Part A – Give a short answer to the following questions. (40marks)

1. What is the type of the `std::cerr` object? (0.5 marks)
2. Write a #include statement to include the `std::shared_ptr` class in your program. (0.5 marks)
3. What is the output of the following program? (2 marks)

```
#include <iostream>
#include <cstdlib>
#include <memory>

class base
{
public:
    base()
        {
            std::cerr << "base created" << std::endl;
        }

    virtual ~base()
        {
            std::cerr << "base destroyed" << std::endl;
        }
};

class derived : public base
{
public:
    derived()
        {
            std::cout << "derived created" << std::endl;
        }

    virtual ~derived()
        {
            std::cout << "derived destroyed" << std::endl;
        }
};

int main(void)
{
    std::unique_ptr<derived>d = std::make_unique<derived>();
    return EXIT_SUCCESS;
}
```

4. Given that the following is valid c++ code:

```
std::unique_ptr<int> myint = std::make_unique<int>(5);
```

Does that mean that `int is a class?` If it is not a class, what is it? Be as precise as you can (1 mark)

5. What is the meaning of the noexcept keyword and how would you use it ? What happens if you throw an exception in a function marked noexcept? (2 marks)
6. Why doesn't the STL list class support random access? (1 mark)
7. Show an example of using the `std::stol` function to extract an integer from a string. Include any exception handling (2 marks)
8. Name the three classes introduced in c++11 that implement the safe pointer idea. Briefly explain the purpose of each type and explain its use (6 marks)

9. Consider the following class declaration:

```
template<typename T>
class ptr_vector : private std::vector<std::unique_ptr<T>>
{
    std::vector<std::unique_ptr<T>> container;
    public:
    void add(std::unique_ptr<T>&& );
    virtual ~ptr_vector(void);
};
```

a) implement the function add() including any appropriate exception handling (4 marks).

b) implement the destructor for this class (2 marks).

You may assume that the add function and destructor and implemented inside the class declaration. (6 marks in total)

10. Consider the following code segment:

```
template<class T>
class tree{
    class node{
        std::unique_ptr<node<T>> left;
        std::unique_ptr<node<T>> right;
        T data;
    }; };
```

We want to make left and right accessible to the tree without accessors and mutators. Outline two ways we may achieve this (6 marks)

11. It is common in C++ to inline a function. What are the two ways we can specify that a function should be inlined? (2 marks)

12. In C, void pointers are used to implement generic programming. C++ provides similar functionality via templates. What benefits do they give us? Is there a runtime performance difference to using templates over void pointers? Which is faster and what cost do we pay for this speed? (5 marks)

13. Discuss the internal data representation that is typically used with a std::list, a std::vector and a std::set. Explain the relationship between the internal representation of each data structure and its expected performance, ranking each datastructure from slowest to fastest in terms of expected performance (6 marks)

14. Write a short program that reads in a line from a file called ints.txt that is delimited by commas and has integers stored between the commas. Tokenize this data and extract the numbers and store them and insert them into a vector. Tokenize this data and convert to integers to store in a vector. Now, save this data to a binary file. You should write out the number of integers in the vector followed by the data stored in the vector (20 marks).

Part B: Operator overloading (60 Marks)

We can model a game board as a vector of vector of cells (a 2d vector of board locations) where each cell may contain a red piece, a white piece or be empty.

1. Write the declaration (not the implementation) of the class board. Assume that this class is defined in a file called board.h. Include the prototypes for the following methods (6 marks):

    a) A default constructor with two optional integer arguments set to the default value of 8. This should create a board where the first argument is the width and the second argument is the height.

    b) A copy constructor.

    c) A move constructor for a board

    d) the assignment operator for this class.

    e) A conversion operator to the int type so that whenever an object of type board fraction is assigned to an integer, it returns the number of squares on the board. For example an 8x8 board would return 64.

    f) The prototype for a static member function for finding whether a board with the width and height specified would be square (same number of squares wide as high.

    g) The prototype for overloading operator<<(). Place this in the correct location and insert any other statements required to make this work. Please note that if other types need to be output you should write operator functions in preference to handling their output within this functions.

    h) A less than operator for comparing boards which returns true when the current board has less area than the one it is being compared to.

    i) The prototype for a destructor.

Note: Assume all functions implemented below are implemented in a file called board.cpp that includes the above mentioned file board.h

2. Implement the default constructor for the fraction class as specified in 1 a). (4 marks)
3. Implement the copy constructor specified in 1 b).(6 marks)
4. Implement the move constructor specified in 1 c) (4 marks)
5. Implement the assignment operator specified in 1 d) (8 marks)
6. Implement the conversion operator specified in 1 e).(5 marks)
7. Implement the static member function specified in 1 f).(4 marks)
8. Implement the operator<<() function specified in 1 h).(10 marks)
9. Implement the less than operator specified in 1h). (3 marks)
10. Implement the destructor specified in 1 i). (4 marks)
11. Write a lambda function that accepts two const cell references as arguments and returns true when the first cell is “less than” the second cell. The ordering of “less than” for a cell shall be(from lowest to highest): EMPTY, WHITE, RED. This lambda should not capture anything from its environment (6 Marks).