CS 141 Homework 3 Report

Alexis Lauren Vu

TOTAL POINTS

70 / 70

QUESTION 1

Short Answers 30 pts

1.1 a) 5 / 5

✓ - **0 pts** Correct

- **2 pts** const makes it a compiler error for this class function to change a member variable of the class
- **1 pts** Did not associate correct page with question

1.2 b) 5 / 5

✓ - **0 pts** Correct

- **5 pts** Shallow Copy
- **0.5 pts** Unmarked or improperly marked page for question

1.3 c) 5 / 5

✓ - **0 pts** Correct

- **3 pts** Shallow Copy of Pointer Fields

1.4 d) 5 / 5

✓ - **0 pts** Correct

- **2 pts** Can not alter const items

1.5 e) 5 / 5

✓ - **0 pts** Correct

1.6 f) 5 / 5

✓ - **0 pts** Correct

QUESTION 2

Simple Scoping 40 pts

2.1 Static Scoping 20 / 20

✓ - **0 pts** Correct

- **10 pts** 10, 6, 12, 12
- **2 pts** Missed Print Statement
- **5 pts** 10, 6, 12, 12
- **20 pts** Missing Question

2.2 Dynamic Scoping 20 / 20

✓ - **0 pts** Correct

- **5 pts** 10,12,10,12
- **10 pts** 10,12,10,12
- **20 pts** Missing Answer

2.

a. **[5] What is the meaning of const after a member function prototype?**
The meaning of *const* after a member function prototype makes the pointer to this immutable, disallowing any member data to be changed.

b. **[5] What happens if you use the default copy constructor for Vector?**
If a default copy constructor is used when handling dynamically allocated memory, a shallow copy will be created meaning *this* will be pointing at the parameter Vector instead of storing correctly allocated data at its own address.

c. **[5] What happens if you use the default assignment operator for Vector?**
Similar to the default copy constructor, when handling dynamically allocated memory the default assignment operator will point this to the same address as the parameter Vector rather than storing a correctly allocated copy of the data.

d. **[5] Why pass Vector by reference but make it const as with operator *?**
Passing Vector by reference does not require a copy of the left operand vector to be made, however by adding *const* to the prototype when overloading *operator** the member data of this vector will be left unchanged.

e. **[5] Why are operators *, +, and << friends and not member functions?**
*, +, and << cannot be member functions because the left operand of these operators is already part of other types/classes. For * and +, the operand is of the type integer while for <<, the operand is of type ostream.

f. **[5] Why does operator [] return a T & as opposed to a T?**
Any value on the left-hand side of the assignment operator must be an l-value. Since the subscript operator can occur on the left-hand side of the assignment operator, it must be an l-value. By returning it by reference, it’s guaranteed to have the qualities of an l-value which is having a memory address.

1.1 a) 5 / 5

✓ **- 0 pts** Correct

- 2 pts const makes it a compiler error for this class function to change a member variable of the class

- 1 pts Did not associate correct page with question

2.

a. **[5] What is the meaning of const after a member function prototype?**
The meaning of *const* after a member function prototype makes the pointer to this immutable, disallowing any member data to be changed.

b. **[5] What happens if you use the default copy constructor for Vector?**
If a default copy constructor is used when handling dynamically allocated memory, a shallow copy will be created meaning *this* will be pointing at the parameter Vector instead of storing correctly allocated data at its own address.

c. **[5] What happens if you use the default assignment operator for Vector?**
Similar to the default copy constructor, when handling dynamically allocated memory the default assignment operator will point this to the same address as the parameter Vector rather than storing a correctly allocated copy of the data.

d. **[5] Why pass Vector by reference but make it const as with operator *?**
Passing Vector by reference does not require a copy of the left operand vector to be made, however by adding *const* to the prototype when overloading *operator** the member data of this vector will be left unchanged.

e. **[5] Why are operators *, +, and << friends and not member functions?**
*, +, and << cannot be member functions because the left operand of these operators is already part of other types/classes. For * and +, the operand is of the type integer while for <<, the operand is of type ostream.

f. **[5] Why does operator [] return a T & as opposed to a T?**
Any value on the left-hand side of the assignment operator must be an l-value. Since the subscript operator can occur on the left-hand side of the assignment operator, it must be an l-value. By returning it by reference, it’s guaranteed to have the qualities of an l-value which is having a memory address.

