CS 161 Exam 2:

FORM 1 (Please put your name and form # on the scantron!!!!)

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

1. To declare a C-string, you should use the type expression `string *`.
2. Memory cannot be allocated after a program is already running.
3. A static array name is a pointer constant because the address it represents cannot be changed during run-time.
4. A one-dimensional array can only store elements of a single data type, but a two-dimensional array can hold data of two different data types.
5. An element of a two-dimensional array is referenced to by the array name and two subscripts, first the element row number and then the element column number.
6. When you pass an array as an argument to a function, the function can modify the contents of the array.
7. To assign the entire contents of one array to another, you can use the assignment operator.
8. The amount of memory used by an array depends solely on the number of elements the array can hold.
9. If a C++ program contains the following array definition
   `int score[10];`
   the following statement would store 100 in the first array element:
   `score[1] = 100;`
10. If a function has no return statement, the flow of control moves to the next function in the file when the closing brace of the function body is reached.
11. It is possible for a function to have some parameters with default arguments and some without.
12. Recursive algorithms tend to be less efficient than iterative algorithms.
13. A recursive function can have local variables.
14. Each recursion causes a new frame to be placed on the stack.

Multiple Choice (3 pts each)

15. A function can have zero to many parameters, and it can have _____ return value(s).
    a. a maximum of ten
    b. no
    c. either zero or one
    d. either one or two
    e. zero to many

16. In a function prototype, in addition to the name of the function, you are required to furnish
    a. the data type of the return value.
    b. an identifier name for each parameter.
    c. a data type for each parameter.
    d. all of the above.
    e. A and C, but not B.

17. A _____ variable is defined inside the body of a function and is not accessible outside that function.
    a. global
    b. counter
    c. local
    d. reference
    e. constant

18. When a function just needs to use a copy of an argument passed to it, the argument should normally be passed
    a. as a default argument.
    b. by variable.
    c. by value.
    d. as a string.
    e. by reference.

19. When used as a parameter, a_____variable allows a function to access and modify the original argument passed to it.
    a. default value
    b. value
    c. static
    d. reference
    e. floating-point

20. When more than one function has the same name they are called_____functions.
    a. sister
    b. overloaded
    c. renamed
    d. parallel
    e. identical

21. The_____statement causes a function to end and the flow of control to move back to the point where the function call was made.
    a. exit
    b. end
    c. return
    d. break
    e. continue

22. What will the following code output?

```
int number = 22;
int *var = &number;
cout << var << endl;
```

    a. The address of the `number` variable
    b. 22
    c. An asterisk followed by 22
    d. An asterisk followed by the address of the number variable

23. Unlike regular variables, arrays can hold multiple
    a. values.
    b. named constants.
    c. variables.
    d. data types.
    e. operators.

24. Given the following function definition

```
void calc (int a, int& b)
{
   int c;

   c = a + 2;
   a = a * 3;
   b = c + a;
}
```

What is the output of the following code fragment that invokes calc?
(All variables are of type *int*)

```
x = 1;
y = 2;
calc(x, y);
cout << x << "  " << y << endl;
```

    a. 1 2
    b. 1 6
    c. 3 6
    d. 1 14
    e. None of these

25. When an array is passed to a function, it is actually_____the array that is passed.
    a. the data type and size of
    b. the value stored in the first element of
    c. the starting memory address of
    d. the data type and name of
    e. a copy of all the values in

26. The null terminator stands for the ASCII code
    a. 100.
    b. 1000.
    c. 57.
    d. 0.
    e. None of the above

27. The function

```
int fact(int k)
{
   return k*fact(k-1);
   if (k==0) return 1;
}
```

    a. does not correctly handle its base case.
    b. computes the factorial on an integer k passed to it as parameter.
    c. returns the value 1 if it is passed a value of 0 for the parameter k.
    d. works for all non-negative values of k, but not for negative numbers.
    e. None of the above

28. If you leave out the size declarator in an array declaration
    a. you must furnish an initialization list.
    b. the value of each array element is set to a default value of 0.
    c. the array size defaults to 100 elements.
    d. the array will contain no elements.
    e. the array cannot be created

