GITHUB LINK: https://github.com/jarenmatthew/CMSC-21/tree/main/1stLongExam%5BOng%5D

Part 1

1. True, because it contains more bytes which can store more values.
2. True, 1 is generally considered as True, but any non-zero number is also True.
3. False, '=' is an assignment operator, and '==' is a binary operator.
4. False, because it is part of the syntax; meaning, C understands or has another use for the keyword.
5. False, address operator (&) is used to tell "scanf" the location in where to store the value inputted.
6. False, because sign qualifiers cannot be applied to float and double.
7. False, because "(a==b) || (b>a)" is True and "True && (d<a)" is True.
8. False, it isn't required if the default case is the last statement.
9. False, the statement becomes true if you replace "&&" with "||".
10. False, it isn't the same if a negative number is the assigned value to 'i'.

Part 2

1. It should be a curly bracket and not a semi colon after the while statement.
```
x = 1;
while (x <= 10){
 ++x;
}
```

2. Since the goal of the program isn't stated, I think that the output of the program should be 0.10000 to 1.000000.
```
for (double y = .1; y <= 1.0; y += .1) {
	printf("%f\n", y);
}
```

3. There is no break statement after case 1. Making the output wrong if "n" is 1.
```
switch (n) {
case 1:
	printf("The number is 1");
	break;
case 2:
	printf ("The number is 2");
	break;
default:
	printf ("The number is not 1 or 2");
	break;

}
```

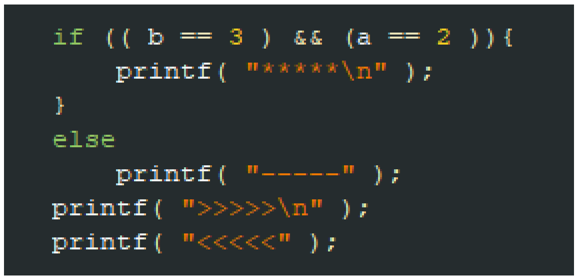
4. The condition in the while loop should be adjusted. Either (n <=10) or (n < 11)
   n = 1;
   while (n <= 10) {
   		printf("%d ", n++);
   }

Part 3

1. There will be no syntax error if a variable isn't declared, but the program might have wrong or different outputs.
2. There will be no value or the return value will be undefined. In the newer versions of C, you may omit the return statement, but some still use it for readability and practice.
3. They differ when used with scanf. %i can be used for inputting hexadecimal or octal numbers; whereas for %d, it won't be the correct value for hexadecimal or octal numbers.
4. a = 10
   b = 5
   c = 0.300000
5. a = 12.300000
   b = 0.6
   c = 45
6.
   a. (a * b) – (c * d) + e
   b. (a / b) % c) / d
   c. (– a) - b + c – (+ d)
   d. ((a * (- b)) / c) – d

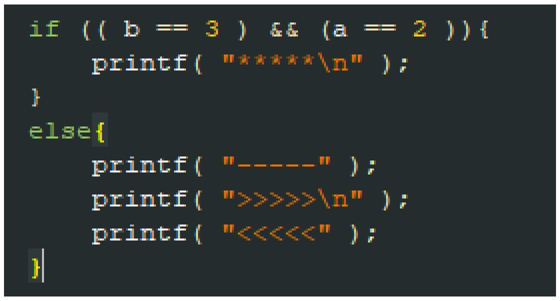
7. for (i = 0; j > 0; j/=2){}

Part 4

8.
   a. "*****>>>>><<<<<". It is very hard to read. We can improve readability by adding indentions or curly brackets for the if else statements.
   b.
      a. 

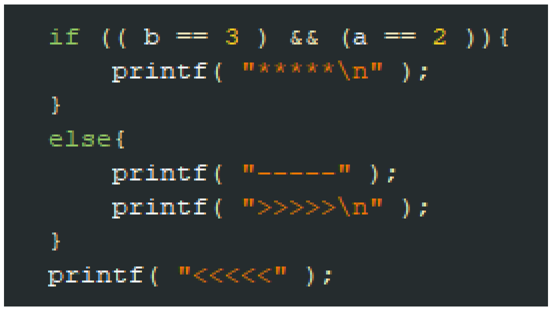
```
if (( b == 3 ) && (a == 2 )){
    printf( "*****\n" );
}
else
    printf( "-----" );
printf( ">>>>>\n" );
printf( "<<<<<" );
```

b.



