**Class project**

**Objective:**

The purpose of this project is to expose you to:

Applying analysis, design and implementation of a linked list using dynamic data structure, discrete methods such as recursion are utilized in solution, objects classes are used in constructing a multi file project. RAM organization and the use of “**this”** pointer to accomplish tasks; in addition to using referenced parameters as constant, all methods are implemented in a procedure like programming to implement class objects and operations.

.

**Problem Specification:**

You are to submit the analysis, design and implementation of a bag that maintains an ordered linked list.

**Details:**

Each node’s data consists of: id number (5 digits), Name (a string), and hours & rate both are of the same type (both are: integers, floats or doubles). This data is provided in the attached file. Your program must read the data from either one of the files and place it in the linked list in descending order (that is highest to lowest) based on the id number.

Each Node of the list contains the data above, and a pointer to the next node. You are to define the Node class in a header file. Methods that will maintain the list are stored in a C++ implementation file.

The client C++ file will contain code that interacts with the user using a menu like interface to test operations on the sorted list. You may choose to define a child class that inherits the Node class or may use object composition. However, every class must have two files a header and an implementation.

Methods may include but are not limited to the following:

* + **Add** is a method that adds a Node at its appropriate position.
  + **Location** is a method that returns a pointer to where action may take place. (Recursive)
  + **Remove** is a method that removes a selected Node if it exists, returns true or false.
  + **Clear** is a method that clears the list of all the Nodes, but before a Node is deleted, it calls a **print** method that prints the Node data to a file. All nodes that are created using the new operator must be de-allocated, deleted.
  + **Peek is** an accessor method that displays, on the screen, the data in the Node given the students name, if the Node with name does not exist the method will display that. The list is unchanged.
  + **Gross** is a method that calculates the gross (gross is not a data member).
  + **Net** is a method that calculates the net after taking 20% taxes (net is not a data member).
  + Other functions may be used such as **getData(), setNext() …** as explained in the book and in previous projects.

**Requirements:**

* Specify each method’s purpose, describe its parameters, and write a pseudo-code version of its header as explained in book.
* A static data member that keeps track of the number of nodes in the list.
* Create a UML class diagram for all classes.
* Use class templates to accommodate employee’s payroll data that are integer, float and double.
* Create a header file and implementation file for the bag.
* Create a header file that contains the class declaration for the class Node. Each node contains data and a pointer to the next node. The last node’s next field should always be NULL.
* Create an implementation file that contains the methods for each class.
* Define a default constructor that initializes the head pointer to NULL.
* Define a destructor, may be virtual, if deleting nodes is done in other methods.
* Define a method to determine if the list **isEmpty.**
* Use guards against duplication of #include.

Create as many ”.h” header files as needed and write your own “.cpp” file that is a driver containing a menu like choices to drive the project. Loading the list is not a menu option, it’s done automatically.

**Grading Criteria:**

|  |  |
| --- | --- |
| 5 points | Good programming practices: Proper spacing, comments, use of variables, indentations and appearance of program. |
| 5 points | Default constructors are defined and performs their tasks. |
| 5 points | Destructors are defined. |
| 5 points | CRC card is submitted. |
| 5 points | Class diagrams are submitted and are correct. |
| 5 points | The preprocessor directive #ifndef is properly used. |
| 5 points | Each class’s header file contains the declarations. |
| 5 points | Implementation file for each header containing the functions implementations. |
| 5 points | Multi file project is used to accomplish task. |
| 10 points | Templates are used properly to allow employees data of different data types. |
| 5 points | Accessor methods are constants and parameters passed are also constants |
| 5 points | The **this** pointer is used to access member data. |
| 5 points | At least one recursive function is used. |
| 10 points | All methods above are clearly defined and self-contained. |
| 15 points | Program solution fits specifications. |
| 5 points | Contents of all files are submitted. |

**Submission Details:**

Hand-in:

* All source program files.
* test results using both data files.
* input/output files and all requirements above.

**\*\*\*\*\* Due Date 05/08/2017 \*\*\*\***

**Final May 8, 2017 5:15PM**