**Confidential CS162 Final Exam – Spring 2024**

**Computer Science II**

**Final Exam**

**Name:**

* **The highest possible awarded grade is 110% on the final exam.**
* **After reaching 100 points, additional raw points will be scaled by 0.25 so that a score of 140 points will result in a grade of 110%.**
* **There are 140 points possible and only 100 are needed to be awarded a grade of 100%. (If you earn, for example, 120 points, you’ll be awarded a grade of 105%.) So relax and enjoy!**

**This is an OPEN-BOOK, OPEN-NOTES exam.**

**Do not cheat! Do not use google, websites, AI, or other people.**

**Feel free to note the textbook page # for each answer you provide. These will be aggregated to provide up to 10 “make-up points” if you finish with a grade less than 100%.**

**You may type out or hand-write your answers.**

**Upload a docx or a pdf scan of your completed exam by 11:59 pm Tuesday, June 12.**

**Remember, I am available on Zoom Monday 2-4 pm and 5:45-6:45 pm.**

**It is estimated that the exam will take between 1-1/2 hours and 3 hours. There is no time limit for the exam, except that it is due Tuesday night.**

**Section I. Getting Started with C++ (**17 points**). Answer all questions.**

**Fill in the blank –** *here are your choices for the following question (not all of these answers will be used; some may be used twice)*

= == != <= () <> ; > !>

{} # ## ! + += -= ++

1. Every complete statement ends in a ;

1. Preprocessor directives such as **include** are preceded by a #

1. The =, ==, !=, +, += operator always requires a variable on the left-hand side.

1. For an **if** statement, the \_\_\_\_\_<=\_\_\_\_\_ comparative operator returns *false* if the left operand is greater than the right operand.

1. We use the \_\_\_\_\_{}\_\_\_\_\_\_ symbols to contain the body of the if statement when there are multiple statements.

1. The\_\_\_\_\_\_++\_\_\_\_\_\_ operator increments a variable by 1.

1. The\_\_\_\_\_\_\_+=\_\_\_\_\_ operator assigns to the left operand the result of adding the left and right operands together.

1. The \_\_\_\_\_!> \_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_-=\_\_\_\_\_ symbols are two of the above symbols that are not legal C++ operators.

1. The \_\_\_\_\_\_\_\_\_\_\_ operator compares two values for equality.

**Do some calculations and place the answer in the blank.**

Assume your variables ALWAYS *start* with these values:

*num1 is 10 num2 is 15 num3 is 2 num4 is 5*

Fill in the blank with the value of the variable(s) *after* the given statement below.

1. num0 = num1 + 1; num0 is \_\_\_\_\_11\_\_\_\_\_\_ num1 is \_\_10\_\_\_\_\_\_\_\_\_
2. ++num1; num1 is \_11\_\_\_\_\_\_\_\_\_\_
3. num2++; num2 is \_\_16\_\_\_\_\_\_\_\_\_
4. num3 +=num4; num3 is \_\_\_\_8\_\_\_\_\_\_\_ num4 is \_\_\_5\_\_\_\_\_\_\_\_
5. num5 = num4++; num4 is \_\_\_\_\_\_\_\_6\_\_\_ num5 is \_\_5\_\_\_\_\_\_\_\_\_
6. num6 = ++num4; num4 is \_\_\_\_6\_\_\_\_\_\_\_ num6 is \_\_6\_\_\_\_\_\_\_\_\_

**Fill in the blank –** *Here are your choices for the following questions (not all of these answers will be used)*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *while* | *block* | *iteration const* | *do* |  | *until* |
| *infinite* | *undefined* | *&& !* | *||* |  | *++* |

1. *const* is a value in memory that cannot be changed once it is stored

1. *block* is a sequence of statements enclosed in curly brackets

1. *while* begins a loop that may never actually execute

1. *!* represents the logical NOT operator

1. The *&&* operation evaluates to true when both operands are true, but is false otherwise

1. Write the truth table for the || operator: Text (Gaddis) Pg 187

True || False – True

False || True – True

False || False – False

True || True – True

**Fill in the blank** *from your own knowledge*

1. A *local* variable is one defined inside a function’s body.

1. A *prototype* tells the compiler that a function exists but may not have been defined yet.

1. If a function doesn’t return a value, *void* appears as the return type in the function declaration.

1. Passing arguments by *value* means that a copy of the argument is made when calling the function.
2. Passing arguments by *reference* means that the function is sharing the calling routine’s memory for the argument

1. Before you can call a function it must be either be *declared* or *defined*.

**Section II. Arrays in C++ (**20 points**). Answer all questions.**

1. **Create a character array** appropriately large enough to hold the phrase “Great Job!”That is, write the statement that will allocate the memory, but do not assign the C-string, just declare the variable for it.