1.2 b) 5 / 5

✓ - **0 pts** Correct

- **5 pts** Shallow Copy

- **0.5 pts** Unmarked or improperly marked page for question

2.

a. **[5] What is the meaning of const after a member function prototype?**
The meaning of *const* after a member function prototype makes the pointer to this immutable, disallowing any member data to be changed.

b. **[5] What happens if you use the default copy constructor for Vector?**
If a default copy constructor is used when handling dynamically allocated memory, a shallow copy will be created meaning *this* will be pointing at the parameter Vector instead of storing correctly allocated data at its own address.

c. **[5] What happens if you use the default assignment operator for Vector?**
Similar to the default copy constructor, when handling dynamically allocated memory the default assignment operator will point this to the same address as the parameter Vector rather than storing a correctly allocated copy of the data.

d. **[5] Why pass Vector by reference but make it const as with operator *?**
Passing Vector by reference does not require a copy of the left operand vector to be made, however by adding *const* to the prototype when overloading *operator** the member data of this vector will be left unchanged.

e. **[5] Why are operators *, +, and << friends and not member functions?**
*, +, and << cannot be member functions because the left operand of these operators is already part of other types/classes. For * and +, the operand is of the type integer while for <<, the operand is of type ostream.

f. **[5] Why does operator [] return a T & as opposed to a T?**
Any value on the left-hand side of the assignment operator must be an l-value. Since the subscript operator can occur on the left-hand side of the assignment operator, it must be an l-value. By returning it by reference, it’s guaranteed to have the qualities of an l-value which is having a memory address.

1.3 c) **5 / 5**

- ✓ **- 0 pts** Correct
- **- 3 pts** Shallow Copy of Pointer Fields

2.

a. **[5] What is the meaning of const after a member function prototype?**
The meaning of *const* after a member function prototype makes the pointer to this immutable, disallowing any member data to be changed.

b. **[5] What happens if you use the default copy constructor for Vector?**
If a default copy constructor is used when handling dynamically allocated memory, a shallow copy will be created meaning *this* will be pointing at the parameter Vector instead of storing correctly allocated data at its own address.

c. **[5] What happens if you use the default assignment operator for Vector?**
Similar to the default copy constructor, when handling dynamically allocated memory the default assignment operator will point this to the same address as the parameter Vector rather than storing a correctly allocated copy of the data.

d. **[5] Why pass Vector by reference but make it const as with operator *?**
Passing Vector by reference does not require a copy of the left operand vector to be made, however by adding *const* to the prototype when overloading *operator** the member data of this vector will be left unchanged.

e. **[5] Why are operators *, +, and << friends and not member functions?**
*, +, and << cannot be member functions because the left operand of these operators is already part of other types/classes. For * and +, the operand is of the type integer while for <<, the operand is of type ostream.

f. **[5] Why does operator [] return a T & as opposed to a T?**
Any value on the left-hand side of the assignment operator must be an l-value. Since the subscript operator can occur on the left-hand side of the assignment operator, it must be an l-value. By returning it by reference, it’s guaranteed to have the qualities of an l-value which is having a memory address.

1.4 d) **5 / 5**

✓ **- 0 pts** Correct

- 2 pts Can not alter const items

Alexis Lauren Vu
UCINetID: alexilv1

2.

a. **[5] What is the meaning of const after a member function prototype?**
The meaning of *const* after a member function prototype makes the pointer to this immutable, disallowing any member data to be changed.

b. **[5] What happens if you use the default copy constructor for Vector?**
If a default copy constructor is used when handling dynamically allocated memory, a shallow copy will be created meaning *this* will be pointing at the parameter Vector instead of storing correctly allocated data at its own address.

c. **[5] What happens if you use the default assignment operator for Vector?**
Similar to the default copy constructor, when handling dynamically allocated memory the default assignment operator will point this to the same address as the parameter Vector rather than storing a correctly allocated copy of the data.

d. **[5] Why pass Vector by reference but make it const as with operator *?**
Passing Vector by reference does not require a copy of the left operand vector to be made, however by adding *const* to the prototype when overloading *operator** the member data of this vector will be left unchanged.

e. **[5] Why are operators *, +, and << friends and not member functions?**
*, +, and << cannot be member functions because the left operand of these operators is already part of other types/classes. For * and +, the operand is of the type integer while for <<, the operand is of type ostream.

f. **[5] Why does operator [] return a T & as opposed to a T?**
Any value on the left-hand side of the assignment operator must be an l-value. Since the subscript operator can occur on the left-hand side of the assignment operator, it must be an l-value. By returning it by reference, it's guaranteed to have the qualities of an l-value which is having a memory address.

