

135 Exam

● Graded

Student

Total Points

75.45 / 100 pts

Question 1

Operators

7 / 8 pts

1.1 — * operator

1 / 2 pts

+ 2 pts Correct

✓ + 1 pt Mentions Pointer (not dereference operator)

+ 0.5 pts Attempt

+ 0 pts Blank

1.2 — & operator

2 / 2 pts

✓ + 2 pts Correct

+ 0.5 pts Attempt

+ 0 pts Blank

1.3 — :: operator

2 / 2 pts

✓ + 2 pts Correct

+ 0.5 pts Attempt

+ 0 pts Blank

1.4 — -> operator

2 / 2 pts

✓ + 2 pts Correct

+ 1 pt Explains this/dereference

+ 0.5 pts Attempt

+ 0 pts Blank

Question 2

Pointer Function

Resolved 7 / 10 pts

✓ + 2 pts Correctly accesses members of the array

✓ + 1.5 pts Loops through the dynamic array

✓ + 1.5 pts Uses if statement to compare the length

+ 3 pts Returns the appropriate value

✓ + 2 pts Returns nullptr if not found

- 0.25 pts Syntax Error

+ 0.5 pts Attempt

+ 0 pts Blank

🔄 Regrade Request

Submitted on: Aug 17

Good afternoon, I believe for this question that I have properly returned the appropriate value for this question. I'll be thankful if this question was reviewed, I apologize with the bother, and may you have a good day.

The problem with this code is that it only checks the first value of array since you used an if/else clause here. The correction needed here is that the return nullptr value cannot be in the else statement.

Reviewed on: Aug 17

Question 3

Dynamic Arrays Short Response

10.5 / 12 pts

3.1 Size

0.5 / 2 pts

+ 2 pts Correct

+ 1 pt Off by one

✓ + 0.5 pts Attempt

3.2 Capacity

2 / 2 pts

✓ + 2 pts Correct

+ 1 pt Off by one

+ 0.5 pts Attempt

+ 0 pts Blank

3.3 Output

2 / 2 pts

✓ + 2 pts Correct

+ 0.5 pts Attempt

3.4 The new keyword

2 / 2 pts

✓ + 2 pts Correct

+ 1 pt Somewhat Correct

+ 0.75 pts Mostly Incorrect

+ 0.5 pts Attempt

+ 0 pts Blank

3.5 — **Deallocate the dynamic array** 2 / 2 pts

Deletes the dynamic array with `delete[] arr;`

✓ + 1 pt Correct

+ 0.9 pts Minor Syntax Error

+ 0.75 pts Major Syntax Errors

+ 0.5 pts Too many syntax errors

Fixes the dangling pointer with `arr = nullptr;`

✓ + 1 pt Correct

+ 0.9 pts Minor Syntax Error

+ 0.75 pts Major Syntax Errors

+ 0.5 pts Too many syntax errors

+ 0.5 pts Attempt

+ 0 pts Blank

3.6 — **Dynamic Array storage** 2 / 2 pts

✓ + 2 pts Correct

+ 0.5 pts Attempt

+ 0 pts Blank

Question 4

Multiple Choice

8.25 / 10 pts

4.1 — Vectors vs Arrays

1.25 / 2 pts

+ 2 pts Correct

✓ + 1.25 pts Mostly Correct

+ 0.75 pts Mostly Incorrect

+ 0.5 pts Attempt (no other rubric items apply)

+ 0 pts Blank

4.2 — Size vs Capacity

2 / 2 pts

✓ + 2 pts Correct

+ 0.5 pts Incorrect

4.3 — Constructors vs Member-Functions

2 / 2 pts

✓ + 2 pts Correct

+ 1 pt Missing one/Has Extra

+ 0.5 pts Attempt (no other rubric items apply)

+ 0 pts Blank

4.4 — Inheritance

1 / 2 pts

+ 2 pts Correct

✓ + 1 pt Incorrect

4.5 — Multiple derived classes

2 / 2 pts

✓ + 2 pts Correct

+ 0.5 pts Incorrect

Question 5

Access specifiers

0.5 / 10 pts

Public

- + 3 pts Correct
 - + 1.5 pts Somewhat Correct
 - + 1 pt Mentioned but not explained
-

Private

- + 3 pts Correct
 - + 1.5 pts Somewhat correct
 - + 1 pt Mentioned but not explained
-

