CSCI 200: Foundational Programming Concepts & Design

Exam I Review

1. What is the result?

a) 2 + 3 * 4 – 6 8

b) 5 + 11 / 3 8.6

c) 11 % 3 * 4 8

d) (2 + 1) * 3 - 1 8

2. Create boolean test conditions

a) myHeight is greater than 2 `myHeight > 2`

b) y is odd and less than 10 `y@2 == 1 && y < 10`

c) At least one of x or y is 3 `x == 3 || y == 3`

d) t is between 2.1 and 2.3 inclusive

`t >= 2.1 && t <= 2.3`

3. What is the output?

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

int main() {
    int x = 12;

    if( (x >= 2) || (x != 17) )
        cout << x << endl;
    else
        cout << "Have a good day" << endl;

    return 0;
}
```

12

4. What is the output?

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

int main() {
    int x = 1;

    if( (x >= 2) || (x != 17) )
        cout << x << endl;
    else
        cout << "Have a good day" << endl;

    return 0;
}
```

1

5. What is the output?

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

int main() {
    int x = 17;

    if( (x >= 2) && (x != 17) )
        cout << x << endl;
    else
        cout << "Have a good day" << endl;

    return 0;
}
```

Have a good day

6. What is the output?

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

int main() {
    int x = 11, y = 5;

    int answer;
    answer = x / y;
    cout << answer << endl;

    return 0;
}
```

```
the remainder is discarded.
2
```

7. What is the output?

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

int main() {
    int x = 9, y = 2;

    cout << x / y << endl;
    cout << (double)x / (double)y << endl;
    cout << (double)x / y << endl;
    cout << x / (double)y << endl;

    return 0;
}
```

4

4.5

4.5

4.5

8. What is the output?

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

int main() {
    int x = 5, y = 10;

    y = x++;
    cout << x << " " << y << endl;   6 5

    y = ++x;
    cout << x << " " << y << endl;   7 7

    return 0;
}
```

9. Find the Errors

```
#include <iostream>
using namespace std

int main() {
    int x = 6;
    double y = 2.5;
    z = 1;
    cin << z;
    if( x = y )
        cout "x and y match";
    else
        cout "x and y do not match";

    return 0;
}
```

x and y do not match

10. Write if/else code

a) Write a series of if statements (use only if) that will output a student's letter grade based on the input. Assume the input (already received) is called examScore and that the value of examScore is greater than 70 and less than 100.

```
if (examScore >= 90 && examScore < 100) { cout << "A"; }
if (examScore >= 80 && examScore < 90) { cout << "B"; }
if (examScore >= 70 && examScore < 80) { cout << "C"; }
```

b) Write an if block (if and else if) that will output a student's letter grade based on the input. Assume the input (already received) is called examScore and that the value of examScore is greater than 70 and less than 100.

```
if (examScore >= 90) { cout << "A"; }
else if (examScore >= 80) { cout << "B"; }
else if (examScore >= 70) { cout << "C"; }
```

11. Write Loop code

a) Write a snippet of code that prints all odd numbers between 0 and X (inclusive), where X is given by the user. Use a while loop.

b) Write a snippet of code that prints all odd numbers between 0 and X (inclusive), where X is given by the user. Use a for loop.

```
int x;
cin >> x;
int count = 0;
while (count <= x)
{

    if (count % 2 == 1)
    {
        cout << "odd number: " << count << endl;
    }
    count++;
}
```

```
for (int i = 0; i <= x; i++)
{
    if (i % 2 == 1)
    {
        cout << "odd number: " << i << endl;
    }
}
```

12. Rewrite as a switch

```
if( (rank == 1) || (rank == 2) )
    cout << "Lower division" << endl;
else {
    if( (rank == 3) || (rank == 4) )
        cout << "Upper division" << endl;
    else {
        if( rank == 5 )
            cout << "Graduate student" << endl;
        else
            cout << "Invalid rank" << endl;
    }
}
```

```
switch (rank) {
    case 1:
    case 2:
        cout << “Lower div” << endl;
        break;
    case 3:
    case 4:
        cout << “Upper div” << endl;
        break;
    case 5:
        cout << “Gradu std” << endl;
        break;
    default:
        cout << “Invalid rank” << endl;
        break;
}
```

13. True or False

a) The statement "x++" adds one to x. T

b) A semi-colon is needed at the end of a while code block. F

c) Once a constant variable has been created, it cannot be changed. T

d) Boolean variables store the values always true, always false, or sometimes true. T

14. Rewrite as a for loop

a)

```
int i = 2;
while( i <= 18 ) {
    cout << "*";
    i += 3;
}
```

```
for (int i = 2; i <= 18; i+=3) {
    cout << “*”;
}
******
```

b) What is the output?

