The Exam Topics:

You are expected to know everything that was covered in the class, but give special attention to the following topics:
Strings and string manipulation
Pass Arrays and Primitive Values to functions and returning them
Sorting
Precedence rules
Control structures (focus on dangling if-else, fall through of switch, translating between them)
Basic IO and program structure

The Exam Format:

Instructor will finalize it but at the moment I think we will have seven questions that you should be able to complete in 100 minutes. One type of questions will check your understanding of the language tools and other questions will check your ability to apply those tools effectively in solving problems.

To check your understanding of C++ you can be asked to recognize how a program is behaving and if there is an error in the code based on language rules. I will strongly encourage you to actually write some programs and go over the rules of expression evaluation, precedence, the control structures, and how the various loops work and related to each other.

To check your problem solving abilities using C++ you will be given two concrete problems for which you will be required to write working programs. If you have a good understanding of language and tools at your disposal (an art that is usually learnt by hands on programming) you will be able to formulate an efficient solution very quickly. We strongly encourage you to write your solution in plain English or pseudo code before writing the actual code, which can actually earn you significant credit as it increases instructors understanding of your approach.

Sample Questions:

Q1. State which of the following are true and which are false. If false, explain your answers.

a. C++ operators are evaluated from left to right.
b. The following are all valid variable names: `_under_bar_`, `m928134`, `t5`, `j7`, `her_sales`, `his_account_total`, `a`, `b`, `c`, `z`, `z2`.
c. The statement `cout << "a = 5;";` is a typical example of an assignment statement.
d. A valid C++ arithmetic expression with no parentheses is evaluated from left to right.
e. The following are all invalid variable names: `3g`, `87`, `67h2`, `h22`, `2h`.

Q2. Write a program that asks the user to enter two integers, obtains the numbers from the user, then prints the larger number followed by the words `"is larger."` If the numbers are equal, print the message `"These numbers are equal."`

Q3. Write a program that reads in two integers and determines and prints if the first is a multiple of the second.

Q4. Write a program that inputs a five-digit integer, separates the integer into its individual digits and prints the digits separated from one another by three spaces each. [Hint: Use the integer division and modulus operators.] For example, if the user types in 42339, the program should print:

```
4   2   3   3   9
```

Q5. State the output for each of the following when `x` is `9` and `y` is `11` and when `x` is `11` and `y` is `9`. Note that the compiler ignores the indentation in a C++ program. The C++ compiler always associates an `else` with the previous `if` unless told to do otherwise by the placement of braces `{}`. On first glance, the programmer may not be sure which `if` and `else` match, so this is referred to as the "dangling-`else`" problem. We eliminated the indentation from the following code to make the problem more challenging. [Hint: Apply indentation conventions you have learned.]

a.

```
if ( x < 10 )
if ( y > 10 )
cout << "*****" << endl;
else
cout << "#####" << endl;
cout << "$$$$$" << endl;
```

b.

```
if ( x < 10 )
{
if ( y > 10 )
cout << "*****" << endl;
}
else
{
cout << "#####" << endl;
cout << "$$$$$" << endl;
}
```

Q6. Write a program that reads in the size of the side of a square and then prints a hollow square of that size out of asterisks and blanks. Your program should work for squares of all side sizes between 1 and 20. For example, if your program reads a size of 5, it should print

```
*****
*   *
*   *
*   *
*****
```

Q7. Find the error(s) in each of the following:

```
a.
For ( x = 100, x >= 1, x++ )
   cout << x << endl;
```

b. The following code should print whether integer `value` is odd or even:

```
switch ( value % 2 )
{
   case 0:
      cout << "Even integer" << endl;
   case 1:
      cout << "Odd integer" << endl;
}
```

c. The following code should output the odd integers from 19 to 1:

```
for ( x = 19; x >= 1; x += 2 )
   cout << x << endl;
```

d. The following code should output the even integers from 2 to 100:

```
counter = 2;

do
{
   cout << counter << endl;
   counter += 2;
} While ( counter < 100 );
```

Q8. Write a program that uses `for` statements to print the following patterns separately, one below the other. Use `for` loops to generate the patterns. All asterisks (`*`) should be printed by a single statement of the form `cout << '*';` (this causes the asterisks to print side by side). [Hint: The last two patterns require that each line begin with an appropriate number of blanks. Extra credit: Combine your code from the four separate problems into a single program that prints all four patterns side by side by making clever use of nested `for` loops.]

```
(a)           (b)               (c)             (d)
*             **********        **********               *
**            *********          *********              **
***           ********            ********             ***
****          *******              *******            ****
*****         ******                ******           *****
******        *****                  *****          ******
*******       ****                    ****         *******
********      ***                      ***        ********
*********     **                        **       *********
**********    *                          *      **********
```

Q9. Assume `i = 1`, `j = 2`, `k = 3` and `m = 2`. What does each of the following statements print? Are the parentheses necessary in each case?

a. `cout << ( i == 1 ) << endl;`
b. `cout << ( j == 3 ) << endl;`
c. `cout << ( i >= 1 && j < 4 ) << endl;`
d. `cout << ( m <= 99 && k < m ) << endl;`
e. `cout << ( j >= i || k == m ) << endl;`
f. `cout << ( k + m < j || 3 - j >= k ) << endl;`
g. `cout << ( !m ) << endl;`
h. `cout << ( !( j - m ) ) << endl;`
i. `cout << ( !( k > m ) ) << endl;`

Q10. Determine whether each of the following is true or false. If false, explain why.

a. To refer to a particular location or element within an array, we specify the name of the array and the value of the particular element.
b. An array declaration reserves space for the array.
c. To indicate that 100 locations should be reserved for integer array `p`, the programmer writes the declaration
d. `p[ 100 ];`
e. A `for` statement must be used to initialize the elements of a 15-element array to zero.
f. Nested `for` statements must be used to total the elements of a two-dimensional array.

Q11. Use a one-dimensional array to solve the following problem. A company pays its salespeople on a commission basis. The salespeople each receive $200 per week plus 9 percent of their gross sales for that week. For example, a salesperson who grosses $5000 in sales in a week receives $200 plus 9 percent of $5000, or a total of $650. Write a program (using an array of counters) that determines how many of the salespeople earned salaries in each of the following ranges (assume that each salesperson's salary is truncated to an integer amount):

a. $200$299
b. $300$399
c. $400$499
d. $500$599
e. $600$699
f. $700$799
g. $800$899
h. $900$999
i. $1000 and over

Q12. Find the error(s) in each of the following statements:

a.

Assume that: `char str[ 5 ];`

```
cin >> str; // user types "hello"
```

b.

Assume that: `int a[ 3 ];`

```
cout << a[ 1 ] << " " << a[ 2 ] << " " << a[ 3 ] << endl;
```

c.

```
double f[ 3 ] = { 1.1, 10.01, 100.001, 1000.0001 };
```

d.

Assume that: `double d[ 2 ][ 10 ];`

```
d[ 1, 9 ] = 2.345;
```

Q13. Given a string (character array), reverse it by operating over its character array. For example when I input “Hello World!” your program should output “!dlroW olleH”

Q14. Find a substring in a string. For example given “My name is Zubair” and you are required to find “Zubair” in this string then you should return ‘11’ (the start of substring Zubair), but if you are asked to find “Zaffar” you should return ‘-1’ to indicate that the substring was not found.

Q15. Write a program that understands following commands:
{Plus, Minus, Multiply, Divide}
Such that if the user inputs on command line:

2 Plus 2
The program should Output
= 4