Protected

- + 3 pts Correct
 - + 1.5 pts Somewhat correct
 - + 1 pt Mentioned but not explained
-

- + 1 pt Explanations are detailed enough
- + 0.5 pts Some explanations are detailed

✓ + 0.5 pts Attempt

- + 0 pts Blank

Question 6

Classes short response

7.75 / 12 pts

6.1 Line 6 const

0.5 / 2 pts

+ 2 pts Correct

+ 1 pt Mentions another use for `const`

✓ + 0.5 pts Attempt

+ 0 pts Blank

6.2 Line 7 const

2 / 2 pts

✓ + 2 pts Correct

+ 1 pt Mentions another use for `const`

+ 0.5 pts Attempt

+ 0 pts Blank

6.3 Line 8 const

0.5 / 2 pts

+ 2 pts Correct

+ 1 pt Mentions another use for `const`

✓ + 0.5 pts Attempt

+ 0 pts Blank

6.4 Data-members

2 / 2 pts

✓ + 2 pts Correct

+ 0.5 pts Attempt

+ 0 pts Blank

6.5 Include Guards

1 / 2 pts

+ 2 pts Correct

✓ + 1 pt Attempt

+ 0 pts Blank

6.6 Parameterized constructor

1.75 / 2 pts

✓ + 1 pt Function Header

✓ + 1 pt Function Implementation

✓ - 0.25 pts Syntax Error

+ 0.5 pts Attempt (None of the other Rubric applies)

+ 0 pts Blank

Question 7

Recursion

10 / 10 pts

✓ + 3 pts Calls the function recursively with updated values

✓ + 2 pts Uses the a correct if statement

✓ + 2 pts Increments the value by the correct value

✓ + 1 pt Prints out values recursively

✓ + 1 pt Prints the correct starting value

✓ + 1 pt Prints the correct ending value through recursion

- 0.25 pts Syntax errors

+ 0.5 pts Attempt (No other Rubric Categories apply)

Question 8

2D Arrays and Vectors

10 / 10 pts

✓ + 3 pts Took the averages correctly

✓ + 3 pts Looped through the 2D Array

✓ + 2 pts Added the values to a vector

✓ + 2 pts Returned a vector

- 0.25 pts Syntax Error

+ 0 pts Blank

+ 0.5 pts Attempt

Question 9

Classes and Inheritance

14.45 / 18 pts

Private Data Members

✓ **+ 1 pt** Contains two correct data members

+ 0.5 pts Contains one correct data member

Default Constructor

✓ **+ 0.5 pts** Contains default constructor

✓ **+ 0.5 pts** It is implemented correctly

Parameterized Constructor

✓ **+ 0.5 pts** Contains parameterized constructor

+ 0.5 pts It is implemented correctly

Getters

✓ **+ 0.25 pts** Getter for band name exists

✓ **+ 0.25 pts** Getter for band name implemented

✓ **+ 0.25 pts** Getter for band member count exists

✓ **+ 0.25 pts** Getter for band member count implemented

Setters

✓ **+ 0.25 pts** Setter for band name exists

✓ **+ 0.25 pts** Setter for band name implemented

✓ **+ 0.25 pts** Setter for number of band members exists

✓ **+ 0.25 pts** Setter for number of band members implemented

Syntax and class definition

✓ **+ 1 pt** Mostly or entirely correct

+ 0.5 pts Some minor syntax errors

+ 0 pts Major syntax errors

- 0.5 pts Too many syntax errors

Default Object

✓ + 1 pt Correctly creates the default object

+ 0.5 pts Incorrect attempt to create the default object

Linkin Park

+ 1 pt Creates a Band object for Linkin Park

✓ + 0.5 pts Attempts to create a band object for Linkin Park

Print out Linkin park

+ 1 pt Correctly prints out information about Linkin Park

✓ + 0.5 pts Attempt to print Linkin Park

Another band dynamically allocated

+ 1 pt Correctly allocate a new band object with a name and number of members

✓ + 0.5 pts Attempt to dynamically allocate a band object

Clean up the dynamically allocated object

✓ + 1.5 pts Properly clean up the dynamically allocated object and deal with the dangling pointer

+ 1 pt Cleans up the dynamically allocated object but doesn't set the pointer to nullptr or vice versa

+ 0.5 pts Attempts to clean up

✓ + 0.5 pts No major syntax errors

- 0.5 pts Too many syntax errors

+ 0 pts Blank

🔄 Regrade Request

Submitted on: Aug 17

Good afternoon, I believe this question I have created an object for Linklin Park and have been able to print out the information about Linkin Park based on the question. In addition to this, I am confident that I correctly allocated a new band object. I would appreciate if you can look over this question for me, thank you for your time.

