Computing Bootcamp
Summer, 2017

Try your hand at these questions to test your proficiency in the foundational material in C++. These questions are from the first-quiz in *IE523: Financial Computing* from the past years. If you were able to work through all of them comfortably, you do not need to attend the Computing Bootcamp. If you struggled with these topics, I recommend you attend the Computing Bootcamp.

1. What is the output generated by the C++ code shown in figure 1?

```
#include <iostream>
#define CUBE(x) x*x*x
int main()
{
        std::cout << CUBE((3+2)) << std::endl;
}
```

Figure 1: Problem 1.

2. What is the output generated by the C++ code shown in figure 2.

```
#include <iostream>
#define CUBE(x) (x)*(x)*(x)
int main()
{
    std::cout << CUBE(3+2) << std::endl;
}
```

Figure 2: Problem 2. Note the difference between this piece of code, and the one shown in figure 1.

3. What is the output generated by the C++ code shown in figure 3.

```
#include <iostream>
#define CUBE(x) x*x*x
int main()
{
        std::cout << CUBE(3+2) << std::endl;
}
```

Figure 3: Problem 3. Note the difference between this piece of code, and the ones shown in figures 1 and 2.

4. Consider the C++ code shown in figure 4. Present a two/three sentence explanation for what the program would present as output.

```
#include <iostream>
int avar = 5;
int main()
{
  int avar = 10;
  {
    int avar = 20;
    {
      int avar = 30;
        std::cout << "avar = " << avar << ", avar = " << ::avar << std::endl;
    }
  }
}
```

Figure 4: Problem 4.

5. What is the output generated by the C++ code shown in figure 5?

```
#include <iostream>
using namespace std;
int main (int argc, char* argv[])
{
  int x, y, *p, *q;
  p = &x;  q = &y;  x = 35;  y = 46;   *p = 78;
  cout << x << " " << y << endl;
  cout << *p << " " << *q << endl;
  return (0);
}
```

Figure 5: Problem 5.

6. Consider the C++ code shown in figure 6. Notice that in the functions' parameters the first dimension of array `a` is left unspecified while the second (i.e. `[5]`) and third dimension (i.e. `[6]`) is specified. Why is this? How is a multi-dimensional array stored in C++.

7. What is the output generated by the C++ code shown in figure 7? Give me short reason for your answer.

8. What is the output generated by the C++ code shown in figure 8? Give me short reason for your answer. Notice the difference in the `cout` statement in figure 8 as compared to the one in figure 7.

```
void print(const a[][5][6]) {
        for (int i = 0; i < 3; i++)
                for (int j = 0; j < 5; j++)
                        for (int k = 0; k < 6; k++)
                                cout << a[i][j];
}
```

Figure 6: Problem 6.

```
#include <iostream>
using namespace std;
int main (int argc, char* argv[])
{
 double **array = new double*[2];
 for (int i = 0; i < 2; i++)
   array[i] = new double[2];
 array[0][0] = 2; array[0][1] = 4; array[1][0] = 6; array[1][1] = 8;
 cout << ++(*array[1]) << endl;
 return (0);
}
```

Figure 7: Problem 7.

9. Consider the C++ code shown in figure 9, which defines a function f that takes two arguments n and m.
   (a) Which variable is passed by *reference*?
   (b) Which variable is passed by *value*?
   (c) Which variable is passed by *address*?
   (d) What is would the following line present as output?

       cout << f(1234, 0) << endl;

   (e) Give me the reasoning behind your answer for part 9d.
   (f) Can you give me an educated guess as to what the function f computes?
10. Consider the C++ code shown in figure 10. Present a two/three sentence explanation for what the program would present as output.
11. Consider the C++ code shown in figure 11, which adds two matrices m1 and m2, and returns the answer.
    (a) What kind of matrices is this function adding? That is, is it adding int-, long-, float-, double- or complex-matrices?
    (b) What does the line T **result = new T*[n]; accomplish? What does the for-loop that follows this line accomplish?

```
#include <iostream>
using namespace std;
int main (int argc, char* argv[])
{
 double **array = new double*[2];
 for (int i = 0; i < 2; i++)
   array[i] = new double[2];
 array[0][0] = 2; array[0][1] = 4; array[1][0] = 6; array[1][1] = 8;
 cout << *(++array[1]) << endl;
 return (0);
}
```

Figure 8: Problem 8. Notice the difference in the cout statement here compared to the one in figure 7.

```
int f(int n, int & m)
{
        if (n < 10)
                return (m+1);
        else {
                m++;
                return (f(n/10, m));
        }
}
```

Figure 9: Problem 9.

12. What is the output of the C++ code shown in figure 12. FYI, putchar writes a character argument to your screen, and the *modulus operator* $m\%n$ returns the remainder after (integer) m is divided by (integer) n.

13. What is the output of the C++ code shown in figure 13. Note there is a change in the putchar statement here compared to that in figure 12.

```
#include <iostream>
using namespace std;

int main()
{
        string *p, name("Fooey"), schmame("Schmooey");
        p = &schmame;
        if (2+2 == 4)
        {
                p = &name;
                cout << *p << endl;
        }
        cout << *p << endl;
}
```

Figure 10: Problem 10.

```
template <typename T>
T **matrix_add(T **m1, T **m2, int n)

{
    int i, j;
    T **result = new T*[n];
    for (i = 0; i < n; i++)
        result[i] = new T[n];

    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            result[i][j] = m1[i][j] + m2[i][j];

    return (result);
}
```

Figure 11: Problem 11.

```
#include <iostream>
using namespace std;

void print_integer (int num)
{
    // cf. http://stackoverflow.com/questions/14646718/if-statement-integer
        if (num / 10)
                print_integer(num / 10);
        putchar (num % 10 + '0');
}
int main()
{
        print_integer (1234);
        cout << endl;
}
```

Figure 12: Problem 12.

```
#include <iostream>
using namespace std;

void print_integer (int num)
{
        if (num / 10)
                print_integer(num / 10);
        putchar (num % 10 + 'A');
}
int main()
{
        print_integer (1234);
        cout << endl;
}
```

Figure 13: Problem 13. Note there is a change in the putchar statement here compared to that in figure 12.