Report

1. Discuss about how you implemented each function

主要需要實作的包含：I2P2_iterator.cpp、I2P2_Vector.cpp、I2P2_List.cpp，而這也是我 implement 的順序，接下來一一說明。

a.) I2P2_iterator.cpp

我是依下述順序完成的

- **const_iterator**

  這邊經過助教的講解，再 trace code 之後，大概可以看出大多數都是將任務 delegate 到 iterator_impl_base，其中可能是 vector_iterator 或是 list_iterator。

  其中我多加了一個 function: Node* const_iterator::node_ref() const，目的是當 list_iterator 需要 node 位置時，可以透過這個 function 回傳而不需要寫 Homework 時的計算。

  剩下部份幾乎都是直接將 operator delegate 到下面，所以直接 call 下面的同一個 operator 就好。

- **iterator**

  這邊跟 const_iterator 幾乎相同，只是為了 stl 而分成這兩個 iterator，內容只是將用到 const_iterator 改成 iterator。

- **vector_iterator**

  這邊就要開始 implement 實際要如何實現那些 operator 以及 function。

  首先我添加了 pointer ptr_to_data 這個 member data，代表的是這個 iterator 現在指到的 data 的 pointer。

  並且加了以下三個 function:

  vector_iterator(pointer p);

  iterator_impl_base *clone() const;

  Node* node_ref() const;

  分別是對這個 ptr_to_date 做初始化的 Constructor；給 const_iterator 以及 iterator 使用的 clone，以及一樣是在 iterator_impl_base 添加的 virtual function node_ref，但這邊並不會使用。

  Vector 這邊比起 List 相對簡單，因為資料是連續的，所以大多都透過簡單的操作就可以實現 operator。主要就是透過 dynamic_cast 轉換 rhs，就可以得到他的 ptr_to_data 並與 this 的做比較。

- **list_iterator**

  這邊我額外添加的 member data 有：Node* _node;　Node* _head;
  前者代表目前只到的 node，而_head 則是為了後續比較用的，也實現在 I2P2_List.cpp。
  前半部分比較簡單只需要將對 pointer 的++--改成 next, prev 就好，但因為也要達到 random access 的標準，所以要多實現一些一般 list 不會提供的 function 例如>= 、<=、[]等等。
  所以我這邊的做法就是用到前面提到的_head，計算 this 跟_head 的距離以及 rhs 跟_head 的距離，比較之後就可以得到答案。
  只是這樣就多了很多 linear 的計算時間。

b.) I2P2_Vector.cpp

這邊多定義了三個 member data: pointer buffer;　size_type size_;
size_type capacity_;
第一個就是紀錄儲存空間，後面分別記錄 size 以及 capacity。
因為空間是連續的，所以比起 List 這部分也比較簡單，大多都類似作業所寫的。
而比較難的是 2 個 erase 跟 2 個 insert。
最後做法是透過 iterator.operator->()跟 buffer 的差距得到 index，在透過這個 index 去實作 operation。
另外因為 size_type 是 unsigned，所以會-1 underflow，導致當 vector 為 empty 時會導致直接跑 for loop 會跑很久，所以多寫了額外判斷的部分。

c.) I2P2_List.cpp

先將會用到的 struct Node 定義在 def.h
在這邊我在 List init 時就先創建一個 dummy node，begin 會是這個 node 的 next，而 end 則會是這個 node，因為我實作的是 circular linked list。
而也是在這部分花很久決定這樣做後，將 list_iterator 也加入這個 dummy node，就可以判斷 node 間差距等等。
且因為是 circular linked list 且又有 dummy node，在一般操作時就不會特地判斷 list 是否為空或是 insert 位置是否為 begin 等等。
而這邊的 iterator 就可以透過 node_ref 這個 function 來回傳真正代表的 Node 的 pointer。找到這個 tragetNode 後就可以做 erase 以及 insert 等。

2. Discuss time complexity

這邊我分成 Vector.cpp、List.cpp、iterator.cpp 三部分討論。

- ✓ **Vector.cpp**

  大部分的 operation 因為空間連續支持 random access 的關係，都是 O(1)，而其中 copy constructor、erase、insert、reserve、shrink_to_fit 因為資料要搬動的關係，所以都是 O(N)，這在 list 也是一樣。
  另外這邊的 pop_front、push_front 資料要搬動，所以也是 O(N)。

