Part I. True of False. Justify your answer.

1. True, because a long double is equivalent to 8 bytes, which is larger than a double.

2. In computer language which takes binary digits 1 and 0, it takes the former as true and the latter as false.

3. False, == is an equality operator used to compare two quantities, while = is an assignment operator where we assign a value to a variable.

4. True, the keywords have a fixed function and meaning, and using it as a variable will only try to change it so it will be invalid.

5. True, when we ask for a user input and we scan it, we use the address operator to tell the compiler to store the user input into a variable for it.

6. True, it can be used for both integers and characters type of data, whether they are signed or unsigned.

7. True, because it will not satisfy the first condition, since a is not equal to b even though the second and last conditions are true. But because the first condition is false, it will generate a false because it has the || operator, meaning that all conditions should be true so the general result will be true.

8. False, it is the very last statement of the switch selection statement, so we can omit it.

9. False, using the && logical operator means that both conditional statements should be true so that the whole statement will be true.

10. True, it will both produce an infinite loop since there is no break statement and there is no limiting condition

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

1. Instead of ++x, x++

2. Variable y is a float, not a double

3. Case 1 doesn't have a break command or statement

4. n < 10 should be n <= 10

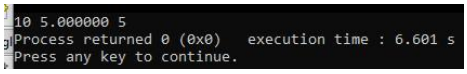
**Part III. Answer the following questions.**

1. The program will produce inconsistent results, and sometimes will not run at all. The program may look like it will work correctly, and it did, but later on won't work correctly.

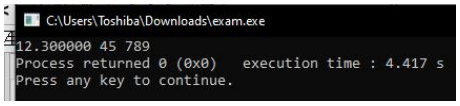
2. Program will return to the main function after the end of the execution, and meaning the return value is undefined, nothing appears on the console.

3. %d assumes base 10 of integer values and takes it as signed decimal integer while %I detects the base of an integer as an integer value, with decimal, hexadecimal and octal data.

4. a and b values will be integers and c will be a float




5. The value of a will be a float and values b and c as decimals




6.

```
a. (a * b) - ((c * d) + e)
b. (a / b) % (c / d)
c. ((- a) - (b + c)) - (+ d)
d. ((a * (- b)) / c) - d
```

7.

```
1  #include <stdio.h>
2
3  int main(){
4
5      int i, j;
6
7      for (i = 0; j > 0; ++i)
8      j /= 2;
9      printf("%d", j);
10 }
11
```

Part IV. Coding Applications

8.

b. Modify the code such that it produces the following outputs (a = 2 and b = 3)

a. *****

>>>>>

<<<<<

```c
#include <stdio.h>

int main(){

    int a,b; // specifying variables as integers

    a = 2; // assigning variables into values
    b = 3;

    if ( b == 3 && a == 2 ) {  //condition for the if statement
        printf( "*****" ); // print statement if if statement is satisfied
        printf( ">>>>>" );
        printf( "<<<<<" );
    }
    else { //else statement if not satisfied
        printf( "-----" );
    }
}
```

```
*****
>>>>>
<<<<<
Process returned 0 (0x0)   execution time : 6.092 s
Press any key to continue.
```

b. *****

```c
#include <stdio.h>

int main(){

    int a,b; // specifying variables as integers

    a = 2; // assigning variables into values
    b = 3;

    if ( b == 3 && a == 2 ) {  //condition for the if statement
        printf( "******" ); // print statement if if statement is satisfied
    }
    else { //else statement if not satisfied
        printf( "-----" );
        printf( ">>>>>" );
        printf( "<<<<<" );
    }
}
```

```
*****
Process returned 0 (0x0)   execution time : 4.757 s
Press any key to continue.
```

c. *****
<<<<<

```c
#include <stdio.h>

int main(){

    int a,b; // specifying variables as integers

    a = 2; // assigning variables into values
    b = 3;

    if ( b == 3 && a == 2 ) {  //condition for the if statement
        printf( "*****" ); // print statement if if statement is satisfied
        printf( "<<<<<" );
    }
    else { //else statement if not satisfied
        printf( "-----" );
        printf( ">>>>>" );

    }
}
```

```
*****
<<<<<
Process returned 0 (0x0)   execution time : 4.608 s
Press any key to continue.
```

9.

A. (cont == 'y')
B. %d
C. row++
D. column < size
E. i = 0
F. j = 0
G. i = size
H. j = size

9. printf (“\n”);
10. scanf (“%c”, &cont);
11. printf(“You have exited the program, thank you for using!”);
12. (cont != ‘y’ && cont != ‘n’)
13. scanf (“%c”, &cont);

```c
#include <stdio.h>

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

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

    for( row = 0 ;row < size ; row++){
        for(column = 0 ; column < size;column++){
            if (i = 0 || j = 0|| i = size|| j = size){
                printf("*");
        }
        else{
            printf(" ");
        }
    }
    printf("\n");
}

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

    if (cont == 'n'){
        printf("You have exited the program, thank you for using!");
    }
    else if (cont != 'y' && cont != 'n'){
        printf("Not a valid choice. \n");

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

10.

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

int main(void){

    double tol, x, y, yn;  //declaring variables as doubles

    tol = 0.00001;   //assigning the tolerance value to the same variable name

    printf("Enter a number: "); //asking input from the user
    scanf("%lf", &x); //scanning and storing the user input into variable x

    y = 1;        //assigning the initial guess 1 into y
    yn = 0.5*(y+(x/y));  //assigning formula for the next guess into variable yn

    while(fabs(yn - y) > tol ){ //iteration that will repeat until the next guess is less than or equal to the tolerance value
        y = yn;       //updating the value of y to the yn + 1 value (next guess that will the previous guess for the next one)

        yn = 0.5*(y+(x/y));  //solving the next guess
    }

    printf(" The approximate square root of %.2lf is %lf", x, y);   //printing and displaying the output

    return 0;
}
```