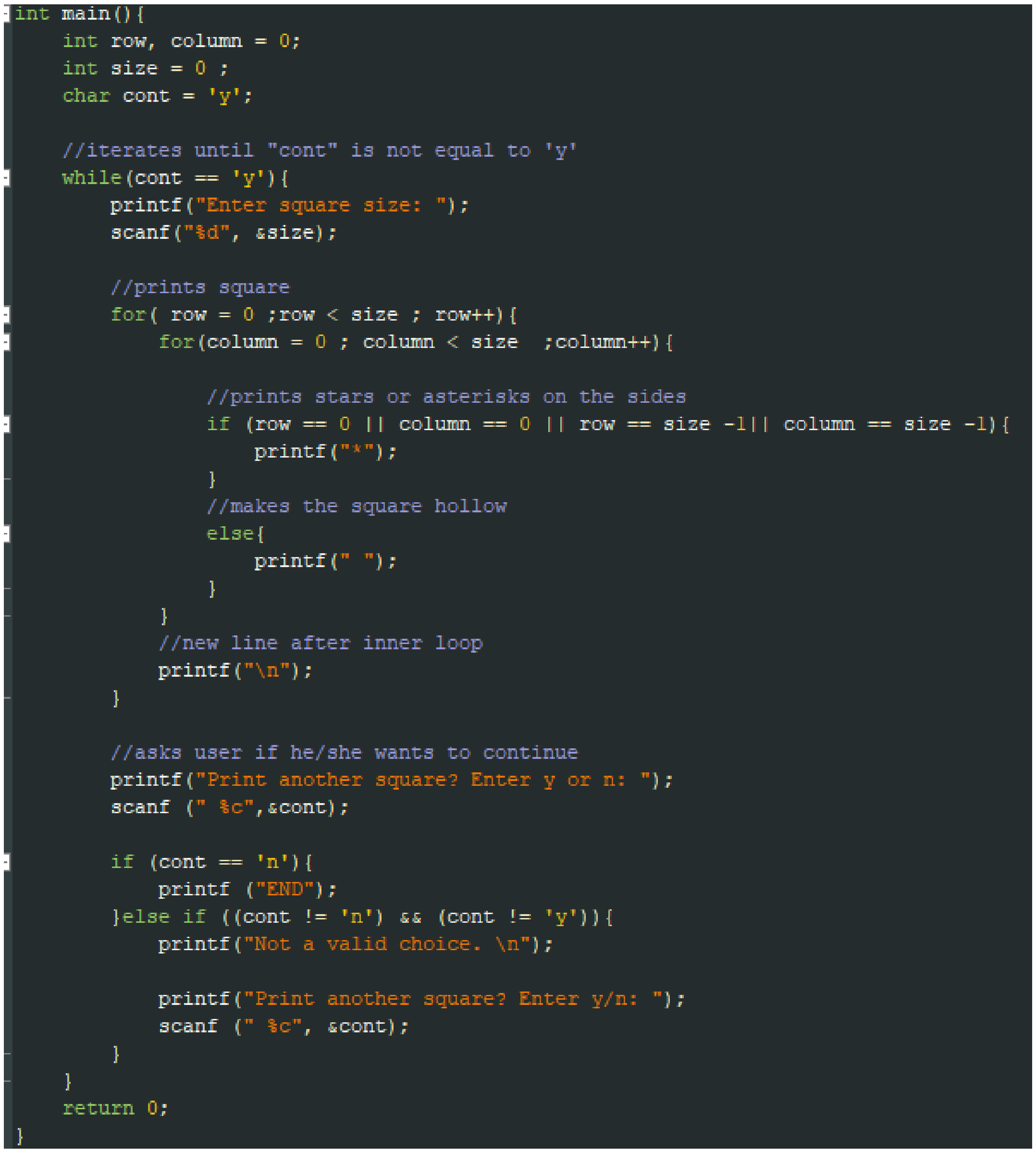
```c
if (( b == 3 ) && (a == 2 )){
    printf( "*****\n" );
}
else{
    printf( "-----" );
    printf( ">>>>>\n" );
    printf( "<<<<<" );
}
```

c.