1.5 e) **5 / 5**

✓ **- 0 pts** Correct

Alexis Lauren Vu
UCINetID: alexilv1

2.

a. **[5] What is the meaning of const after a member function prototype?**
The meaning of *const* after a member function prototype makes the pointer to this immutable, disallowing any member data to be changed.

b. **[5] What happens if you use the default copy constructor for Vector?**
If a default copy constructor is used when handling dynamically allocated memory, a shallow copy will be created meaning *this* will be pointing at the parameter Vector instead of storing correctly allocated data at its own address.

c. **[5] What happens if you use the default assignment operator for Vector?**
Similar to the default copy constructor, when handling dynamically allocated memory the default assignment operator will point this to the same address as the parameter Vector rather than storing a correctly allocated copy of the data.

d. **[5] Why pass Vector by reference but make it const as with operator *?**
Passing Vector by reference does not require a copy of the left operand vector to be made, however by adding *const* to the prototype when overloading *operator** the member data of this vector will be left unchanged.

e. **[5] Why are operators *, +, and << friends and not member functions?**
*, +, and << cannot be member functions because the left operand of these operators is already part of other types/classes. For * and +, the operand is of the type integer while for <<, the operand is of type ostream.

f. **[5] Why does operator [] return a T & as opposed to a T?**
Any value on the left-hand side of the assignment operator must be an l-value. Since the subscript operator can occur on the left-hand side of the assignment operator, it must be an l-value. By returning it by reference, it’s guaranteed to have the qualities of an l-value which is having a memory address.

1.6 f) **5 / 5**

✓ **- 0 pts** Correct

3. **[40] Show the output of the following program (written in a hypothetical Ada-like language) executed twice: 1) assuming static scoping, and 2) assuming dynamic scoping. Assume that appropriate 'put' subroutines are defined to print out their arguments in a nice format. NOTE: Be sure you can execute this type of problem with mental tracing and drawing pictures of memory because you will do it several more times on a quiz and on the final quiz.**

```
PROCEDURE Simple_Scoping IS
     m: integer;
     PROCEDURE P IS
     BEGIN
          m := 12;
     END P;
     PROCEDURE Q IS
          m : integer;
     BEGIN
          m := 6;
          P;
          put("In Q m = ", m);
     END Q;
BEGIN
     m := 10;
     put("In Simple_Scoping Initially   m = ", m);
     Q;
     put("In Simple_Scoping after Q   m = ", m);
     P;
     put("In Simple_Scoping after P   m = ", m);
END Simple_Scoping;
```

1. **Static Scoping**
   Output:

```
In Simple_Scoping Initially m = 10
In Q m = 6
In Simple_Scoping after Q m = 12
In Simple_Scoping after P m = 12
```

2. **Dynamic Scoping**
   Output:

```
In Simple_Scoping initially m = 10
In Q m = 12
In Simple_Scoping after Q m = 10
In Simple_Scoping after P m = 12
```

2.1 Static Scoping 20 / 20

✓ **- 0 pts** Correct

- 10 pts 10, 6, 12, 12

- 2 pts Missed Print Statement

- 5 pts 10, 6, 12, 12

- 20 pts Missing Question

3. **[40] Show the output of the following program (written in a hypothetical Ada-like language) executed twice: 1) assuming static scoping, and 2) assuming dynamic scoping. Assume that appropriate 'put' subroutines are defined to print out their arguments in a nice format. NOTE: Be sure you can execute this type of problem with mental tracing and drawing pictures of memory because you will do it several more times on a quiz and on the final quiz.**

```
PROCEDURE Simple_Scoping IS
     m: integer;
     PROCEDURE P IS
     BEGIN
          m := 12;
     END P;
     PROCEDURE Q IS
          m : integer;
     BEGIN
          m := 6;
          P;
          put("In Q m = ", m);
     END Q;
BEGIN
     m := 10;
     put("In Simple_Scoping Initially   m = ", m);
     Q;
     put("In Simple_Scoping after Q   m = ", m);
     P;
     put("In Simple_Scoping after P   m = ", m);
END Simple_Scoping;
```

1. **Static Scoping**
   Output:

   ```
   In Simple_Scoping Initially m = 10
   In Q m = 6
   In Simple_Scoping after Q m = 12
   In Simple_Scoping after P m = 12
   ```

2. **Dynamic Scoping**
   Output:

   ```
   In Simple_Scoping initially m = 10
   In Q m = 12
   In Simple_Scoping after Q m = 10
   In Simple_Scoping after P m = 12
   ```

2.2 Dynamic Scoping 20 / 20

✓ - **0 pts** Correct

- **5 pts** 10,12,10,12

- **10 pts** 10,12,10,12

- **20 pts** Missing Answer