Correct, but note that the question states "write individual instructions". The way you created the Linkin Park Band was not a single instruction so I cannot award any higher points there. For printing and dynamic allocation, the same issue applies. The correct approach would be to use the parameterized constructor instead of the default one and combining the print into one line.

PopBand

✓ + 0.2 pts `PopBand` default constructor

+ 0.2 pts `PopBand` parameterized constructor

✓ + 0.2 pts Class definition shows `public` inheritance from `Band`

✓ + 0.2 pts Correct placement of attributes in `private` and `public`

✓ + 0.2 pts Doesn't repeat inherited attributes

✓ + 0.2 pts Contains data member `lead_vocalist`

✓ + 0.25 pts Contains the `lead_vocalist` setter

✓ + 0.25 pts Contains the `lead_vocalist` getter

RockBand

✓ + 0.2 pts `RockBand` default constructor

+ 0.2 pts `RockBand` parameterized constructor

✓ + 0.2 pts Class definition shows `public` inheritance from `Band`

✓ + 0.2 pts Correct placement of attributes in `public` and `private`

✓ + 0.2 pts Doesn't repeat the inherited attributes

✓ + 0.2 pts Contains the data member `lead_vocalist`

✓ + 0.2 pts Contains the data member `lead_guitarist`

✓ + 0.25 pts Contains the `lead_vocalist` getter

✓ + 0.25 pts Contains the `lead_guitarist` setter

✓ + 0.25 pts Contains the `lead_guitarist` getter

Implementations

+ 0.3 pts `PopBand` constructors are implemented correctly

+ 0.3 pts `RockBand` constructors are implemented correctly

✓ + 0.3 pts `PopBand` getters are implemented correctly

✓ + 0.3 pts `RockBand` getters are implemented correctly

✓ + 0.3 pts PopBand setters are implemented correctly

✓ + 0.3 pts RockBand setters are implemented correctly

+ 0.35 pts No Syntax Errors

✓ - 0.2 pts Too many syntax errors

+ 0.5 pts Attempt (nothing else applies or the applied rubric items don't go above 0.5)

+ 0 pts Blank

🔄 Regrade Request

Submitted on: Aug 17

Hello, I apologize with my many regrade requests, but I would appreciate a review on this question. I have points taken a way due to syntax errors, but from what I see I don't notice any. This maybe a misunderstanding due to the fact that I crossed out a lot of things in this question so it may be hard to read my code. Again, I apologize for bothering you and thank you for your time.

The issue here was that you don't have individual setters. Now I could have docked points(-0.6) for the setters not being implemented correctly, but instead I choose to give you those points, but give too many syntax errors(-0.2) because you understood what needed to be done, but the setters were implemented incorrectly.

Reviewed on: Aug 17

FINAL EXAM
CSCI 135: Software Analysis and Design I
Hunter College, City University of New York
Instructor: Sadab Hafiz

26 July 2023

Exam Rules

- Show all your work where applicable. Your grade will be based on the work shown.
- The exam is closed book and closed notes.
- When taking the exam, you may have with you pens, pencils, and erasers.
- You may not use a computer, calculator, tablet, phone, earbuds, or other electronic devices.
- **Do not open this exam until instructed to do so.**

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

I understand that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.	
Name: _____	
EmpID:	0 1 4
Email: _____	
Signature: _____	

1. Describe why the following operators are used:

*	The pointer operator is used to store the memory address of a variable/object. This allows the pointer to be a copy of the variable/object and can be used for useful operations instead of variables/objects that can be large with data.
&	The & operator gets the memory address of a variable which can allow users to directly manipulate the variable in situations like passing the memory address of a variable in a function. This is called passing by reference.
::	The :: operator allows methods to be defined its class by having the class name of the method and a operator in between the class name and the function header thus referring back to the class.
->	The arrow operator calls the class a method via a pointer. It is a shorthand of the having the pointer calling the method using the dot then having it surrounded by parentheses with the pointer operator outside each parentheses.