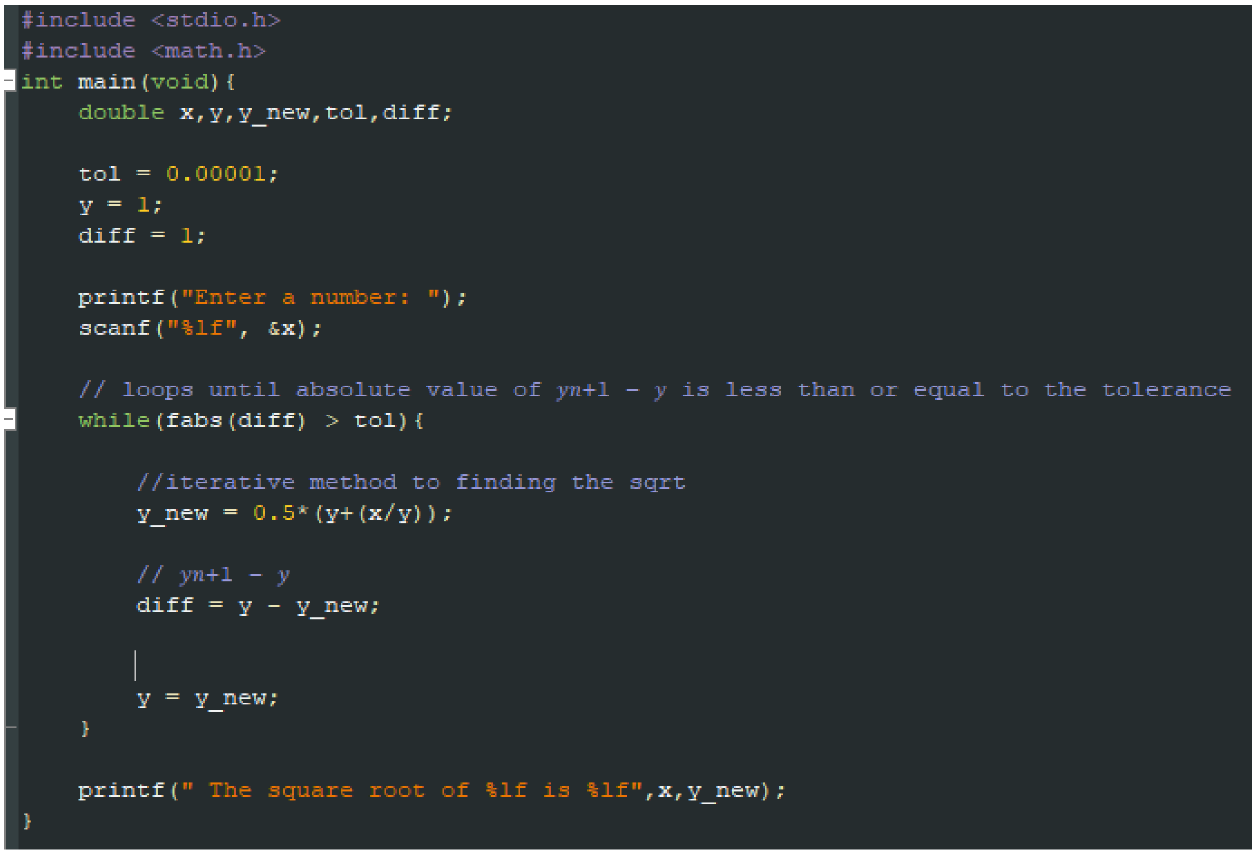
```c
if (( b == 3 ) && (a == 2 )){
    printf( "*****\n" );
}
else{
    printf( "-----" );
    printf( ">>>>>\n" );
}
printf( "<<<<<" );
```

9.



```c
int main(){
    int row, column = 0;
    int size = 0 ;
    char cont = 'y';

    //iterates until "cont" is not equal to 'y'
    while(cont == 'y'){
        printf("Enter square size: ");
        scanf("%d", &size);

        //prints square
        for( row = 0 ;row < size ; row++){
            for(column = 0 ; column < size  ;column++){

                //prints stars or asterisks on the sides
                if (row == 0 || column == 0 || row == size -1|| column == size -1){
                    printf("*");
                }
                //makes the square hollow
                else{
                    printf(" ");
                }
            }
            //new line after inner loop
            printf("\n");
        }

        //asks user if he/she wants to continue
        printf("Print another square? Enter y or n: ");
        scanf (" %c",&cont);

        if (cont == 'n'){
            printf ("END");
        }else if ((cont != 'n') && (cont != 'y')){
            printf("Not a valid choice. \n");

            printf("Print another square? Enter y/n: ");
            scanf (" %c", &cont);
        }
    }
    return 0;
}
```

10.



```
#include <stdio.h>
#include <math.h>
int main(void){
    double x,y,y_new,tol,diff;

    tol = 0.00001;
    y = 1;
    diff = 1;

    printf("Enter a number: ");
    scanf("%lf", &x);

    // loops until absolute value of yn+1 - y is less than or equal to the tolerance
    while(fabs(diff) > tol){

        //iterative method to finding the sqrt
        y_new = 0.5*(y+(x/y));

        // yn+1 - y
        diff = y - y_new;

        y = y_new;
    }

    printf(" The square root of %lf is %lf",x,y_new);
}
```