COSC230 Assignment 2

This assignment consolidates material from *Practical 3* on "Linked Lists". Please submit all of your assignment files through `Turing` (see the link Assignment Submission via `Turing`). Before you submit your assignment please check your code compiles on `Turing` and does what you expect it to.

Question 1: [*25 marks*]

Download *Assignment 2 code files* from myLearn. Enhance the class `Linked_List` from *Practical 3* by implementing a member function `void reverse_list()` to reverse a singly linked list using only one pass through the list. Hint: One way to achieve this is to introduce, and increment, three temporary pointers. Draw a box and pointer diagram as shown in Practical 3 to help write your code, and then check your code using the code test provided. Please submit both your diagram (as a `pdf` file) and your source code file `sll.h` via `Turing`. Note: it is not possible to compile implementations separately from interfaces in the way we have been doing up until now when using template classes.

Question 2: [*25 marks*]

Use Θ notaton to write the worst case time bound for each of the following operations on data structures that you have implemented in `C++` code. Please submit a `pdf` file for this question.

(a) Insert a new element at the head of a linked list, and find a given element in a linked list. Explain how to find the memory address in each case.

(b) Insert a new element at the tail of a dynamic array, and find an element with a given index in an array. Explain how to find the memory address in each case. Why don't we insert an element at the head of a dynamic array?

(c) Which data structure would you choose for operations that frequently add and remove data from a dynamic set? Which data structure would you choose for operations that do frequent item lookups?

Question 3: [*20 marks*]

Download *Assignment 2 code files* from myLearn. Enhance the class `Array_Linked_List` from *Practical 3* by implementing the following member functions:

```
// Delete node with key k
void remove(Node_Array<T>&, T);

// Search list for key and return its index
int search(Node_Array<T>&, T);

// Print list starting from head
void print_list(Node_Array<T>&);
```

Please submit your source code file `dll_array.h` via `Turing`.

Question 4: [*30 marks*]

Download *Assignment 2 code files* from myLearn. There, you will find the files `database.cc` and `demo.cc`. The code in these files is very similar to that in the data-base exercise in *Practical 3*. Use the `makefile` and run `demo` to see this. Extend `database.cc` so that it stores a linked list of `Passenger` objects instead of `strings`. The data members of the `Passenger` class should include each passenger's first name, last name, and destination. You will also need to overload the `<` and `==` operators in order to make use of the STL functions. Do this so that for equality, both first names and last names are compared; and so that the list is sorted in lexicographical order on the last name. You may also find it convenient to overload the `<<` operator to generate nice output. Please include all of your code (including your `Passenger` class) in the `database.cc` file, and submit that for marking. Check your implementation using the unit test provided.