than Mid-Night.**

**No late assignments will be accepted – No extension will be granted!**

**Tips:**

1. Start working on your project immediately and plan to finish at least a week in advance of the deadline date. Avoid, at all costs, trying to submit your assignment on the day of the deadline.
2. Build your code incrementally. Add one functionality at a time. At every stage in your code development, you should have a working project. In the unlikely event you did not complete the assignment by the due date, submit what you have completed and make sure those parts are working. **Projects that do not compile will be given an automatic zero and will not be assessed any further.**
3. Save your work often and maintain multiple backup copies. At the very least, make a backup copy for each new addition you make in your project.