Note: multiplication and && operator doesn't count

2. Write a function that returns a pointer to the first occurrence of the string longer than string query in the dynamic string array. Return nullptr if none of the words are longer.

```
string* longer_than_query(string* array, string query, int size)
```

```
{ String* array = new string
  for (int i = 0; i < size; i++) {
    String* word = array[i];
    if (*word.length() > query.length()) {
      return (word);
    }
  }
  else {
    return return (nullptr);
  }
}
```


3. Short Response (based on the code below)

```
1  int sum(int* arr, int size){
2      int total = 0;
3      for(int i=0; i<size; i++){
4          total += arr[i];
5      }
6      return total;
7  }
8
9  int main(){
10     int* arr = new int[10];
11     arr[0] = 10; arr[1] = 20; arr[2] = 30;
12     arr[3] = 40; arr[4] = 50; arr[5] = 60;
13     cout << sum(arr+1, 3) << endl;
14 }
```

(a) What is the size of the dynamic array?

The size of the dynamic array is 10.

(b) What is the capacity of the dynamic array?

The capacity of the dynamic array is 10.

(c) What will the program print?

The program will print 90.

(d) What does the new keyword do?

~~The new keyword makes a dynamic array name arr with a capacity of 10 elements.~~ The new keyword creates a dynamic array/pointer of any type, thus allocating memory from the heap.

(e) Write the code to deallocate the above dynamic array and fix the dangling pointer.

delete [] arr;
arr = nullptr

(f) Which part of the memory is dynamic arrays allocated? Stack or Heap?

Dynamic arrays memory is allocated to the heap.

4. Multiple Choice

- (a) Why are vectors better than arrays? Select all that apply.
- ☒ i. The size is not required during the declaration
 - ii. Vectors can be passed to functions by value
 - ~~iii.~~ More space efficient
 - ☒ iv. Vectors can be returned directly from functions
- (b) The size of a dynamic array is always less than the capacity. True or False?
- ☒ i. True
 - ii. False
- (c) How are constructors different from member functions? Select all that apply.
- i. Unlike constructors, member functions cannot call other member functions
 - ☒ ii. Constructors cannot return anything
 - ☒ iii. Constructors are only called once during object initialization
 - iv. Constructors cannot call other member functions
- (d) Which one of the following is true about inheritance?
- ☒ i. Private data members are inherited and can be accessed directly by derived classes
 - ~~ii.~~ Private data members are not inherited by derived classes
 - iii. Private data members are inherited but cannot be accessed directly by derived classes
 - ~~iv.~~ Private data members are not inherited by derived classes but still can be accessed
- (e) Multiple classes can be inherited from one base class. True or False?
- ☒ i. True
 - ii. False

5. What are the three access specifiers (also called access modifiers) and what do they do?

The 3 access specifiers are: setters, getters, and mutators.
Setters initialize the data members of the class, getters return the value of a data member, and mutators act on data members, thus able to change their value.

6. Short Response (based on the code below)

```

1      #include <iostream>
2      class Date {
3      public:
4          Date();
5          Date(std::string weekday, int month, int year, int day);
6          std::string getWeekday() const;
7          std::string setWeekday(const std::string &weekday);
8          const Date* copyDate();
9      private:
10         std::string weekday;
11         int month;
12         int year;
13         int day;
14     };

```

- (a) What does the const keyword do in ^{getWeekday()}~~printDate()~~ function defined in line 6?
 The const keyword makes sure that the function will only follow the its definition.
- (b) What does the const keyword do in ^{setWeekday()}~~getWeekday()~~ function defined in line 7?
 The const keyword in setWeekday() function will make sure ~~whatever memory address of a variable that's passed into the function~~ the value of the memory address of a variable that's passed into the function. Will
- (c) What does the const keyword do in copyDate() function defined in line 8? Not change.
 The const keyword assures ~~that return value~~ that the function returns a pointer of type Date.
- (d) What are the data members of this class?
 The data members of this class are, ~~two~~ variables called month, year, and day of type int and a variable called weekday of type string.
- (e) Mark is complaining that this file leads to redefinition errors. How would you fix this?
 I would fix this by including the const keyword at the end of the function header, that's causing the problem.
- (f) Write the code to implement the parameterized constructor.

