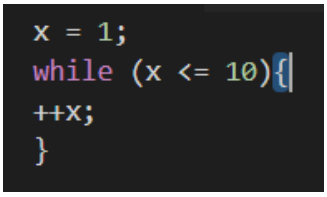
Part I. True of False. Justify your answer.

1. A long double can be used if range of a double is not enough to accommodate a real number.
2. A zero value is considered false and a non-zero value is considered true.
3. = is used for comparison whereas == is used for assignment of two quantities.
4. The keywords cannot be used as variable names.
5. The address operator (&) is used to tell the compiler to store and data at an address.
6. Sign qualifiers can be applied to all data types.
7. Suppose you declared a variable a = 6, b = 7, d = 3, (((a==b) || (b>a)) && (d<a)) will generate a false.
8. The break statement is required in the default case of a switch selection statement
9. The expression (x > y && a < b) is true if either x > y is true or a < b is true.
10. while(i > 0) printf("T minus %d \n", i--) will produce a similar output with while (i) printf("T minus %d \n", i--)

Answers:

1. True. A long double has higher bytes than double.
2. True. Zero is used to represent false, while 1 a non-zero value is for representing true
3. False. == is used for comparison and = is used to assign a value.
4. True. Keywords cannot be used as it would execute the intended use of the word instead of becoming a variable
5. True. & is used in scanf to store values in a variable
6. False. It can only be used in characters or integer data types
7. False. F or T and T will generate a true
8. False. It is not needed as it is the last option in the switch selection statement.
9. False. Both needs to be true in order for the expression to be true.
10. True. Both would result in the same output.

Part II. Find the errors in the following program. Indicate the possible correction.

1. 
```
x = 1;
while (x <= 10){
++x;
}
```
- "{" is added

2.
```c
for (double y = 0.1; y != 1.0; y += 0.1) {  // .1 to 0.1
 printf("%f\n", y);
}
```
3.
```c
switch (n) {
case 1:
 printf("The number is 1");
 break; //adding break
case 2:
 printf ("The number is 2");
 break;
default:
 printf ("The number is not 1 or 2");
 break;
}
```
4.
```c
n = 1;
    while (n <= 10) { // instead of <10 we use <=10
        printf("%d ", n++);

    }
}
```

Part III. Answer the following questions.

1. – the uninitialized variable's value can be anything, but after trying it out, it's value just becomes zero "0" .

2. Same with what happens with uninitialized variables, function becomes undefined. I noticed that even when the return statement is not in the code, the program exits itself just fine.

3. %i stands for integer while %d for decimal integer, they don't have any difference when used in printf but has in scanf. The way they differ is that %d assumes that the input is base 10 while %i checks for decimal, octal or hexadecimal values.

4. Values would be:
   a = 10
   b = 5
   c = 0.300000

5. Values would be:
   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; i++, j /= 2)

Part IV. Coding Applications

8. a. There are two if statement, when you can put them in one.

b. a.

```c
#include <stdio.h>

int main() {
    int a,b;
    a = 2;
    b =3;

    if ( b == 3 && a == 2){ // joining the two if statement together
        // adding \n to space it up and make it cleaner
        printf( "*****\n" );
        printf( ">>>>>\n" );
        printf( "<<<<<\n" );
    }
    else {
        printf( "-----" );
    }
}
```

b.

```c
#include <stdio.h>

int main() {
    int a,b;
    a = 2;
    b =3;

    if ( b == 3 && a == 2){ // joining the two if statement together
    // adding \n to space it up and make it cleaner
        printf( "*****\n" );
    }
    else {
        printf( ">>>>>\n" );
        printf( "<<<<<\n" );
        printf( "-----" );
    }
}
```

c.

```c
#include <stdio.h>

int main() {
    int a,b;
    a = 2;
    b =3;

    if ( b == 3 && a == 2){ // joining the two if statement together
    // adding \n to space it up and make it cleaner
        printf( "*****\n" );
        printf( "<<<<<\n" );
    }
    else {
        printf( ">>>>>\n" );
        printf( "-----" );

       }
}
```

9.

```c
#include <stdio.h>
int main(){
    int row, column = 0;
    int size = 0 ;
    char cont = 'y';

    while(cont == 'y'){ //a
        printf("Enter square size:");
        scanf("%d", &size); //b

        for( row = 0 ;row <= size ; row++){  //c    , also tweaked row < size to row <= size
            for(column = 0 ; column <= size; column++){ //d
                if (row == 0 || row == size|| column == 0|| column == size){ // e f g h
                    printf("*");
                }
                else{
                    printf(" ");
                }
            }
            printf ("\n"); // 9
        }

        printf("Print another square? Enter y or n: ");
        scanf ("%c", &cont); //10

        if (cont == 'n'){
            printf ("END"); //11
        }
        else if (cont == 'x'){  //12
            printf("Not a valid choice. \n");
            printf("Print another square? Enter y/n: ");
            scanf ("%c",&cont); //13
            }
    }
    return 0;
}
```

10.

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

int main(){
    // Variables
            // sets variables for brackets of operands , absval for absolute value
    int x, i;
    double second_set, absval;
    float y = 1;
    float first_set, tol;

    tol = 0.00001;

    // Input
    printf ("Please Enter a Value: ");
    scanf ("%d", &x);

    // Loop
    for ( i = 1; absval <= tol; ++i){
        for( y = 1 ; absval <= tol ; y =  1 / 2 * first_set){
            first_set = y + x / y;
            second_set = y - 1;
            absval = fabs(second_set);
        }
    }
     printf ("%f", y);

     return 0;
}
```

Reference:

IV. no.9

Delgado, C. (2019, April 24). *How to print a hollow square/box/rectangle pattern with asterisks or a custom character in the C language*. Our Code World. Geraadpleegd op 22 April 2022, van https://ourcodeworld.com/articles/read/919/how-to-print-a-hollow-square-box-rectangle-pattern-with-asterisks-or-a-custom-character-in-the-c-language

Delgado, C. (2019, March 15). *How to print a square pattern with asterisks or a custom character in the C language*. Our Code World. Geraadpleegd op 22 April 2022, van https://ourcodeworld.com/articles/read/855/how-to-print-a-square-pattern-with-asterisks-or-a-custom-character-in-the-c-language