Lab 10: Need to submit Task 10.2, Task 10.3 and Task 10.4 as a part of assignment 3

Task 10.1 Write a statement or set of statements to accomplish each of the following. Assume that all the manipulations occur in main (therefore, no addresses of pointer variables are needed), and assume the following definitions:

```
struct bankEmployee {
      char name[20];
      int salary;
      struct bankEmployee *next;
};

typedef struct bankEmployee BANKEmployee;
typedef BANKEmployee *BANKEmployeePtr;
```

a. Create a pointer to the start of the list called `startPtr`. The list is empty.

b. Create a new node of type `BANKEmployee` that's pointed to by pointer `newPtr` of type `BANKEmployeePtr`. Assign the string `"Justin"` to member `name` and the value `1000` to member salary. Provide any necessary declarations and statements.

c. Assume that the list pointed to by `startPtr` currently consists of 2 nodes one containing `"Justin "` and one containing `"Sam"`. The nodes are in alphabetical order. Provide the statements necessary to insert in order nodes containing the following data for `name` and `salary`:

```
"Antony"        200
"Tony"          300
"Peter"         400
```

Use pointers `previousPtr, currentPtr` and `newPtr` to perform the insertions; State what `previousPtr` and `currentPtr` point to before each insertion. Assume that `newPtr` always points to the new node, and that the new node has already been assigned the data.

d. Write a while loop that prints the data in each node of the list. Use pointer `currentPtr` to move along the list.

e. Write a `while` loop that deletes all the nodes in the list and frees the memory associated with each node. Use pointer `currentPtr` and pointer `tempPtr` to walk along the list and free memory, respectively.

f. Incorporate all these into a complete C++ program, your linked list should contain all 5 nodes (Antony, Justin, Peter, Sam, Tony in alphabetical order).

Task 10.2 Create a linked list using the following structure (Need submit with assignment 3)

```
struct studentname {
      char letter;
      struct studentname *next;
};

typedef struct studentname STUDENTName;
typedef STUDENTName *STUDENTNamePtr;
```

Create a linked list manually (without using any loops or recursive functions or any functions) that contains five nodes where the data part (structure element *letter*) of the nodes should be the **first five letters of your last name**. One letter will go to one node and the node insertion should happen one after another in alphabetical order. When you insert a new node it should be in the right place to get alphabetical order as shown in Task 10.1 – refer to sample solution). If your last name doesn't have five letters fill the remaining nodes with letters from your first name starting from the first letter (Eg: if your name is Devin Ly, then nodes will contain L, y, D, e and v).

Example:

Assume your name is **Daenerys Targaryen**; take the first five letters of the last name, which is **Targa** so the insertion order is

```
newptr = new STUDENTName;
newptr -> letter = 'T';
      .
      .
      .
      a
      .
      .
      r
      .
      g
      .
      a
```

and when you print the linked list it should be like as shown below (Note, you are not using any sorting to get this in alphabetical order, you manually insert nodes in the right place to get it in alphabetical order).

Task 10.3 Write a complete C++ program that inserts 10 random integers from 0 to 50 ***in order*** in a **linked list**. Create a separate function to calculate the floating point average of the elements. You must use a **self-referential structure** to create the **linked list** for this program. (Need submit with assignment 3)

Task 10.4 Write a complete C++ program that uses a **stack** to determine whether a string is a palindrome (i.e., the string is spelled identically backward and forward). Assume your string doesn't have spaces and punctuation. You must use a **self-referential structure** to create the **stack** for this program. (Need submit with assignment 3)