300103 Data Structures and Algorithms

Practical 3

(due in Week 4)

(2 marks)

Task 3.1
Rewrite your code for **Task** 1.3, instead of storing the generated ten lists of the numbers in a STL "vector", store them in a STL linked list "list". Print the list of lists in the format shown in **Task** 2.4.

Hint: *If your solution to **Task** 1.3 stores the list of lists as a vector of vectors:*

```
vector<vector<int>>
```

do NOT simply change it to

```
list<vector<int>> or list<list<int>>
```

*Create a class, naming it **State**, to specify a list of nine numbers as specified in **Task** 1.2. Then 10 lists of the numbers will be 10 objects of this class.*

Task 3.2
Download the file *Code_for_Task3_2.zip*, which contains two files - *state.h* and *Task3_2.cpp*. The program fills a 3*3 matrix with randomly generated nine integers between 0 and 6 (possibly repeated) and print the matrix. In order to make the matrix looks more like a Mini-SHRDLU state, a function `pushDown()` is to be implemented in order to push down all the non-zero numbers towards the bottom as shown below:

```
//before push-down
2  2  0
4  3  1
2  0  0

//after push-down
2  0  0
4  2  0
2  3  1
```

Add your code to implement the function by transfering the non-zero numbers of each column into a STL stack and then storing these numbers back to the respective column of the matrix.

Hint: *Fill with zero if the stack size is less then BOARDSIZE*

Task 3.3
Download the file *Code_for_Task3_3.zip*, which contains *linkedStack.h* and *stackADT.h*. Read the code carefully to fully understand all the classes and functions

that have been implemented. Redo **Task** 3.2 by using `linkedStackType` instead of STL stack.

Hint: *Understand the provided code would take a while but write code for this task should take no more than five minutes once you have a solution to* ***Task*** *3.2; otherwise, you would be in a wrong direction.*

Task 3.4
Download the file *Code_for_Task3_4.zip*, which contains *linkedList.h* and *unorderedLinkedList.h*. Read the code carefully to fully understand all the classes and functions that have been implemented. Write a C++ application to demonstrate a use of *orderedLinkedList* by redoing **Task** 3.1 (instead of using STL list) and **Task** 2.3 (use the search function of `unorderedLinkedList,` instead of writing your own search)

Hint: *You may follow the OO style of the code for* ***Task*** *3.2 to rewrite your code for* ***Task*** *3.1 if your code is not in OO style. You might need to overload operator == and != to use linkedList.h* and *unorderedLinkedList.h. See the code provided for Task 3.2 for an example of overloading the operators* (`bool operator==(State s)` and `bool operator!=(State s)`).