~~Date(std::string day, Date~~

```

Date::Date(std::string weekday, int month, int year, int day) {
    then weekday = weekday;
    then month = month;
    then year = year;
    then day = day;
}

```

7. Write a recursive function that prints the numbers in a given range from `start` to `stop` while skipping numbers based on the `step` parameter. You can assume that the `start` will always be less than the `stop`. Separate each number being printed with a space. For example, calling `printRange(1,7,2)` prints 1 3 5 7

```
void printRange(int start, int stop, int step);
```

```
void printRange(int start, int stop, int step){
    if (start >= stop) {
        return;
    }
    else {
        cout << start << " ";
        printRange(start + step, stop, step);
    }
}
```

8. Write a function that takes a 2D integer array, finds the average of each row, and returns a vector with the averages. For example, `get_averages({{2,3,4},{4,5,6},{6,7,8}}, 3)` would return the vector {3, 5, 7}. The second parameter represents the number of rows in the 2D array.

Explanation:

$$2+3+4 = 9/3 = 3$$

$$4+5+6 = 15/3 = 5$$

$$6+7+8 = 21/3 = 7$$

```
const int COLS = 3;
```

```
vector<int> get_averages(int nums[][COLS], int rows){
```

```
    vector<int> vec;
```

```
    for (int i = 0; i < rows; i++) {
        int total = 0;
        for (int k = 0; k < COLS; k++) {
```

```
            total = 0;
            total = total + nums[i][k];
        }
```

```
        int average = total / COLS;
        vec.push_back(average);
    }
```

```
    return vec;
}
```

```
}
```

9. (a) Create a class Band. A Band object has the following **private** data members:

- Name of the band
- Number of members

Write the class interface, the constructors, accessor(getters), and mutators(setters) member functions of this class for all data members. All data members must be initialized at the moment of object creation. Assume that everything goes in a single file and provide the implementation of each function.

Class Band {

Private:

~~std::string name;~~ std::string name;

~~int num-of-members;~~ int num-of-members;

Public:

Band();

~~Set-Name(std::string band);~~ void Set-Name(std::string b);

~~Set-num-mem(int num);~~ void set-mem(int num);

~~Get-name~~ std::string Get-name() const;

int get-num-men() const;

};

void Band::Set-Name(std::string b) {

name=b;

}

void Band::Set-mem(int num) {

num-of-members = num;

}

std::string Band::Get-name() {

return(name);

}

int Band::get-num-mem() {

return(num-of-members);

}

Band::Band() {

name = "";

num-of-members = 0;

}

- (b) In the main function, write individual instructions to accomplish the following:
- Create a default Band object
 - Create a Band object with the name "Linkin Park" and 5 members
 - Print out the name and the number of members of Linkin Park
 - Create another Band object with your favorite band but this time allocate it dynamically
 - You don't need this object anymore so do everything necessary to clean it up

4

```
int main() {
```

```
    Band group;
```

```
    group.Set_Name("Linkin Park");
```

```
    group.Set_mem(5);
```

```
    cout << group.Get_name() << endl;
```

```
    cout << group.get_num_mem() << endl;
```

```
    Band *foo = new Band;
```

```
    foo->Set_Name("Kiss");
```

```
    foo->Set_mem(6);
```

```
    delete foo;
```

```
    foo = nullptr;
```

```
    return(0);
```

```
}
```

- (c) Derive classes RockBand and PopBand from Band using the most efficient simple inheritance hierarchy. Both classes will have a `lead_vocalist` but only the RockBand will have a `lead_guitarist`. It must be possible to change the `lead_guitarist` of the RockBand and `lead_vocalist` of PopBand later. All data members must be initialized at the moment of object creation and you must implement all the functions. Assume that everything is on the same file.

```

class RockBand: public Band
{
std::String lead_vocalist;
std::String lead_guitarist;
private:
    std::String lead_vocalist;
    std::String lead_guitarist;

public:
    RockBand();

void
    void set_lead_guitarist_vocal(std::String guitar, std::String vocal);
    std::String lead_g(g) const;
    std::String lead_v() const;
};

RockBand::RockBand() {
    lead_vocalist = "";
    lead_guitarist = "";
}

void RockBand::
    void set_lead_guitarist_vocal(std::String guitar, std::String vocal) {
        lead_guitarist = guitar;
        lead_vocalist = vocal;
    }

std::String
    std::String RockBand::lead_g() {
        return(lead_guitarist);
    }

std::String RockBand::lead_v() {

```

Continued in the back
of this page