I.

1. True. It is because long will extend a datatype's range (int , double, float).
2. True. In logic, 0 means FALSE while 1 means TRUE. In programming languages, other value than 0 can be interpreted as TRUE.
3. False. == is used for comparison while for declaring variables, = is used.
4. True. There will be a conflict in the compiler if this happens. It will produce an error.
5. True. When using scanf, '&' is used. For example: scanf("&d", &num);
6. False. Sign qualifiers can only be used on int and char.
7. False. FALSE || (TRUE && TRUE) ➔ FALSE || TRUE ➔ TRUE
8. False. It is not necessary on default because it is the last possibility. However, it is required on all cases except default.
9. False. Both expressions should be true since it is &&.
10. True. If the variable i is initialized, it will print the same output. In the first statement it will keep subtracting until i > 0. It is the same on the second statement because the variable i has a value and it is still accepted in the while expression where it will keep subtracting until it becomes less than 1.

II.

1. Error in Line 1. – Correction: int x = 1;
   Error in Line 2: - Correction: while(x <= 10){
2. Error in Line 1. – Correction: y < 1.0;
   Error in Line 2. – Correction: printf("%lf\n", y);
3. Error in Line 1. – Correction: Initialize n
   Error in Line 4. – Correction: Add break;
   Error in Line 9. – Correction: Remove break;
4. Error in Line 1. – Correction: int n = 1;
   Error in Line 2. while(n < 11) {

III.

1. The compiler will still execute the program; however, the value of the variable is always an undefined value that is stored in the system's memory, which is random. For example, I used following code.

   ```
   int n;
   printf("%d", n + 1);
   ```

   In my execution, it produced 16, which is very random.

2. The program will still run, however, return 0 is a signature for C that indicates that the program ran successfully. It will raise an anomaly in the system but not in the execution of the program.
3. %i is appropriate for integers while %d is for signed decimal integers. If we are just going to print using the predefined value of the same variable (example int i = 5;), there will be no difference.

However, if we ask an input, there will be a different behavior since the former auto detects the base of the number while the latter converts it into base 10 (example i = 013, j = 013; output: i = 11, j = 13).

4. 10 0.300000 5
5. 12.300000 45 0.6
6. A. ((a * b) – (c * d)) + e
   B. ((a / b) % c) / d
   C. ((- a - b) + c) - + d
   D. (a * - (b/c)) - d
7. for (int j = 5; j > 0; j /=2)

IV.

8. A. *****>>>>><<<<<
   - Since there are multiple if-else statements, there should be braces. Also, lines 1 and 2 can be joined with a logical operator.

   B.

```
const char NEWLINE = '\n';
int a = 2;
int b = 3;


if (a != 2 && b != 3){
    printf( "*****" );
}
else{
    printf( "*****" );
    printf("%c", NEWLINE);
    printf( ">>>>>" );
    printf("%c", NEWLINE);
    printf( "<<<<<" );
}
```

   C.

```
const char NEWLINE = '\n';
int a = 2;
int b = 3;


if (a == 2 && b == 3){
    printf( "*****" );
}
```

```
else{
    printf( "*****" );
    printf("%c", NEWLINE);
    printf( ">>>>>" );
    printf("%c", NEWLINE);
    printf( "<<<<<" );
}
```

D.

```
const char NEWLINE = '\n';
int a = 2;
int b = 3;


if (a == 2 || b == 3){

    printf( "*****");
    printf("%c", NEWLINE);
    printf( "<<<<<" );
}
else{
    printf( "*****" );
    printf("%c", NEWLINE);
    printf( ">>>>>" );
    printf("%c", NEWLINE);
    printf( "<<<<<" );
}
```

9. A. cont = ‘y’
   B. %d
   C. row++;
   D. column < size;
   E. row == size - 1
   F. row == 0
   G. column == size - 1
   H. column == 0
   9. printf(“\n”);
   10. scanf(“ %c”, &cont);
   11. break;
   12. cont != ‘y’
   13. scanf(“ %c”, &cont);

10.

```c
#include <stdio.h>
#include <math.h>

int main(void){

    int x;
    double y = 1.0;
    float tolerance = 0.00001;
    const char NEWLINE = '\n';
    const char TAB = '\t';

    printf("Enter x: ");
    scanf("%d", &x);

    double x_over_y = x/y;
    double equation = (y + x_over_y)/2;

    printf("x %c y %c         x/y %c         1/2(y + (x/y)%c", TAB, TAB, TAB, NEWLINE);
    printf("%d %c %lf %c %lf %c %lf %c", x, TAB, y, TAB, x_over_y, TAB, equation, NEWLINE);

    while (fabs(equation - y) >= tolerance){
        y = fabs(equation);
        x_over_y = x/y;
        equation = (y + x_over_y)/2;


        printf("%d %c %lf %c %lf %c %lf %c", x, TAB, y, TAB, x_over_y, TAB, equation, NEWLINE);
    }

    printf("%cThe square root of %d is: %lf", NEWLINE, x, equation);

    return 0;
}
```

Github Link:

https://github.com/dreeew05/CMSC21/tree/master/FirstLongExam

PS. Nasa code sa github po ang comments. Thanks!