1. A class which does dynamic memory allocation most likely needs...
   A. a base class to control memory flow  
   B. a destructor to free memory  
   C. a member-wise copy function  
   D. a constructor to allocate memory

2. How are the limitations of memberwise-copy overcome?
   A. by defining an overloaded copy constructor  
   B. by defining an overloaded assignment operator  
   C. by using the default copy constructor  
   D. by relying on the default assignment operator

3. Which statement below relates to the idea of encapsulation?
   A. data is protected from users running the program  
   B. private data and public functions  
   C. functions that can only be called by data members  
   D. data members are made public

4. Which of the following most closely describes the idea of abstraction?
   A. publicly accessible data members and functions  
   B. a class that has been converted back to individual functions  
   C. class member functions have abstract names  
   D. a class which provides access to data but hides the details

5. When using inheritance, what is generally true?
   A. the data is inherited, but not the member functions  
   B. the data names are inherited, but not the data itself  
   C. the data members and functions are inherited  
   D. the base class inherits the derived class

6. Which of the following most closely describes the idea of polymorphism?
   A. a derived class that inherits different base classes  
   B. a base class that can act like many different derived classes  
   C. when a derived class changes its runtime function  
   D. a virtual class is overloaded into a base class

7. Circle the name of the base class in the following statement.
   class Student : public Person

8. Grouping of data and functions that act on the data is called what?
   Encapsulation (class = half credit)

9. A class that is never instantiated, and contains at least one pure virtual member function is known as an ___abstract___ class. (starts with A)

10. What is a pointer?
    a variable (or object) that stores a memory address.
Circle True or False

11.  T  F  A friend function has access to the private members of the class declaring it a friend.
12.  T  F  If a class dynamically allocates memory, there is no need for a copy constructor.
13.  T  F  When overloading the [] operator, an array must be passed as a parameter to the function.

14. Write code that will allocate memory to the following pointer.

```cpp
double *arr;  
arr = new double;
or arr = new double[100];
```

16. Complete the code below that will assign a random value that falls in the range -5 to 5.

```cpp
int value = rand() % 11 - 5;
```

17. Write a boolean expression that will evaluate to true if a variable named x is greater than 4 but not greater than 24. Write just the boolean expression. Sample: (a == 1). Do not write an if-statement.

```cpp
(x > 4 && x <= 24)
```

18. Fill in the blank to give the correct return type to this function that overloads an operator.

```cpp
bool operator > (const Number &v) { return (value > v); }
```

19. What is the exact output of the following code?

```cpp
int a = 1;
for (int i=0; i<5; i++) {
    a += (a/2);
    cout << a;
}
```

20. When an element is added to a queue, where is it added? Circle one: front  back

21. Which element is returned when a stack is popped?

- the top element, the last element to be pushed, the element on the stack for the shortest time, etc.

22. Name the two major functions of a queue. enqueue, dequeue

23. Your program has a stack defined with push and pop operations. Write 4 lines of code showing the function calls necessary to swap the top two elements of the stack. Assume any data type you want.

```cpp
one solution...
int a = pop();
int b = pop();
push(a);
push(b);
```
24. What is the output of `cout << foo(3);`  (circle your answer)

```cpp
int foo(int n) {
    if (n == 0)
        return 0;
    return 3 + foo(n-1);
}
```

**answer is 9.**

25. The factorial function, indicated by !, returns the product of a series of whole numbers. For example, 4! is 1*2*3*4 = 24. By definition, 0! = 1.
   Write a recursive function that will accept n as an argument and return n!.
   (hints: 4! = 4 * 3!. Do not use a for or while loop. The function is just a few lines of code.)

```cpp
int factorial(int n) {
    if (n == 0)
        return 1;
    return n * factorial(n-1);
}
```

26. Insert the following values into an initially-empty binary search tree, in this order:

```
10  8  12  14  6  7  3  9  13
```
   and draw the tree structure.

```
   10
  /  
 /    
8     12
 /    /    
6    9    14
/    /     /     
3    7    13
```

a. What is the height of the tree? **3**

b. List the nodes in pre-order: **10, 8, 6, 3, 7, 9, 12, 14, 13**

27. Write the definition of a structure used for a binary search tree node.

```cpp
struct Node {
    int data;
    Node *left;
    Node *right;
};
```

28. How do you calculate a hash table's load factor?

```
number of data elements divided by size of hash table
example of code: float loadfactor = (float)n / (float)size;
```
29. An empty hash table is below with size 10 and hash function hash(x) = x mod 10. Linear probing is used to resolve collisions. Insert the following keys into the table, in this order: 22, 15, 24, 34, 89, 88, 1, 8, 38.

30. What is the expected number of comparisons to find a key value in each of the following data structures. Give your answer in terms of n using Big-O notation.

- Linked list: O(n)
- Binary search tree: O(log n)
- Hash table: O(1)
- Ordered array: O(log n)

31. Name two algorithms we learned about that use a divide-and-conquer strategy.

accepted answers: Merge sort, Quicksort, BST, binary search, heap sort

32. Reduce the following expressions asymptotically and show using Big-O notation.

a. 4n^2 + 10n + 1024 = O(n^2)

b. log n + n^2 + 99n = O(n^2)

c. n (log n) + 100n = O(n \log n)

33. Heapify the following array values: 3 4 2 6 5 9 8 7

Circle one of the following to indicate your heapify method: top-down, bottom-up

Show the final heap as a binary tree structure.

Top-down:

```
  9
 / \    / \    / \    / \    / \
 7   8  7   8  5   4   6   5   2   3
 / \    / \    / \    / \
3   4
```

Bottom-up:

```
  9
 / \    / \    / \    / \
 7   8  7   8  5   4   6   5   2   3
 / \    / \    / \    / \
3   4
```