I. True or False

1. True

2. True

3. False

4. False

5. False

6. False

7. False

----------True = illegal, False = not----------

8. False

9. True

10. True

11. True

II. Provide Answers

1. The first dimension can be filled automatically if there is an initializer, the rest of the dimensions will be needed to be filled as it would determine what multidimensional array you are making.

2.

```
a. bool isPalindrome (char *string);

b. float computeAverage (float arr[20]);

c. void reverseSentence ();

d. float squareRoot (int num);
```

3.

a. Problems = function inside a function, " symbol to the right of %s,and %s

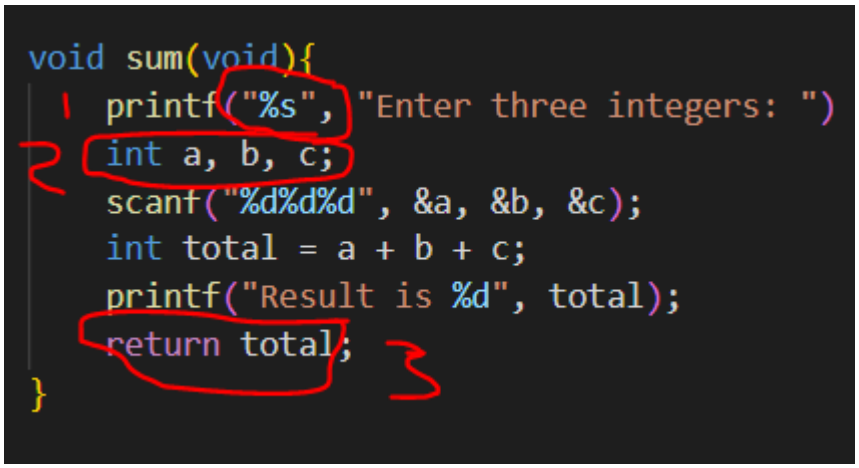
As nested functions is not possible in C, you should separate them into two functions. After that the " symbol besides the %s confuses the group of words as a variable and also errors such that \n should be inside " ". removing the one to the right of %s should fix it. The %s should be removed, as there is no variable to base from, if there is one it should be changed to %d or %i as the function returns int value type.

b. No Error

c. Problems = void fun (float a); ,and float a;

There is a ; symbol besides the void (float a), remove that so the function will be able to function. For the float a; , it should also be removed, the data type has already been declared and doesn't need to be declared once again.

d.



1 – should be removed as it is not needed

Case 1: if function
2 – instead of declaring them there, we should put them inside (void) as 3 variables to be taken from)

3 – we can just disregard and remove this

Case 2: make the sum into main

2 – bring it above the printf ("Enter three Integers: ");

3- change to return 0;

```c
// Solution 1
void sum(int a,int b,int c){
    printf("Enter three integers:");
    scanf("%d%d%d", &a, &b, &c);

    int total = a + b + c;

    printf("Result is %d", total);
}
// Solution 2
void main(void){
    int a, b, c;
    printf("Enter three integers:");
    scanf("%d%d%d", &a, &b, &c);
    int total = a + b + c;
    printf("Result is %d", total);
    return 0;
}
```

4.

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

b. int *ptr;

c. ptr = &numbers[0]

d. printf("%d \n",*ptr);

e. printf("%d \n",*numbers);

f. 1. numbers[1]
   2. *(numbers+1)
   3. *(ptr+1)
   4. *(ptr+1)

g. Address of the third element. Value is 3
```

5.

```
a. ++xp;
 --> none

b. num = xp; //use pointer to access first element (assume xp is initialized)
 --> lacks the sign * to refer to the array

c. num = *xp[1]; //assign element 1 (value 2) to num
 --> you cannot use [] as xp is not an array, instead use *(xp+1) to cycle to element 1

d. ++x;
 --> add a [n] to specify what value from the array you will add to; n < size of array
```

III. Code