15. What is the output?

```
int number = 0;
int sum = 0;
int limit = 20;

while( number > limit ) {
    sum += number;
    number += 2;
}

cout << sum << endl;
```

0

16. What is the output?

```
int number = 100;
int sum = 0;
int limit = 20;

while( number > limit ) {
    sum += number;
    number += 2;
}

cout << sum << endl;
```

```
number: 100
sum: 100

number: 102
sum: 202
…

infinite loop
```

17. What is the output?

```
int number = 0;
int sum = 0;
int limit = 20;

while( number < limit ) {
    sum += number;
    number += 2;
}

cout << sum << endl;
```

```
number: 0
sum: 0

number: 2
sum: 2

sum = 0+2+4+….+18=20*8+10=90
```

18. What is the output?

```
for( int i = 0; i < 4; i++ ) {
    for( int j = i; j < 6; j++ )
        cout << "*";
    cout << endl;
}
```

```
0,1,2,3

0 1 2 3
1 2 3 4
2 3 4 5
3 4 5
4 5
5

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

19. Random Numbers

- Write a program that prints 10 random numbers between 1 and 100. Use an appropriate seed.

```
random_device rd;
mt19937 gen( rd() );
uniform_int_distribution<int>dist(1,100);

for (int i = 0; i < 10; i ++ ) { cout << dist(gen) }
```

20. True or False

a) void functions return a value. F

b) Function prototypes do not require parameter names. T

```
// Both are valid prototypes:
int add(int x, int y);     // With parameter names
int add(int, int);         // Without parameter names (just types)
```

21. What is printed?

```
void my_func( int x, int y ) {
    x = 52;
    y = 7;
}

int main() {
    int x = 0;
    int y = 0;
    my_func( x, y );
    cout << "x = " << x << endl;0
    cout << "y = " << y << endl;0
    return 0;
}
```

22. What is printed?

```
int my_function( double a, double b, double c ) {
    a = 2 * b;
    b = 15 + c;
    c = 3 * a;
    return (a + b + c);
}

int main() {
    double a = 1;
    double b = 2;
    double c = 3;
    double d = my_function( a, b, c );    6
    cout << "d = " << d << endl;
    return 0;
}
```

23. Which are legal statements?

```
void func_A( int x, int y, int z );
int func_B( int x, double y );
```

b, d, e

```
a) cout << func_A( 5, 4, 3 ) << endl;
b) cout << func_B( 5, 4.0 ) << endl;
c) func_A( 5, 4 );
d) func_A( 5, 4.7, 3 );
e) int x = func_B( 5, 6 );
f) int y = func_A( 5, 4, 3 );
```

24. Write Function Prototypes

a) Write a function prototype with the name "cool_func" that has no parameters and does not return a value.

```
void cool_func(){};
```

b) Write a function prototype with the name "hot_func" that has no parameters and returns a double.

```
double hot_func();
```

c) Write a function prototype with the name "neutral_func" that returns an integer and whose parameters in order are an integer named foo, a double named bar, and a character named baz.

```
int neutral_func(int foo, double bar, char baz);
```

25. Write a Function

- Write a function that calculates and returns the area of a square for whole numbers.

```
int getArea(int x) { return x * x; }
```

26. Find the errors

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

int main() {
    int 7;
    add_one( x );
    cout >> "7 plus one is " << x << endl;
    return 0;
}

void add_one( int x ) {
  ++x;
}
```

add_one is not defined

27. Memory Time

- With Two's Complement and Floating Point Binary Representation, what does the leading bit correspond to?

  leading bit = sign bit
  it's used to store negative or positive

- What does this do to the number of integers that can be stored?

it results in a different distribution of
the integers that can be stored in memory.
It moves half of the storing capacity to store negative integers

28. Data Type Modifiers

- What affect to the allowable values of a standard **`int`** data type do the following modifiers have?
  - **`unsigned long long int`**
  - **`long long int`**

```
int: 32 bits (4 bytes)
unsigned: Removes sign bit, eliminates negative numbers
long long: Increases size to 64 bits (8 bytes)
```

- What's the difference between the two?

```
unsigned long long int: can only store positive integers with bigger range
long long int: can store both negative / positve integers with less range
```

29. Data Type Modifiers

- What is the output of the following code?

```
unsigned short int q = -1;
cout << "q is: " << q << endl;
```

a) -1

b) 65535

c) Compiler Error

d) Runtime Error

```
unsigned can't store negative integers, it wraps around.
For 16-bit unsigned: 1111111111111111 = 65,535
```

- Why?

30. Building

- When compiling our C++ code into a binary object file, the object file has a larger file size than the C++ file. Why?

```
object file is the translation of machine level language,
and sometimes include all the libraries
```

31. Makefiles

- What advantages to the build process does a Makefile provide?

```
1. only recompile files that contain changes
2. Dependencies?
```

- Why do we separate out the compile and link steps of our build process?

```
Compile step: Source files → Object files (.cpp → .o)
Link step: Object files → Executable (.o → final program)
Advantage: Only recompile changed source files, always relin
```