*Char phrase[11];*

1. Now, show how to read this C-string from the console and store it into the array:

*Cout << “Enter Good Job!: ”;*

*cin.getline(phrase, 11);*

*cout << phrase << endl;*

1. Write a loop to count the number of lowercase letters in the C-string. **Use a for -loop.**

*Int count = 0;*

*For(int idx = 0; idx < 11; ++idx)*

*If (islower(phrase[idx])){*

*count++;*

*}*

1. Now, what if the above is to be a function? Write the prototype statement for it.

*Void phraseIO(char arr[]);*

1. Rewrite the above code counter using a **while** loop instead of a for-loop.

I*nt count = 0;*

*While(arr[0] != ‘0’){*

*If (islower(arr[0]))*

*Count++;*

*Arr++*

*}*

**Section III. Functions and Arguments in C++ (**18 points**). Answer all questions.**

1. Give an example of a **function prototype** named “display” that (a) requires clients to pass adouble-precision floating point number **by value** and(b) **returns nothing:**

*void display(double num);*

Show how you can call this function: *(include all variable definitions)*

*double tacos = 3.14159;*

*display(tacos);*

1. Give an example of a **function prototype** named “input” that requires passing a double-precisionfloating point argument **by reference** and **returns nothing:**

*void input(double& num);*

Write an appropriate function call for this function: *(include all variable definitions)*

*double tacos = 3.14159;*

*input(tacos);*

1. Give an example of a **function prototype** named “input” that has no argumentsand **returns a true or false value:**

*Bool input();*

Write an appropriate function call for this function: *(include all variable definitions)*

*Bool userIn = input();*

**Section IV. Classes in C++ (**15 points**). Answer all questions.**

1. Classes differ from structs in that the default member access level for structs is *public* but for classes it is *private*.
2. What are the three member access specifiers?

*Protected, Private, Public*

1. Protected members are available only to (1)*Base class* (2) *derived class*, and (3) friends. Text (Gaddis) Pg 916
2. Accessor member functions include what key word in their declarations that distinguishes accessor member functions from mutator member functions?

*const*

1. True or False? Inline member functions are written inside the class declaration.

*True*

1. What is the function prototype for the **default constructor** for the class Point?

*Point();*

1. What is the full and complete function prototype for the **copy constructor** for the class Point?

*Point (const Point& p);*

1. What is the function prototype for the **destructor** for class Point?

*~Point(); Text (Gaddis) Pg 767*

1. In the declaration of the following constructor, what is the phrase or term that is used to refer to the information after the colon (before the body)?

**Person(string a\_name, int a\_age): name {a\_name}, age {a\_age} {}**

*The “initializer list”*

1. When deciding whether to make an operator overloading function a member function or a friend function, what is a relevant factor (that may be the determining factor) with regard to the relative position of the operands in the expression containing the overloaded operator?

*The factor to consider is if the left side operand is an object of the class. If it is, keep the function as a member, otherwise make the function a friend.*

1. Is it possible to overload the < operator to compare two Point objects and return true or false based strictly on comparing the objects’ **x** values without regard to their **y** values? Yes or No

*Yes.*

1. When using public inheritance for defining a derived class, what syntax is used in the declaration of the derived class?

*Class derClass : pubClass*

*{*

};

1. When using public inheritance, which data members of the base class can be accessed from within the derived class and which cannot?

*Protected and Public can be accessed in the derived class.*

*Private cannot be accessed in the derived class.*

**Section V. Functions and Arguments in C++**

1. **Structs -** Short Answer **(**10 points**)**

1. The fields (variables) in a structure are called *Data Members*
2. Structs should always be passed to functions by *reference Text (Gaddis) Pg 633 says “preferable”*
3. An array of 10 structs has how many elements 10
   * The first index of such an array is index 0
   * The last index of such an array is index 9
   * We need a *counter* to keep track of how many elements are currently being used (have data in them)
4. The beauty of a struct is that it can hold *multiple* types of data
5. Create a structure to hold an Animal (with a 20 character name, 40 character breed, and an integer age – the user will type up to 20 alpha characters and spaces to specify the name and will type up to 40 alpha characters and spaces for the breed):

*Struct Animal{*

*Char name[20] = “”;*

*Char breed[40] = “”;*

*Int age = 0;*

*};*

# Pointers and Structs - Short Answer (10 points)

1. Create a pointer to an Animal:

Animal\* animalPtr = new Animal;

1. Now show how to dynamically allocate an array of 10 Animals using your pointer: new Animal[10];
2. Show **with pointer arithmetic** (instead of array indexing notation, [ ]) how to access the third animal in that array (to output the name)

