**Honor Pledge: By submitting this exam I affirm that I have neither given nor received any unauthorized help on this assignment, and that all work is my own. This exam is open book, open note, open computer (for accessing notes, research, but not for communicating).**

For multiple-choice questions, your answer(s) must use my lettering/numbering. Only the letter/number is needed, not the actual words.

Your uploaded file’s name must be your name (i.e. BobMann.docx)



(log\_2(296000)) == 18 times

1. 1)Time, 2) Cost, and 3) Quality(Content)

template<class objectType>

int SomeClass<objectType>::MethodName() const {  
 return item;  
 }



Best comparison to use is average case; this is because the best case comparison is usually an insignificant comparison in terms of number of executions. The worst case should try to be avoided; the most key comparison to do between to functions/execution statements is the average time, since this will be likely in the normal execution range of the program 🡪 so it is key to make the average case be the most efficient case.

1. 3 permission levels are: r, w, x (read, write, execute)  
   3 classes of users are: owner, group, other
2. 1) Define problem, 2) business requirements, 3) analysis and design, 4) build and test 5) implementation, 6) maintenance
3. 3 types of errors: compile time (syntax), logic error (runtime), unexpected error (runtime)
4. \*shallow copy == bitwise copy; data member to data member (including memory addresses of pointers, which is why you cannot use this when an object contains pointers).  
   \*deep copy == has same functionality as shallow copy, except that for memory addresses (pointed to by pointers) are not copied but rather only the value pointed to by that pointer.
5. ADT is abstract data type, and classes and structures are abstract data types which can makeup even more ADTs such as stacks, heaps, queues, trees, etc.   
   An ADT is a collection of data and a set of operations on that data that it contains/manages which is why classes and structs are also ADTs (I am pretty sure classes and structs are ADTs)
6. The primary function of a friend class is so that one class is able to access all data members of another class – including the private data members. It is similar to inheritance except inheritance will never allow private member access, and friend classes have an exclusive relation ship with that is one-way and that cannot be passed down to other classes. For example, class A can be a friend of class B, but B is not a friend of A, and class A cannot pass its friendship down to any other classes (hence, this is why inheritance still place a key role since friendships lack this ability).  
     
   To make a friend, use the key ‘friend’ as a data member of the class like this:   
   friend class otherClass; ( I think it has to be under private data members, which classes default to private)
7. I think it means that your return is unchangeable
8. a binary search algorithm keeps cutting a (sorted; list must be sorted) list in half and comparing the value to be found with the median of each half of the elements. It starts after the first cut   
   with checking the initial median; if that element has a value that is too large it will cut the lower median into another half and repeat the comparison process; if the element that has a value that is too small, then it will cut the upper half again and repeat the comparison process. This is a more efficient approach than a linear search of a sorted list
9. A hashing function is a function that converts larger numbers into smaller numbers that can be organized into what we call a hash table ( a table storing the location of elements (hash values) that correspond to a value in the array) . The purpose of using a hash algorithm is to reduce the time it would take to search an otherwise very large unsorted list with very large values. The primary concern with hashing is value storage collision; this occurs when a value’s hash value is the same as another, so that when it is stored it is stored at an element already occupied by another value. But this can be mitigated through linear, quadratic, or other re-hashing methods, but those will change the time complexity depending on the level of re-hashing required
10. g++ file1.cpp file2.cpp -o nameOfObjectFile created after compilation
11. f.) 0 is false and anything else is true
12. A) 1, 2, 6  
    B) 1, 2, 3, 6
13. A
14. D

Always use Rule of 3! Constructor, copy constructor, destructor (and overload assg operator)

1. struct Node {  
    int info;  
    Node \* next;  
    };  
     
   void InsertTail( int info, Node \* head ){  
     
    while(head->next != nullptr){  
      
   cout << “I forgot how to do this darn it but it is in my head!!! Shoot!\n”  
    }  
     
     
     
   }
2. B, C, E, H
3. C
4. A
5. Initial case, recursive case, terminating/base case
6. Private, Protected, and Public
7. Add, remove, display, isFull, isEmpty
8. selection sort creates two groups of the elements to be sorted: sorted group, and unsorted group. Each time it checks the unsorted group, it swaps the smallest value for whatever is at top of list, then that value now moves to sorted (or upper depending on sorting for larger or smaller value and if it goes at the start or end of list) . Now the process continues swapping out the smallest unsorted value and placing it in the correct place, which also counts that value as now being sorted.  
     
   Bubble sort simply sends a value down or up the list to until it finds a value larger than or smaller than it self then it keeps moving along the list, using swap with the next element to move. This repeats all the way through starting at top of list (or bottom) until all elements have been compared.  
     
   insertion sort makes two lists: sorted and unsorted, making the first element in the list the first element that goes into the sorted list. Then each value in the unsorted list is compared unilaterally either from top or from bottom to each value in the sorted list (greater than or less than) to determine where it will be inserted  
     
   Selection sort has average time complexity O(N^3), Bubble has O(N^4), insertion has O(N^4)  
   With the easiest to implement is bubble sort

For each statement, show what is printed, the final value of ptr, and the final values in array myList:

1. a) \*ptr++ == 3504   
    b) \*(++ptr) == 10  
    c) --\*ptr == 4  
    d) \*--ptr == 3496