Name: Carl Jorenz S. Gimeno

Part I.

1. True
2. True
3. False
4. True
5. True
6. False
7. False
8. False
9. False
10. True

Part II.

1.
```
x = 1;
while (x <=10){x++;} //Add opening brace
```

2.
```
for (int y = 1; y != 10; y +=1){
        printf("%f\n",y/10.0); //Reconstruct the whole loop to use int instead of double
        }
```

3.
```
switch (n) {
case 1:
        printf("The number is 1");
        break; //Add a break
case 2:
        printf ("The number is 2");
        break;
default:
        printf ("The number is not 1 or 2");
        break;
}
```

4.
```
int n = 1; #Add data type
while (n <= 10) { //Include 10 in condition
        printf("%d ", n++);
}
```

Part III.

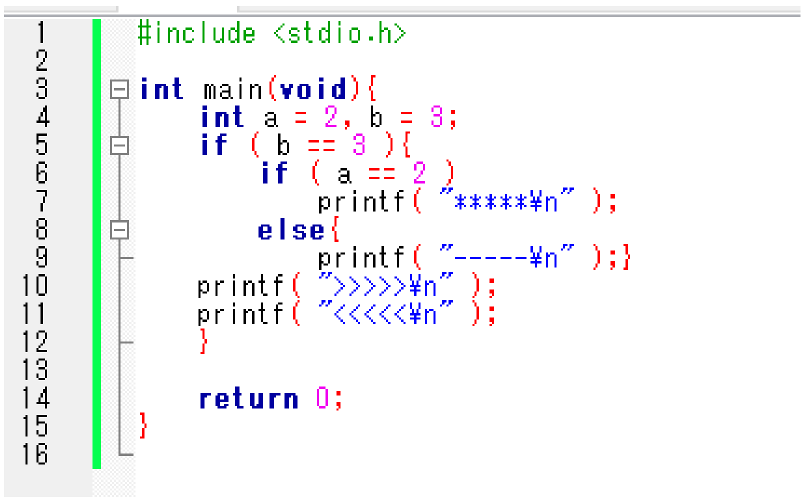
1. Accessing the memory address of an uninitialized variable results getting the garbage or leftover data in that specific address and using it as value.
2. The main function uses the return statement to determine the status of the program execution. Not having a return statement will not do anything but at the same time, will not show if the program has completed successfully or not as it will default to 0.
3. %i specifies integers while %d specifies signed decimals.
4. a = 10, b = 5, c = 0.300000

https://github.com/CarlJorenzGimeno/CMSC21

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 (j ; j > 0 ; j/=2)

Part IV

8.
    a. The output is *****>>>>><<<<<. The lack of parentheses and indentation can easily confuse the programmer on how conditional statements are nested.
    b. Modify the code such that it produces the following outputs (a = 2 and b = 3)
        i.

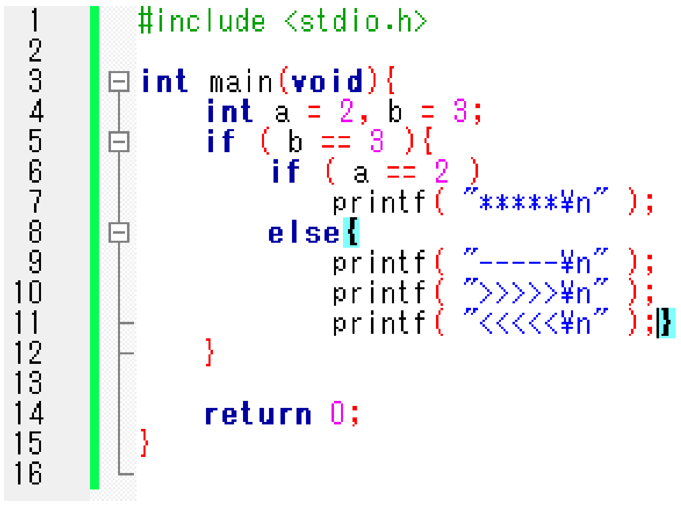


```
#include <stdio.h>

int main(void){
    int a = 2, b = 3;
    if ( b == 3 ){
        if ( a == 2 )
            printf( "*****¥n" );
        else{
            printf( "-----¥n" );}
    printf( ">>>>>¥n" );
    printf( "<<<<<¥n" );
    }

    return 0;
}
```

        ii.

