I

1. True
2. True
3. False
4. False
5. False
6. False
7. True

Number 8 is the illegal one.

8. False
9. True
10. True
11. True

II

1. For example, in 2d dimensional array the first dimension represents the row while the second one represents the columns, now without the second dimension or the column, the compiler will not know the number of elements in each row because it doesn't know the offset of the number of columns so that's why specifying the number of columns of the array is important and mandatory.
2.

Declare: bool isPalindrome(char *string);

a.
```
bool isPalindrome(char *string)
{
        //code;
}
```

Declare: float computeAverage(float arr[20]);

b.
```
float computeAverage(float arr[20])
{
        //code;
}
```

Declare: void reverseSentence(void);

c.
```
 void reverseSentence(void)
{
        //code;
{
```

Declare: float squareRoot(int num);

d. float squareRoot(int num)
{
//code;
}

3.

a. Nested functions are not allowed in C, so enable to fix the problem you have to separate the functions into one to prevent the two functions from merging in one function. Plus, the quotation mark doesn't fit well inside the printf function you must use only a pair of quotation mark when printing strings.
b. The error there is it will not return anything if you use that function, you must put a printf function then also put a return result there to display the product.
c. In the argument you already declare the variable a into a float so you don't have to declare the variable a again as a float inside the function, plus there's a semicolon after the function that must be removed because you only put a semicolon on a function on the part where you are declaring it.
d. Using a void function means you don't have to return anything when you use that, that's the error on that code because you use void function therefore you don't have to return anything in your code.

4.

a.

```c
#include <stdio.h>
#define SIZE 5 //a

int main()
{
    int numbers[SIZE] = {1, 2, 3, 4, 5};//a.
```

b.

```c
    int *ptr; //b
```

c.

```c
ptr = numbers; //c.
```

d.

```c
    printf("Using pointer/offset notation with the pointer ptr \n");
    for (int i = 0; i<SIZE; i++)
    {
        printf("%d\n", i, *(ptr + i));
    }
```

e.

```
    printf("\n");
    printf("Using the array name as the pointer \n");
    for (int i = 0; i<SIZE; i++)
    {
        printf("%d\n", i, *&numbers[i]);
    }
```

f.

```
    printf("\n(f.1) array index notation: %d", numbers[2]);
    printf("\n(f.2) pointer notation with array name as the pointer: %d",
   *&numbers[2]);
    printf("\n(f.3) pointer index notation with ptr: %d", ptr[2]);
    printf("\n(f.4) pointer notation with ptr: %d", *ptr + 2);
```

g. Assume that the address of the first element of the *ptr is 2500, then if you will change it to *ptr + 2 then I must add 2 times 4, why 4? Because it is 4 bytes.

   2500 + (4*2) = 2508 is the address of *ptr + 2.

5.

a. Not an error.
b. Error, the correct way should be num = *xp
c. Error, invalid type of argument of unary * it should be num = xp[1].
d. Error, because the array cannot be incremented.

III.

Coding Part:

https://github.com/JohnLuis07/john/tree/main/CMSC%2021/Long%20Exam%202