29. The_____, also known as the address operator, returns the memory address of a variable.
    a. exclamation point (!)
    b. asterisk (*)
    c. percent sign (%)
    d. ampersand (&)
    e. None of the above

dereference

30. The statement double *num;
    a. initializes a variable called *num.
    b. defines a variable of type double called num.
    c. defines and initializes a pointer variable called num.
    d. defines a pointer variable called num.
    e. None of the above

31. Assuming that arr is an array identifier, the statement sum += *arr;
    a. will always result in a compiler error.
    b. adds the value stored in arr[0] to sum.
    c. is illegal in C++.
    d. adds the address of the pointer arr to sum.
    e. None of the above

32. The delete operator should only be used on pointers that
    a. point to storage allocated by the new operator.
    b. have not yet been used.
    c. have been correctly initialized.
    d. are appropriately dereferenced.
    e. None of the above

33. Which of the following statements is not valid C++ code?
    a. float num1 = &ptr2;
    b. int ptr = &num1;
    c. int ptr = int *num1;
    d. All of the above are valid.
    e. All of the above are invalid.

34. Which of the following statements correctly deletes a dynamically-allocated array pointed to by p?
    a. delete p;
    b. delete array p;
    c. delete [ ] p;
    d. p delete[ ];
    e. None of the above

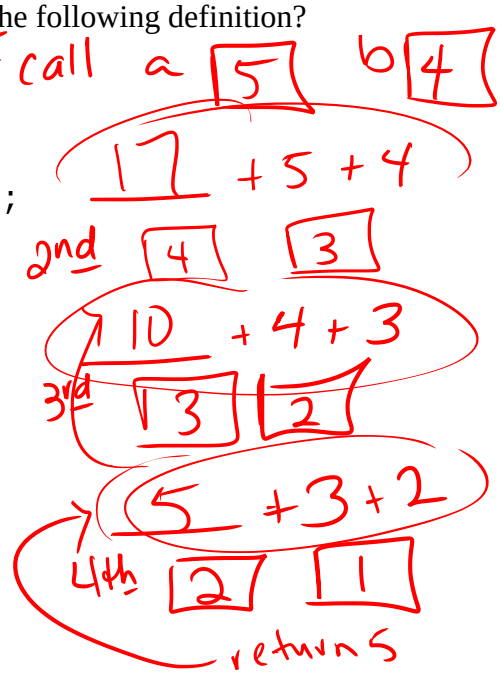
35. If arr is an array identifier and k is an integer, the expression arr[k] is equivalent to
    a. arr + k.
    b. arr + *k.
    c. *(arr + k).
    d. &arr[k].
    e. None of the above

36. A recursive function that does not correctly handle its base case may
    a. cause a continuous chain of recursive calls.
    b. return FALSEand stop.
    c. reach the NULL terminator and stop.
    d. return 0 and stop.
    e. None of the above

37. What would be the result of the call `doTask (5, 4)`, given the following definition?

```
int doTask (int a, int b)  {
     if (a <= 2)
          return 5;
     else
          return doTask(a-1, b-1) + a + b;
}
```

a. 5
b. 10
c. 17
d. 26
e. None of the above



38. An array of characters ending with the null terminator is called
a. a C++ String.
b. a string class object.
c. a C-string.
d. a string literal.
e. None of the above

Extra Credit (2 pts each):

39. The _____ of recursion is the number of times a recursive function calls itself.
a. depth
b. level
c. type
d. breadth
e. None of the above

40. True (A) / False (B): C++ arrays check for out-of-range index values. F

41. The statement

```
int grades[ ] = { 100, 90, 99, 80 };
```

is an example of
a. default arguments.
b. implicit array sizing.
c. data encapsulation.
d. an illegal array declaration.
e. an illegal array initialization.

explicit is int grades[4]

42. The expression strcmp("ab", "ab")returns 0
a. the value equivalent to false.
b. a non-zero positive integer
c. a negative integer.
d. the boolean value true.
e. None of the above

43. An array called `aList` contains integers 5, 3, 7, 2, 8. What are the contents of aList after the function call `workOnArray (aList, 4)`, if and the definition of workOnArray is:

```
int workOnArray (int a[], int n)
{
      if (n == 1)
            return a[0] + 3;
      else
      {
            a[n] = workOnArray (a, n-1);
            return 7;
      }
}
```

a. 5, 3, 8, 8, 8
b. 5, 6, 10, 5, 11
c. 5, 3, 8, 7, 7
d. 8, 6, 10, 5, 11
e. None of the above