FORM 2 (Please put your name and form # on the scantron!!!!)

CS 161 Exam II:

True (A)/False (B) (2 pts each):
1. The declaration below declares three pointer variables of type pointer to double that is a pointer of type 
   (double*).
   double* p1, p2, p3;  
   [F]
2. A recursive function can have local variables.  
   [T]
3. If we want to find the median of 100 scores, we need 100 separate variables to hold the data.  
   [F]
4. Consider the array declaration, int x[20];. There is no memory allocated for x[20].  
   [T]
5. C++ arrays check for out-of-range index values.  
   [F]
6. Given the two C++ array declarations:
   ```
   int a[10], b[10];
   ```
   You can successfully compute one array, say a, then assign b to a:
   ```
   a = b;
   ```
   [F]
7. A for-loop is a convenient way to step through an array.  
   [T]
8. A recursive function is a function whose definition contains a call to the function being defined.  
   [F]
9. Arrays in C++ may have several different types stored in them.  
   [F]
10. Pointer variables hold memory addresses that can be assigned to one another without regard to type.  
    [X]
11. One can use the & operator to extract the value that a pointer points to.  
    [F]
12. A static array name is a constant pointer to where the array begins in memory.  
    [F]
13. You can assign a C++ string object to a C-string variable.  
    [F]
14. Each recursion causes a new frame to be placed on the stack.  
    [T]
15. A C-string is a sequence of characters.  
    [T]
16. The C-string library functions use the null terminator to decide when to stop processing.  
    [F]
17. A pointer is a variable that holds the address of some location in memory.  
    [T]

Multiple Choice   (3 pts each)

18. Which of the following is required in a recursive function?
   a) At least one recursive call and at least one stopping case.  
      [ ]
   b) Calls to functions other than itself.  
      [ ]
   c) Zero or more stopping cases after a sequence of calls to one or more recursive cases.  
      [ ]
   d) A static local variable.  
      [ ]
   e) All of the above  
      [ ]
19. Which of the following is not a C-string containing “Hello”?
   a) char stringVar[10] = "Hello";
   b) char stringVar[10] = {'H', 'e', 'l', 'l', 'o'};
   c) char stringVar[10] = {'H', 'e', 'l', 'l', 'o', '\0'};
   d) char stringVar[] = "Hello";

20. The header file that you must #include to have access to the C++ Standard Library string class is
   a) cstring
   b) string.h
   c) string
   d) String.h
   e) None of the above

21. Given the array declaration, int a[20]; The last (legal) element is written as:
   a) a[2]
   b) a[0]
   c) a
   d) a[20]
   e) a[19]

22. Which of the following is correct?
    When a function having an array parameter is called, the array parameter
    a) points to the location of the array.
    b) refers to the pointer that points to the array in the calling function
    c) is passed the address of the argument
    d) refers to the array using a name that is always different from the argument.

23. 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 a smaller problem than P(n)
    d) None of the above

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

25. Given the definition and code fragment:
    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
26. Which of the following statements correctly returns the memory to the heap/freestore from the dynamic array pointer, p1?
   a) delete [ ] p1;
   b) delete p1[ ];
   c) delete *p1;
   d) delete p1;

27. What is the output of the following code (assuming it is embedded in a correct and complete program)?
   char letter[5] = {'o', 'k', 'c', 'e', 'g'};
   for(int i = 4; i >= 0; i-- )
      cout << letter[i];
   cout << endl;
   a) okceg
   b) gecko
   c) ecko followed by a character from an out of bounds access.
   d) kcceg followed by a character from an out of bounds access.
   e) None of the above.

28. In C++, array indices that are subscript/index values must be
   a) An unsigned integer less than the size
   b) Negative
   c) Positive numbers starting at 1
   d) Less than or equal to the declared size of the array
   e) None of these is correct

29. Which of the following array declarations is not correct?
   a) int x[4] = {8, 7, 6, 5, 4};
   b) int x[ ] = {8, 7, 6, 5, 4};
   c) int x[4] = {8, 7, 6};
   d) const int SIZE =4;
   int x[SIZE];

Given the recursive function below, answer question #30 and #31.
   void recursive( int i )
   {
      if (i < 8) {
         recursive(++i);
         cout << i << " ";
      }
   }

30. Give the output of the recursive function above when called with an argument of 5.
   a) 6 7 8
   b) 5 6 7
   c) 8 7 6
   d) 7 6 5
   e) None of the above.