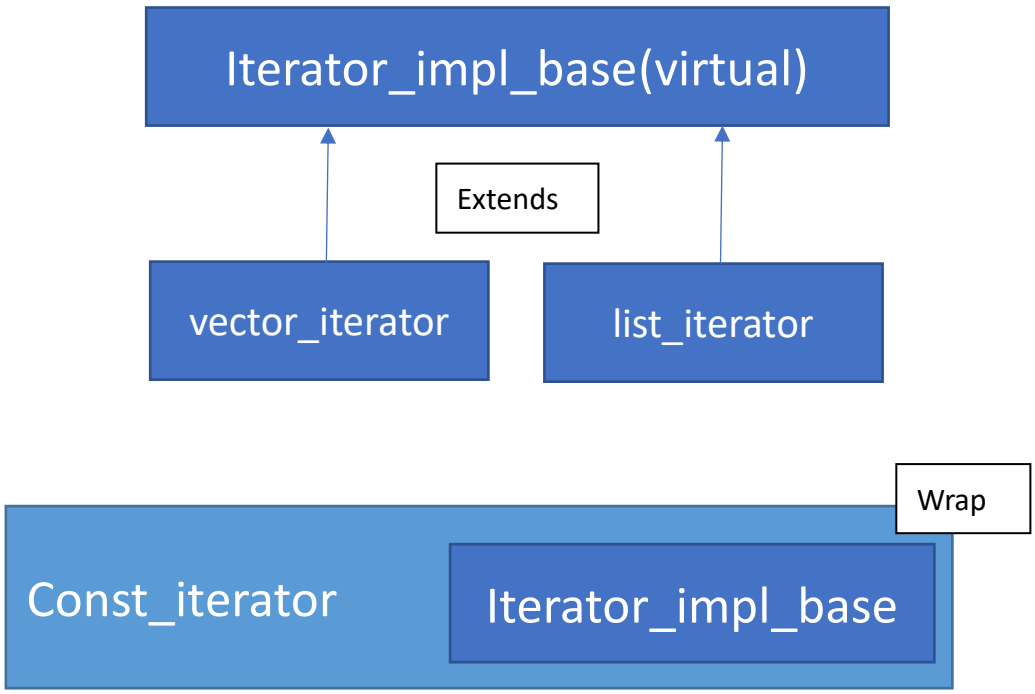
- ✓ **List.cpp**

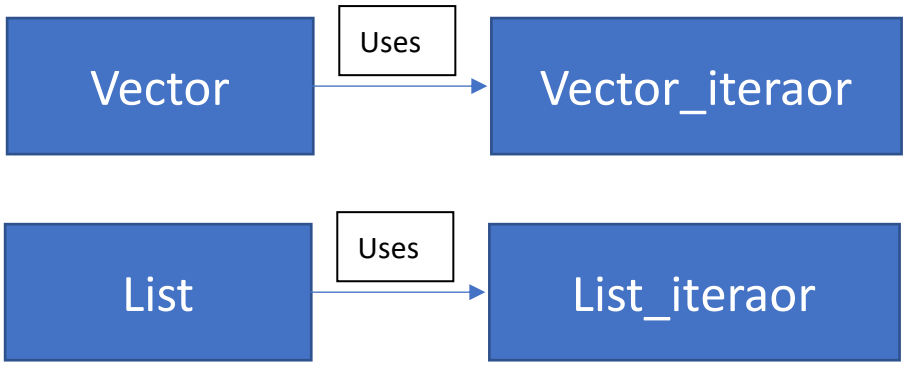
  除了上述提到是 O(N)的之外，因為 clear 要一個一個 Node 去 delete，所以這也是 O(N)，pop_front、push_front，因為不用搬動，所以為 O(1)。

- ✓ **Iterator.cpp**

  因為 vector_iterator's member data 為 pointer，所以可以直接比較或是加減，所以大多為 O(1)。
  但是 list_iterator 為 Node，移動時必須一個一個移，導致大多數 operation 包含+=、-=、[]、比較等等，都必須花費 O(N)來完成。

