Winter 2019 CS 161 Exam II FORM 1

Please put your name, ID number and form number on the scantron, and leave the section number blank!!!!

Please use a No. 2 pencil to fill the scantron!!!!

True(A) / False(B) (28 pts, 2 pts each):

1. The name of an array stores the value of the first array element.

2. In a function with pass-by-reference parameters, the values of the actual arguments are passed to the function.

3. C++ performs array bounds checking, making it impossible for you to assign a pointer the address of an element out of the boundaries of an array.

4. A recursive function can have at most one base case.

5. A recursive function must return a value.

6. In some cases, using recursion enables you to give a natural, straightforward, simple solution to a program that would otherwise be difficult to solve.

7. You should not assign a non-zero, random integer value to a pointer variable.

8. Default arguments can be located anywhere in the list of function parameters.

9. A static array name is a constant pointer because the address stored in it cannot be changed.

10. Assume **`array1`** and **`array2`** are the names of two arrays. To assign the contents of **`array2`** to **`array1`**, you would use the following statement:
    `array1 = array2`;

11. The elements accessed using two square brackets in a two-dimensional array are all the same data type.

12. The C string **`company[12]`** can hold 12 characters and a **`'\0'`** character.

13. The size of a static array is defined at compile time.

14. Just like pointers, you can change what any C++ reference refers to at any point.

Multiple Choice (72 pts, 3 pts each)

15. If your program makes too many recursive function calls, your program will cause a _____.
    A. compiler error
    B. runtime error
    C. syntax error
    D. heap error

Questions 16-19 are based on the following code:

```
void read_strings(     ③      , int size) {
      for (int i = 0; i < size; i++)
            cin >> array[i];
}
int main(){
      string *arr = new string [5];
      read_strings(    ①     , 5); //Function call
      _____②_____ //Free memory
      return 0;
}
```

16. Which of the following is valid at ①?
    A. **arr[5]**
    B. **arr**
    C. ***arr**
    D. **&arr**

17. Which of the following will not give you a memory leak at ②?
    A. **delete arr;**
    B. **delete arr [];**
    C. **delete [] arr;**
    D. **delete *arr;**

18. Which of the following is valid at ③?
    A. **string *array;**
    B. **string &array;**
    C. **string arr[];**
    D. **A and C;**

19. Assume ① - ③ are correct, is each element in **arr** filled after calling **read_strings**?
    A. Yes. The memory address of the array has been passed into the function, and therefore we can change the content in the original array.
    B. No. We passed the copy of the array into **read_strings**, and thus the content in the original array does not change.

20. If a C-style string variable **word** stores "apple", what will **strlen(word)** return?
    A. 0
    B. 4
    C. 5
    D. 6

Questions 21-25 are based on the following code:

```
void test( ① ){
      // code
}
int main () {
      int a = 0;
      int* ptr1 = &a;
      int** ptr2 = &ptr1;
      test( ② );
      return 0;
}
```

21. Which of the following should be filled into ② if ① is **`int &var`** ?
    A. **`&a`** or **`*ptr1`**
    B. **`a`** or **`ptr1`**
    C. **`a`** or **`*ptr1`**
    D. **`&a`** or **`ptr1`**

22. Which of the following should be filled into ② if ① is **`int *var`** ?
    A. **`&a`** or **`*ptr1`**
    B. **`a`** or **`ptr1`**
    C. **`a`** or **`*ptr1`**
    D. **`&a`** or **`ptr1`**

23. Which of the following should be filled into ② if ① is **`int **var`** ?
    A. **`*ptr1`** or **`ptr2`**
    B. **`&ptr1`** or **`*ptr2`**
    C. **`*ptr1`** or **`*ptr2`**
    D. **`&ptr1`** or **`ptr2`**

24. Which of the following should be filled into ① if ② is **`&ptr1`** ?
    A. **`int var`**
    B. **`int *var`**
    C. **`int **var`**
    D. **`int ***var`**

25. Which of the following should be filled into ① if ② is **`&ptr2`** ?
    A. **`int var`**
    B. **`int *var`**
    C. **`int **var`**
    D. **`int ***var`**

26. 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][2]** is

A. 2
B. 3
C. 4
D. 5
E. 6

27. What is the output of the following function call, given the function definition below?

```
int F(int n){
     if (n == 1)
          return 1;
     if (n == 2)
          return 2;
     if (n == 3)
          return 3;
     return F(n-1) + F(n-2) + F(n-3);
}
cout << F(5) << endl;
```

A. 5
B. 6
C. 11
D. 20

28. Assuming **ptr** is a pointer variable, what will the following statement output?

```
cout << *ptr;
```

A. The value in the variable whose address is stored in **ptr**.
B. The string **"*ptr".**
C. The address of the variable whose address is stored in **ptr**.
D. The address of the variable stored in **ptr**.