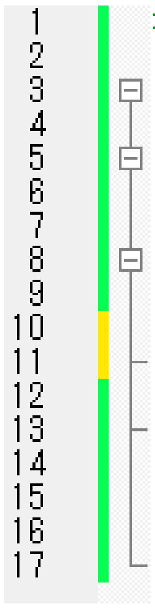


```
#include <stdio.h>

int main(void){
    int a = 2, b = 3;
    if ( b == 3 ){
        if ( a == 2 )
            printf( "*****¥n" );
        else{
            printf( "-----¥n" );
            printf( ">>>>>¥n" );
            printf( "<<<<<¥n" );}
    }

    return 0;
}
```

iii.

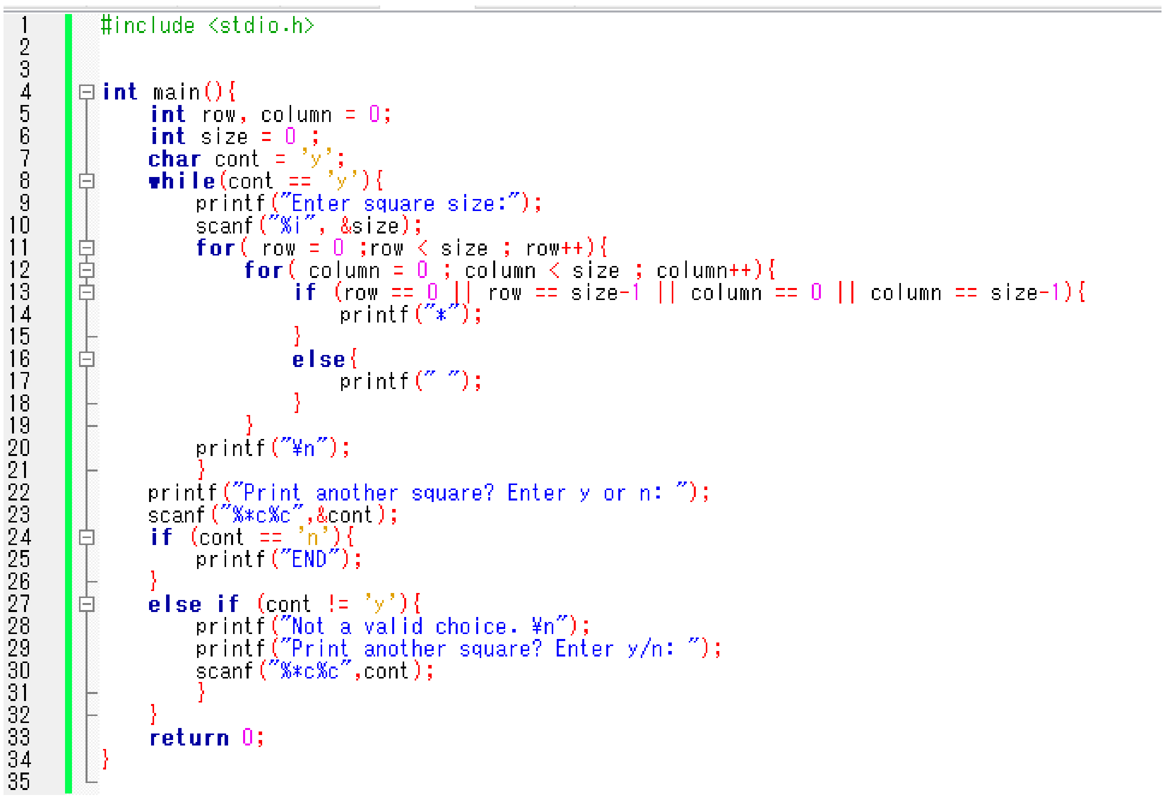


```c
#include <stdio.h>

int main(void){
	int a = 2, b = 3;
	if ( b == 3 ){
		if ( a == 2 )
			printf( "*****¥n" );
		else{
			printf( "-----¥n" );
			printf( ">>>>>¥n" );
			}
	printf( "<<<<<¥n" );
	}

	return 0;
}
```

9.