31. How many times is the recursive function above invoked by the call recursive(4)?
   a) 2
   b) 3
   c) 4
   d) 5
   e) None of the above.
32. Which of the following function declarations will accept the following 2-D array?
   ```c
   int pages[10][30];
   ```
   a) `void f1(int pages[][], int size);`
   b) `void f1(int pages[][30], int size);`
   c) `void f1(int pages[10][], int size);`
   d) `void f1(int &pages, int size);`

33. Consider the following function definition:
   ```c
   void tripler(int &n)
   {
       n = 3 * n;
   }
   ```
   Given this definition, which of the following is not an acceptable function call?
   ```c
   int a[3] = {3, 4, 5}, number = 2;
   a) tripler(a[1]);
   b) tripler(a[number]);
   c) tripler(a);
   d) tripler(number);
   e) b and c
   ```

34. Given the function, and the main function calling it: What is the output of the following code if you omit the ampersand (&) from the first parameter, but not from the second parameter?
   ```c
   void func(int &x, int &y) {
       int t = x;
       x = y;
       y = t;
   }
   int main() {
       int u = 3, v = 4;
       cout << u << " " << v << endl;
       func (u, v);
       cout << u << " " << v << endl;
       return 0;
   }
   ```
   a) 3 4
   b) 3 3
   c) 3 4
   d) 4 3
   e) none of the above.

35. 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;`
36. Consider the function definition and array declaration. Which is the correct call to the function `make_1` passing `array1`?

```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, 20 );`
b) `make_1( array1, 20 );`
c) `make_1( *array1, 20 );`
d) `make_1( array1[], 20 );`

37. Consider the following function and code segment.

```c
void One( int first, int *second )
{
    first = 17;
    *second = first + 1;
}
int main() {
    int j = 4, k = 3;
    One(j, &k);
    // other code ..
}
```

After the call to `One(j, k);` what are true statements for j and k?

a) `j == 4, k == 3;`
b) `j == 17, k == 18;`
c) `j == 4, k == 18;`
d) `j == 17, k == 3;`

38. Suppose you have the following array declaration in a program.

```c
int yourArray[5];
```

Further suppose that in the implementation of C++ you are using an `int` that requires 4 bytes.

i) When your program runs, how much memory is required for this array (in bytes)?

ii) Suppose further that your array starts at memory location decimal 100. What will be the address of `yourArray[3]` (addresses based on bytes)?

iii) If you wrote to the (illegal) index 7 position in `yourArray` to what address would this clobber?

a) i) The array takes 20 bytes, ii) `yourArray[3]` will be an int located at Address 106. iii) writing to `yourArray[7]` will clobber an int starting at location 114.

b) i) The array takes 5 bytes, ii) `yourArray[3]` will be an int located at Address 103. iii) writing to `yourArray[7]` will clobber an int starting at location 107.

c) i) The array takes 20 bytes, ii) `yourArray[3]` will be an int located at Address 112. iii) writing to `yourArray[7]` will clobber an int starting at location 128.

d) The purpose of a high level language is to insulate the programmer from these details. It isn’t possible to know this without probing the source to the operating system and the compiler, or extensive debugging.
39. Given that p1 is a pointer to a C++ string object, string *p1, which of the following are legal statements?
   a) p1 = new int;
   b) cout << *p1;
   c) p1 = new char[10];
   d) *p1 = new string;
   e) B and D

   Extra Credit (2 pts):

40. Given that p1 is an integer pointer variable, and a1 is a static integer array, which of the following statements is not a legal statement?
   a) p1= a1;
   b) cout << p1[0];
   c) cin >> p1[0];
   d) a1 = p1;

41. True (A)/False(B) In C++, the preprocessor symbol NULL and the C-string terminator, the null character, share the value 0, and they are used interchangeably in the language.

42. If you write a function that should use call-by-reference, but forget to include the ampersand, 
   a) The program will not compile
   b) The program will not link
   c) The program will not run without a run-time error
   d) The program will run with incorrect results
   e) It doesn't matter

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

44. True (A)/False(B) You can check to see if all the characters in two C-strings are equal by using ==, i.e.
   str1 == str2;