True (A) / False (B)  (2 pts each)

1. Structure definitions are usually local (defined inside functions).
2. A structure member is access using the index operator [], with the member name as index. []
3. The following definition of the structure variables of type myStruct s and t could be made using several definitions using the structure tag.
   ```
   struct myStruct
   {
      int i;
      double d;
   } s, t;
   ```
4. Consider the struct definition below. This will set a’s i member variable to 3.
   ```
   struct myStruct
   {
      int i;
      double d;
   };

   myStruct a;
   a->i = 3;
   ```
5. In C++, accessing an element outside your array bounds can cause a segmentation fault.
6. It is legal to assign C-string variables, i.e. str1 = str2;
7. A structure variable can be defined directly in the same statement that defines a structure definition.
8. A design technique is to break a problem into smaller tasks, with the prospect that a smaller problem will be easier to solve. If the smaller task is the identical to the original task excepting only that the size is smaller, the problem may be solved using a recursive algorithm, and implemented with a recursive function.
9. A proper recursive solution requires at least two cases: a recursive function that calls the recursive function with a smaller problem, and a base, or stopping case.
10. A recursive function must not return a value.
11. C-string literals are written ‘Hello’.
12. A C-string variable is just an array of characters.

13. It is illegal to write `char message[] = "Go Beavers!";

14. An array behaves like a list of variables each of which is of the same type and for which there is a uniform, convenient naming convention that can be declared in a single line of code.

15. When using an array, it is perfectly legal to access indexed variables with index values less than 0 or greater than or equal to the declared size of the array.

16. In the definition, `double d[10] = {1.0};`, all the elements in `d` are initialized to 1.0.

17. You can assign a C++ standard string to a C-string variable, using the `c_str()` member function of the C++ string object.

18. Dynamic variables or dynamically allocated variables in C++ are created and destroyed according to the program’s needs.

19. Indexed variables for an array are stored wherever the computer finds memory for the first indexed variable, then the next one is stored next to the first if there is space, and someplace else if not.

20. To call a function with an array parameter, write the array argument as the array name followed by empty square brackets, as in `f(a[], 7);

21. One can use the * operator to extract the value that a pointer points to.

22. With arrays, indices start at any number the programmer chooses to indicate in the definition.

23. You can use the `size()` function to determine the length of a C-string.

24. To read a character at a time, or to write a character at a time, declare a character variable `ch` and write this:
   ```
   cin >> ch;
   ```

25. You can get a pointer value to initialize a pointer variable from an object of an appropriate type with the “address-of” operator, &.

26. When declaring several pointer variables, there must be one pointer declarator * for each pointer variable, i.e. `double *p1, *p2, *p3;`
Multiple Choice  (3 pts each)

27. Which of the following is a correct statement?
   When a function having an array formal parameter is called, \texttt{void f(int a[]);}, the formal array parameter ...
   
   a) names a copy of the array argument.  
   b) refers to exactly the same array as the calling program  
   c) is passed the address of the argument, and the function needs further information about the array size to safely use an array parameter  
   d) refers to the array using a name that is always different from the calling program's argument.

28. Which of the following is a valid C-string?
   a) ‘\n’  
   b) ‘n’  
   c) ”Lamb\n”  
   d) ”Mary\0”  
   e) ”M”

29. Given the array declaration, \texttt{int a[20];}  The second element is written as:
   a) a[1]  
   b) a[0]  
   c) a  
   d) a[2]  
   e) a[3]
30. How many times is the following code invoked by the call recursive(4)?

```cpp
void recursive( int i )
{
    using namespace std;
    if (i < 8)
    {
        cout << i++ << " ";
        recursive(i);
    }
}
```

a) 2  
b) 4  
c) 8  
d) 32  
e) This is an infinite recursion.

31. In a recursive solution to a problem, we solve a problem P(n) by solving another problem P(k) where

a) P(k) is the hardest part of P(n)  
b) P(k) is a larger problem than P(n)  
c) P(k) is smaller or equal to P(n)  
d) None of the above
32. Which of the following is not a rule that ensures a correct recursive function?
   a) Each stopping case must perform a correct action (or return the correct value for stopping case) for the condition that invokes it.
   b) The recursion must make no more than 1000 recursive calls.
   c) The chain of recursive calls eventually must reach one of the stopping cases.
   d) For cases that involve recursion: If each recursive calls correctly solve the subproblem (or return the correct value for the subproblem) it solve, then the final

33. A recursive function is one that
   a) Speeds up a program.
   b) Is always slower than a nonrecursive function.
   c) Calls itself
   d) Calls no functions at all.
   e) Calls another function

34. Consider the function definition and array declarations. Which is the correct call to the function make_1?

   ```c
   void make_1 ( int a[], int size )
   {
      for (int i = 0; i < size; i++ )
         a[i] = 2;
   }
   
   int array1[20];
   a) make_1( &array1, 30 );
   b) make_1( array1, 10 );
   c) make_1( *array1, 50 );
   d) make_1( array1[], 50 );
   ```
35. Given the definition and code fragment:
   
   ```c
   int matrix[2][3];
   int k = 0;
   for(int i =0; i < 2; i++)
   for (int j=0, j < 3; j++)
   matrix[i][j] = k++;
   ```

   The value of `matrix[1][0]` is
   a) 0
   b) 1
   c) 2
   d) 3
   e) 4

36. Some pointer arithmetic is allowed. Which of the following arithmetic operators is not allowed?
   a) pointer + integer
   b) pointer + pointer
   c) pointer - integer
   d) integer * pointer
   e) both c and d

37. The header file that you must #include to have access to the Standard Library C-string functions is
   a) C-string
   b) string.h
   c) string
   d) String.h
   e) None of the above
38. Given the definitions,
   ```c
   int *p1, *p2;
p1 = new int;
p2 = new int;
   ```
Which is the correct way to make `p1` point to the same thing as `p2`?
   a) `*p1 = *p2;`
   b) `p1 = &p2;`
   c) `p1 = p2;`
   d) `*p1 = p2;`

39. Given the structure type and variable definitions
   ```c
   struct ShoeSize
   {
       char width;
       int number;
   };
   struct ShoeType
   {
       char style;
       ShoeSize size;
       double price;
   };
   ShoeType shoe1;
   ```
Which of these is an illegal call to a member variable?
   a) `shoe1.style`
   b) `shoe1.width`
   c) `shoe1.size.number`
   d) `shoe1.price`
40. Here is a recursive function. How many times is Hip, printed with the following call, 
\[ \text{rec_cheers}(5); \]
\[
\text{void rec_cheers(int n) }
\\
\{ 
\text{using namespace std; }
\text{if(1==n)}
\text{cout << "Hurray!" << endl; }
\text{else}
\text{ }
\text{cout << "Hip, "; }
\text{rec_cheers(n-1); }
\}
\}
\]
a) 5  
b) 4  
c) 3  
d) 2  
e) None of the above 

41. Which is the correct way to dynamically create a 1-d array of 6 integers on the heap? 
\[a) \text{int p1[6];} \]
\[b) \text{int *p1 = new int*[6];} \]
\[c) \text{int *p1[6];} \]
\[d) \text{int *p1 = new int[6];} \]

42. Which of the following does not define a C-string containing “Hello”?  
\[a) \text{char stringVar[10] = “Hello”;} \]
\[d) \text{char stringVar[6] = “Hello”;} \]
\[e) \text{char stringVar[] = “Hello”;} \]