Paragraph 1 Based on the context and the files answer the questions (1-12)

table.hpp

```
#pragma once
#include <iostream>
#include <vector>
using std::ostream;

class Table {
   private:
     using row_t = std::vector<int>;
     std::vector<row_t> table_;
     std::vector<row_t> * p_acc_table_;
     size_t width_;
     size_t height_;
   public:
     //constructors
     Table(size_t, size_t, int val=0) ;
     Table(const Table& other);
     Table(Table&& other);

     //operator overloading
     Table& operator=(const Table& other);
     Table operator+(const Table& other1);

     //Getter and Setter
     size_t GetWidth() const {return width_;};
     size_t GetHeight() const {return height_};

     //move assignment
     Table& operator=(Table&& other) = delete;

     //Destructor
     ~Table(){p_acc_table = nullptr;};

     void PrintTable(ostream&)const;
     bool SetValue(size_t, size_t, int);
     int GetValue(size_t, size_t) const;
}
```

table.cpp

```
// assignment operator
Table& Table::operator=(const Table& other) {
     width_ = other.width_;
     height_ = other.height_;
    table_ = other.table_;
     p_acc_table_ = &this->table_;
   return *this; }

// Move constructor
Table::Table(Table&& other) {
     width_ = other.width_;
     height_ = other.height_;
     table_ = std::move(other.table_);
     p_acc_table_ = &this->table_;
}

// Copy constructor
Table::Table(Table& other) {
     width_ = other.width_;
     height_ = other.height_;
     table_ = std::move(other.table_);
     p_acc_table_ = &this->table_;
}

// Setter of table_
bool Table::SetValue(size_t col, size_t row, int
val)
   {
   (*p_acc_table_)[row][col] = val;
   return true;
}

// Getter of table_
int Table::GetValue(size_t col, size_t row) const{
   return (*p_acc_table_)[row][col];}

// Printing of table_
void Table::PrintTable(ostream &os)const{
   for( auto & row : table_){
     for( auto & x : row){
       os << x << ',' ;}
     os << std::endl; }
   }

// + operator overloading
Table Table::operator+(const Table& other1) {
   Table new_table = Table(other1.GetWidth(),
other1.GetWidth());
   for (size_t i = 0; i < other1.GetHeight();
i++){
   for (size_t j = 0; j < other1.GetWidth(); j++)
{
   new_table.SetValue(j, i, other1.GetValue(j,
i) + this->GetValue(j, i)); } }
   return new_table; }
```

main.cpp

```
#include <iostream>
#include ẗable.hpp¨
using std::ostream;

int main() {
    Table t1(5, 5);
    Table t2 = t1;
    Table t3 = t1 + t2 ;
    Table t4{t1};
    // changing t2 should not change t1
    t2.SetValue(3, 2, 232);
}
```

1. In the `PrintTable` function, what do you replace `auto` with in the line `for( auto & row : table_)`? For this line to compile and run successfully.

   (a) `row_t`
   (b) `const row_t`
   (c) `row_t *`
   (d) `*row_t`
   (e) `const row_t &`
   (f) `const row_t *`

   Correct answers: (b)

   Explanation: When the function of the class is constant, all the variables will be of const type. Therefore, in the for loop, the correct type of the element is `const row_t`.

2. What is the primary difference between the assignment operator (=) and the copy constructor?

   (a) `copy constructor can be chained`
   (b) `assignment operator returns pointer to class`
   (c) `copy constructor returns pointer to class`
   (d) `assignment operator can be chained`
   (e) `None of the following, They are same`
   (f) `2 of the options are correct`

   Correct answers: (f)

   Explanation: The return type of copy constructor is None but the return type of assignment is pointer to the class which allows us chain that implies Table t1 = t2 = t3 will work

3. What are the essential functions of the Rule of 5 needed for the (`main.cpp`) to run as expected?

   (a) `copy constructor`
   (b) `copy assignment`
   (c) `destructor`
   (d) `move constructor`
   (e) `move assignment`
   (f) `Rule of 5 is not applicable`
   (g) `both copy and move constructor`
   (h) `both copy and move assignment`
   (i) `copy and move assignment as well as constructors.`
   (j) `All of the Rule 5 are mandatory.`

   Correct answers: (j)

   Explanation: As there is a pointer `std::vector<row_t> * p_acc_table_;` it is mandatory to have all the constructors to have desired behaviour and also we need to destroy the pointer once class goes out of scope to avoid overflow and we also need to delete the default type to avoid the erratic behaviour.

4. Suppose the assignment operator `Table& Table::operator=(const Table& other) { ... }` is not defined in the `table.cpp`. Which of the following would happen?

   (a) `Will result in a compilation error, as the function is defined in the header file but doesn't have an implementation.`
   (b) `Will result in a runtime error.`
   (c) `Default assignment operator will be called.`
   (d) `t2.SetValue(3, 2, 232) will also make t1[3][2] = 232.`
   (e) `Pointer in t2 will still hold the reference of t1.`
   (f) `t2.SetValue(3, 2, 232) will not affect elements in t1.`
   (g) `2 of the following are true.`

   Correct answers: (g)

Explanation: If the assignment operation is not written the class will fall back to the default behaviour of copy all the objects in the class but will not result in any incorrect behavior, but if set and get value the of t1 will also be updated. In our case this behavior is not needed.

5. Which of the following will result in an error when the following line is added to the end of `main.cpp`?

   (a) `Table t5{t1};`
   (b) `Table t5 = t2 = t1;`
   (c) `None of the following`
   (d) `t2.SetValue(3, 2, 232);`
   (e) `All of the above`
   (f) `t2.SetValue(3, 2, t1.GetValue(2,3));`

   Correct answers: (c)

   Explanation: We have defined the behaviour for each of the following, first option is copy constructor, which is defined and in non-delete mode and second option is chaining, which is possible through move constructor and assignment operator and getter and setter are defined for the function so all the above will not give an error.

6. What does `Table t3 = t2 + t1` do or what functions are invoked when the following line is executed?

   (a) `Table::operator+`
   (b) `Move constructor`
   (c) `Destructor`
   (d) `Copy constructor`
   (e) `operator+`
   (f) `Default constructor`
   (g) `None of them are true`
   (h) `4 of them are true`
   (i) `3 of them are true`
   (j) `2 of them are true`

   Correct answers: (h)

   Explanation: `Table t3 = t2 + t1` we need to understand how this works in C++, first the operator+(t2,t1) is called. In the first line of the function we need are creating a new class instance new_table which will invoke default constructor and we are returning local variable which needs to be moved (move operation) out then finally the local variable is destroyed using destructor.

7. Regarding the implementation of the addition operator `Table Table::operator+(const Table& other1)`, which statement is correct?

   (a) `Compilation error occurs because we are returning a pointer to a local variable.`
   (b) `It runs perfectly; when returning local objects, the move function is invoked.`
   (c) `There is an incorrect declaration in the header file.`
   (d) `:operator+ should not be part of the class`
   (e) `:operator+ should be written in main`

   Correct answers: (b)

   Explanation: `Table Table::operator+(const Table& other1)` is a valid function and results in no error all of these are handled by using the above question.

8. What is the better practice for writing getter and setter functions?

   (a) `Getter should be a const function.`
   (b) `Getter should never be written in header (.hpp) files.`
   (c) `Setter should be a const function.`
   (d) `Setter should never be written in header (.hpp) files.`
   (e) `The current code is written according to the best practice.`

   Correct answers: (a)

   Explanation: The general notion of witting a getter function is to make it const so we dont change items accidentally.