GITHUB LINK: https://github.com/jarenmatthew/CMSC-21/tree/main/2nd%20Long%20Exam

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

II. 1. Since an array is stored in one single line, not by column and row, if the 2nd dimension is left unspecified the computer or language will be unable to tell how many values there are in the 1st dimension. If the 1st dimension is left unspecified but there's a value in the 2nd dimension, the computer will understand that this number of values will end here then it will go to the next index.

2. a.
```
bool isPalindrome(char *p){
    . . .
      return 0;
    . . .
      return 1;
}
```

b.
```
float computeAverage(float array[20]){
    . . .
    return average;
}
```

c. void reverseSentence (void){...}

d.
```
float squareRoot(int num){
    . . .
    return square_root;
}
```

3.

a. You cannot have nested functions. So to fix the error. You can just create 2 separate functions one of which should be a nested function or in another functions. After creating 2 separate functions, you can simply call the other function in the other function.

Like this,

```
int fun(void){
    printf("%s", Inside function fun\n");
    bored();
}

int bored(void){
    printf("%s", Inside function bored\n");
}
```

Also, another error could be seen in the prinf function because there is a lacking quotation mark. So we can just add another quotation mark.

Lastly, since both functions return nothing, though no error is being shown, it is better to just make the function void. For example, void fun(void) or void bored(void)

b. The code in letter b is fine but not fully completed. It takes in 2 integer values and is probably supposed to return or print the product. So we can fix or add to the function “return result;” or “printf(“%d”,result);” at the end of the code.

c. The semicolon after the function name “void fun(float a); {…}” should be omitted. Making it, “void fun(float a) {…}”. Also, we don’t need to initialize or declare a as a float again since it is already declared in the parameters, so we can simply remove that line.

d. If the return type of the function is void, the function won’t return anything. If you want the function to return something you have to change the return type of functions to something like ‘int sum(void){…}’.
But if you don’t want to return any value, you can leave it as it is and remove the last line which is “return total;” since it is purposeless. But the function given is still going to work as it is.
Lastly, a semicolon should be added on the line “printf(“%s”…)”.

4. a. #define SIZE 5
int array[SIZE] = {1,2,3,4,5};

b. int *ptr;

c. ptr = &array[0]

d. for (int i = 0; i < SIZE; i++){
    printf("%d ", *ptr++);
  }

e. for (int i = 0; i < SIZE; i++){
    printf("%d ", *array + i);
  }

f. f.1 number[2]

f.2 *&number[2]

f.3 ptr[2]

f.4 *(ptr + 1)

g. The address ptr+2 is referenced to is the 2nd index of the array (array[2]), and the value stored there is 3.

5. a. No error, if it is already assigned to array x.

b. num = *xp should be the case

c. “xp” is already a pointer referencing to array x, so there shouldn’t be an asterisk.

d. Error, because you can’t add to an array or add an array.

PART 3

III. 1.

```c
#include <stdio.h>
#include <ctype.h> /* toupper, isalpha */
#include <stdbool.h> /* bool */

// declares functions
void scan_word(int occurrences[26]);
bool is_anagram(int occurrences1[26], int occurrences2[26]);


int main(void) {

    // arrays that keep track of the value of each letter from the 1st and 2nd word
    int first_word[26] = {0}, second_word[26] = {0};

    printf("Enter first word: ");
    // calls function scan_word
    scan_word(first_word);
    printf("Enter second word: ");
    scan_word(second_word);

    // calls function is_anagram
    if (is_anagram(first_word,second_word)) {
        printf("The words are anagrams.\n");
        return 0;
    }
    printf("The words are not anagrams.\n");
    return 0;
}

void scan_word(int occurrences[26]){
    char c;

    while ((c = getchar()) != '\n'){
        // adds value to the array/s
        // if the letter inputed is C.  'C' - 'A' = 2
        // then the value at occurences[2] will be added
        if (isalpha(c)){
            occurrences[toupper(c) - 'A']++;
        }
    }
}

bool is_anagram(int occurrences1[26], int occurrences2[26]){

    for (int i = 0; i < 26; i++){

        // checks if both arrays have the same values throughout all its elements
        // returns false if values are not equal
        if (occurrences1[i] != occurrences2[i]){
            return false;
        }
    }
    return true;
}
```

2.

```c
#include <stdio.h>
#include <ctype.h> /* toupper, isalpha */
#include <stdbool.h> /* bool */

// declares functions
void scan_word(int occurrences[26]);
bool is_anagram(int occurrences1[26], int occurrences2[26]);


int main(void) {

    // arrays that keep track of the value of each letter from the 1st and 2nd word
    int first_word[26] = {0}, second_word[26] = {0};

    printf("Enter first word: ");
    // calls function scan_word
    scan_word(first_word);
    printf("Enter second word: ");
    scan_word(second_word);

    // calls function is_anagram
    if (is_anagram(first_word,second_word)) {
        printf("The words are anagrams.\n");
        return 0;
    }
    printf("The words are not anagrams.\n");
    return 0;
}

void scan_word(int occurrences[26]){
    char c;

    // points to base address of array
    int *p = occurrences;

    while ((c = getchar()) != '\n'){
        // adds value to the array/s
        // if the letter inputed is C.  'C' - 'A' = 2
        // then the value at p[2] will be added

        if (isalpha(c)){
            // p == &occurrences[...]
            p[toupper(c) - 'A']++;
        }
    }
}

bool is_anagram(int occurrences1[26], int occurrences2[26]){

    // points to base addresses of arrays
    int *p1 = occurrences1, *p2 = occurrences2;

    for (int i = 0; i < 26; i++){

        // checks if both arrays have the same values throughout all its elements
        // returns false if values are not equal
        if (p1[i] != p2[i]){
            return false;
        }
    }
    return true;
}
```