Assignment 2:

Part1:
contains 3 files, Node.h, singleLinkedList.h, main.cpp

Node.h contains the declaration of a node with data fields “data” and pointer next .

singleLinkedList.h: contains the following functions

PushFront: Adding a new item to the list. We can add an item to an empty list or to a non-empty list.
1) Empty list: we need to allocate memory for one new Node, then update the LinkedList’s first and last pointers. Since this new Node is the only item in the List, it is both the first and the last item.
2) Non-empty list: Adding a Node to the beginning of the list, replacing the current first item and updating the pointers.

PushBack: Similar to PushFront functionality, except we’re working at the opposite end of the list. Adding a new end item. We have two scenarios:
1) Adding to an empty list: same process with PushFront.
2) Adding to a non-empty list: there are one or more Nodes in the list. We will be adding a Node to the end of the list, replacing the current last item and updating the pointers.

Insert: Add a new item to the list at a specific index and update the pointers.
1) If the index is 0, call PushFront(newItem) function.
2) if index is itemCount-1, call PushBack(newItem) function.
3) if the index is invalid, throw an exception.
4) else: insert the new item at index and update pointers

PopFront: Removing the first item of the list and setting up a new first item
1)Remove the last item: There is only one item in the list, free the memory by “delete head”, then update pointers.
2) There are 2 or more items in the list : Remove the first item and set a new first in the list. Update pointers as shown in the code.

PopBack: Removing the last item of the list
1) Removing the last item: Same as PopFront scenario1.
2) There are 2 or more items in the list: Remove the last item and set a new last in the list. Then, update pointers.

Remove: Remove an item at a specific index and update pointers.
1) If the index is 0, call PopFront() function.
2) else if index is itemCount-1, call PopBack() function.
3) else if the index is invalid, throw an exception.
4) else update pointers as shown in the code.

Front() : Get the data stored within the first Node via head->data.
Back(): Get the data stored within the last Node via tail->data.

find(): Walk through the linked list to find an element at a specific position in the list.

Note: I worked for long hours for this part and I didn't figure out what is the issue with my program. The compiler does not show any error but the program does not work as intended.

Part2:
I created a separate project for this part because I had issues with the first part and I couldn't fix them.

This project contains 7 files: employees.h, main.cpp, nonprofessional.h, nonprofessional.cpp, professional.h, professional.cpp

The file employees.h contains the abstract class "Employees", which contains the data shared by all employees like: name and ID. It contains the following member functions calculWeeklySalarry, calculVacation(), calculHealthContribution(), read_employees_data(), to_string().

There are 2 derived classes inherited from the base class Employees: professional_employees derived class and nonprofessional_employees derived class. Each derived class has a different way to compute the weekly salary, the paid vacation and the health contribution.

professional_employees derived class contains additional data fields: hrWorked and annualSalary;
nonprofessional_employees derived class contains additional data fields:hourlyRate, hrWorked.