3. Discuss the hierarchy relationship





4. template part

因為這邊 spec 沒有要求很多，所以我直接建立兩個.h 檔並把內容也寫在一起。

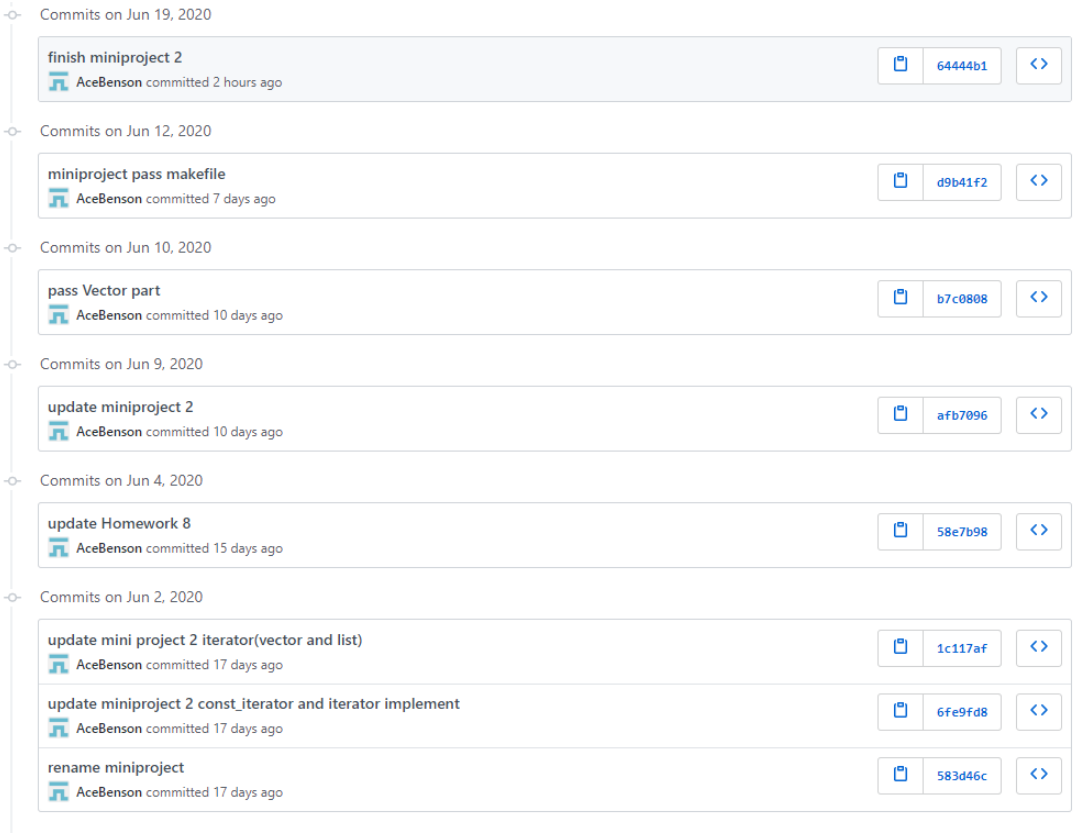
Vector 以及 List 的差異都在前面提過，而兩者改成 template 版本並沒有什麼太大不同，function 大致上與前面內容相近。

我先在裡面定義 base_Vectoriterator 以及 base_Listiterator，並透過 typedef 將其定義成分別的 iterator，這樣才能透過補上 operator--支援 --end() 的呼叫，其他部分都只是加上 template 的語法。

另外在 Vector 部分，因為之前 value_type 都定義好了，所以搬移時可以直接 call buffer[i] = buffer[i+1]這種語法，但這時候要用 replacement new，來在分配好的空間補上資料並 call T 的 constructor。

5. Appendix

Github 上的 history



I2P2_main.cpp 的 test 結果

```
(base) PS D:\Current Semester\ProgrammingII\MiniProject2\106060004> make
g++ I2P2_main.cpp src/*.cpp -DTEST_VECTOR -DTEST_LIST -DDOUBLE -std=c++11
(base) PS D:\Current Semester\ProgrammingII\MiniProject2\106060004> .\a.exe
Checking list ...
Checking vector ...
Finished
47.674S
(base) PS D:\Current Semester\ProgrammingII\MiniProject2\106060004>
```

sample_test.cpp 的 test 結果

```
(base) PS D:\Current Semester\ProgrammingII\MiniProject2\106060004> cd .\template\
(base) PS D:\Current Semester\ProgrammingII\MiniProject2\106060004\template> make
g++ sample_test.cpp -std=c++11
(base) PS D:\Current Semester\ProgrammingII\MiniProject2\106060004\template> .\a.exe
Checking Vector...
Checking List...
Finished
5.607S
(base) PS D:\Current Semester\ProgrammingII\MiniProject2\106060004\template>
```