Questions 29-32 are based on the following code:

```
void func ( ____①____  , int a, int b) {
      //code
}
int main(){
      int **two_d_arr = new int*[3];
      for(int i = 0; i < 3; i++){
            two_d_arr[i] = new int[4];
            for(int j = 0; j < 4; j++)
                  two_d_arr[i][j] = 1;
      }
      func (___②___  , 3, 4);
      _________③_________ //Free memory
      return 0;
}
```

29. C++ is row major, meaning that a two-dimensional array will be laid out by rows in memory. How many rows and columns does **two_d_arr** have, based on the code above?

A. 3 rows, 4 columns
B. 4 rows, 3 columns
C. Cannot be defined
D. None of the above

30. According to the code above, which of the following is correct?

A. The double pointer **two_d_arr**, the row pointers, and the columns are all on the stack
B. The double pointer **two_d_arr** and row pointers is on the stack, and the columns are on the heap
C. The double pointer **two_d_arr** is on the stack, and the row pointers and columns are on the heap
D. The double pointer **two_d_arr**, the row pointers, and the columns are all on the heap

31. Which of the following is valid for ① and ②?

A. ① **int** array** ② **two_d_arr**
B. ① **int** array** ② **&two_d_arr**
C. ① **int array[][]** ② **two_d_arr**
D. ① **int array[3][4]** ② **two_d_arr**

32. Which of the following will not give you a memory leak or segmentation fault at ③?

A.
```
for (int i = 0; i < 3; i++)
      delete [] two_d_arr[i];
  delete [] two_d_arr;
```
B.
```
for (int i = 0; i < 4; i++)
      delete [] two_d_arr[i];
```
C.
```
delete [] two_d_arr;
```
D.
```
delete [][] two_d_arr;
```

33. What is the output of the following code, given the function definitions below?

```
void tester (int a, int &b){

      int c = 0;

      c = a + 2;

      a = a * 3;

      b = c + a;

}

int main () {

      int x = 2, y = 3;

      tester(y, x);

      cout << x << " " << y << endl;

      return 0;

}
```

A. 2 3
B. 2 10
C. 14 3
D. 14 9

34. Which of the following statement is valid C++ code?

A.
```
int num = 0;
int ptr = &num;
```
B.
```
float *ptr = 10.04;
```
C.
```
char word1[5] = "word";
char word2[10] = "computer";
word1 = word2;
```
D. None of the above

35. Which of the following is a valid C string declaration?

A. `char array[5] = "hello" ;`
B. `char array[4] = "hello";`
C. `char array[] = "hello";`
D. None of the above

36. Which of the following is a valid C++ array declaration?

A. `int []array;`
B. `float payments[10.23];`
C. `string numbers[];`
D. `double scores[25];`

37. Which of the following statement is true about function overloading?

A. Overloaded functions may have the same name, same parameters, and same return type.
B. Overloaded functions may have the same name, same return type, but different parameters.
C. Overloaded functions may have the same name, same parameters, but different return type.
D. B and C

38. What does the following statement do?

```
int *ptr = NULL;
```

A. Create a variable named **`*ptr`** that will store an integer value.
B. Create a variable named **`*ptr`** that will store an asterisk (*) and an integer value.
C. Create a pointer variable named **`ptr`** that will store the address of an integer variable.
D. Create a variable named **`*ptr`** that will store the **`NULL`** value.

Extra Credit (2 pts each):

39. What is wrong with the following code?

```
int *p1, *p2;

p1 = new int;

p2 = new int;

*p1 = 11;

*p2 = 0;

delete p2;

p2 = p1;

delete p1;
```

A. nothing
B. cannot reuse p2 after deleting it
C. You have a memory leak, since p2 is not deleted after re-assigned.
D. B and C

40. Which of the following is true about this statement:

```
sum += *array++;
```

A. This statement is illegal in C++.
B. This statement adds the dereferenced pointer's value to sum, then increments the address stored in the pointer.
C. This statement increments the dereferenced pointer's value by one, then adds that value to sum.
D. None of the above

41. What is the output of the following code?

```
int array[2][3];

for(int i = 0; i < 2; i++){

    for(int j = 0; j < 3; j++){

        array[i][j] = i + j;

        cout << array[i][j] << " ";

    }

    cout << endl;

}
```

A. 0 1 2
   1 2 3

B. 0 1
   1 2
   2 3

C. 0 0 0
   1 1 1

D. 0 0
   1 1
   2 2

42. True(A)/False(B) You may use **`strcmp()`** to compare all the elements between two C-style strings.

43. True(A)/False(B) Elements in a two-dimensional static array have contiguous memory on the heap.