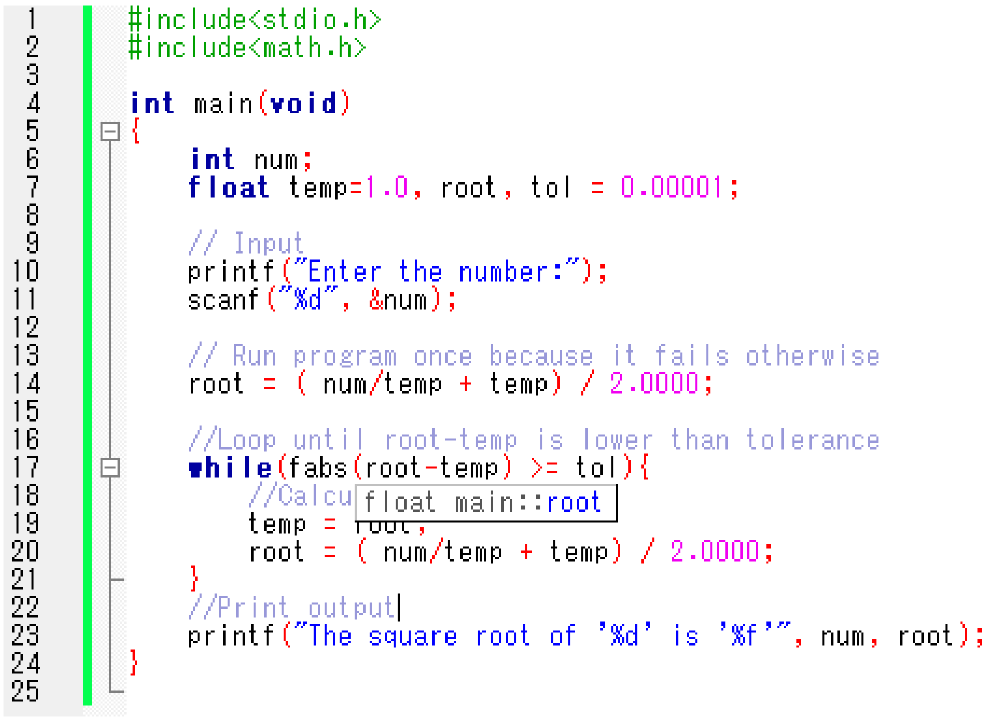


```c
#include <stdio.h>

int main(){
	int row, column = 0;
	int size = 0 ;
	char cont = 'y';
	while(cont == 'y'){
		printf("Enter square size:");
		scanf("%i", &size);
		for( row = 0 ;row < size ; row++){
			for( column = 0 ; column < size ; column++){
				if (row == 0 || row == size-1 || column == 0 || column == size-1){
					printf("*");
				}
				else{
					printf(" ");
				}
			}
		printf("¥n");
		}
	printf("Print another square? Enter y or n: ");
	scanf("%*c%c",&cont);
	if (cont == 'n'){
		printf("END");
	}
	else if (cont != 'y'){
		printf("Not a valid choice. ¥n");
		printf("Print another square? Enter y/n: ");
		scanf("%*c%c",cont);
		}
	}
	return 0;
}
```

https://github.com/CarlJorenzGimeno/CMSC21

10.



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

int main(void)
{
    int num;
    float temp=1.0, root, tol = 0.00001;

    // Input
    printf("Enter the number:");
    scanf("%d", &num);

    // Run program once because it fails otherwise
    root = ( num/temp + temp) / 2.0000;

    //Loop until root-temp is lower than tolerance
    while(fabs(root-temp) >= tol){
        //Calcu
        temp = root;
        root = ( num/temp + temp) / 2.0000;
    }
    //Print output
    printf("The square root of '%d' is '%f'", num, root);
}
```