*cout << animalPtr+2 ->name;*

1. When we use the -> operator, what type of data goes in front? (e.g., xx->name)

*Pointer variable*

1. When we use the . operator, what type of data goes in front? yy. name

*member data*

# Dynamic Memory - Short Answer (15 points)

1. Show an example of code that would cause a memory leak:

*Int \*somePtr = new int[100];*

*//function ends without delet [] somePtr;*

*Text (Gaddis) Pg 531*

1. When does memory allocated with **new** get deallocated? By what two possible commands?

*When the delete operator is used*

1. A C++ class *member* is implicitly called when the lifetime of a class object is over (such as at the ending curly-brace block for a local object).

1. Show how to allocate an array of characters DYNAMICALLY for the C-string “Computer Science”:

*Const int SIZE = 16;*

*char\* arr = new char[SIZE];*

1. Show a useful example of using the dereference operator with this array:

s

*Int nums(char\* arr)*

*int num = 0;*

*for (int idx = 0; idx < SIZE; idx++)*

*{*

*num += \*arr;*

*arr++;*

*}*

*return num;*

*}*

1. Show how to deallocate the array:\_ *Delete [] arr;*

1. With **pointer arithmetic**, show how to output the last ‘e’ in the name:

*cout << char+15 ->name;*

# Fundamentals of Linear Linked Lists Part I - Short Answer (10 points).

Tip #1: A linear linked list (LLL) has a head pointer, but no tail pointer. Each node has a next pointer, but no prev pointer. The list ends with the last node’s next pointer equal to nullptr.

Tip #2: To traverse the list, you establish a current pointer (declared thusly: Node\* curr = head;). By starting at the first node, you can then use a while loop to traverse the entire list all the way to the last node.

Tip #3: To insert nodes, you traverse to the target spot and appropriately set (1) the next pointer of the added; (2) the next pointer of the node before the new node, if any; otherwise the head pointer.

Tip #4: To delete nodes, you traverse the list until curr is pointing to the node *before* the one that you want to delete. Then you nullify the target node’s next pointer (curr->next->next = nullptr), zero its data for secure programming (curr->next->data = 0;), delete the target node using the delete operator (e.g., delete curr->next), then set the current node’s next pointer to nullptr (curr->next = nullptr;), because the current node just became the last node in the revised list, didn’t it?

Tip #5: For simplicity on this exam, you can assume (or make) everything related to the LLL and its nodes public.

1. A linear linked list has a *head* pointer, which points to the first node.
2. A node has a *next* pointer, which points to another node in sequence.
3. The last node is special because the null pointer must be \_\_\_\_\_\_\_\_\_\_\_.
4. When we traverse, if we want to visit every node in the list, then we should traverse until current is *nullptr*.
5. When we traverse to add at the end, we should traverse until current->next is(*tail)*
6. When we traverse to remove the last node, we need pointers to which nodes in the list? *Head and Tail*
7. **Fundamentals of Linear Linked Lists Part II -** Short Answer **(**10 points**)**

# Assume that you have this exact LLL to start with…

35

10

45

55

h

ead

|  |
| --- |
| current |

Explanatory Notes:

1. The list contains a head pointer to the first node.
2. Each node contains an int data member and a next pointer.
3. The slash represents that a node’s next pointer equals nullptr.
4. An arrow represents that a node’s next pointer contains the address of the next node.
5. First, show how to get current to point to the first node. (Write the code below.)

*Node\* curr = head;*

1. Show how to delete that first node:

*Node\* val = head;*

*Head = head->next;*

*Delete val;*

1. Now show the code to get current to point to the last node with 55 in it:

*curr = head;*

*while (curr = curr->next;*

1. Show the code to then add a new node with 75 to the end:

*Node \*node\_pointr;*

*node\_pointr = new Node(75);*

# C++ Linear Linked Lists *– Traversal and Removal* (15 points)

Assume that you have a linear linked list of **dynamically allocated arrays of characters** (movie titles) and there is a head pointer…and that there may be items in the list already, *but we aren’t 100% sure!*

1. Write the code to swap the order of the last two nodes in a linear linked list

Node\* lastTwo = tail->getPrev();

Node\*lastThree = lasttwo->getPrev();

1. Write the code to remove the last node in the LLL

Text(Gaddis) p 1140

*Void numList::deleteNode(int num){*

*nodePointr = head;*

*while(nodePointr != null && nodePointr->value != 55){*

*previous = nodePointr;*

*nodePointr ->next;*

*}*

*}*

*DLL.deleteNode()*

1. Lastly, duplicate the last node (e.g., add a node at the end which is the same as the last node’s data)