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. It is okay to leave the first dimension of an array parameter unspecified but not the other dimensions, because the first dimension does not represent individual elements inside the array. The second dimension, third dimension, and so on is to be specified since these dimensions are representative of the individual elements inside the array and is used by the compiler to be able to locate these individual elements of the array.
2. A. bool isPalindrome(char *string);
   B. float computerAverage(float arr[20]);
   C. void reverseSentence(void);
   D. float squareRoot(int num);
3. A. **You cannot have nested functions in C.**

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

}
```

   B. **I don't see any errors in the snippet aside from missing the return statement (which is negligible).**

```
int product(int a, int b){

        int result = a*b;

        return result;

}
```

   C. **You cannot redeclare an argument. There is also no need to put a semicolon after declaring the function.**

```
void fun(float a){
        printf("%f", a);
}
```

D. **There is no need to put %s when printing a string. You can just directly print the string in the statement. Also, you cannot return a value when your return type of the function is void. Lastly, there are missing semicolons.**

```
void sum(void){
        printf("Enter the three integers: ");
        int a, b, c;
        scanf("%d%d%d", &a, &b, &c);
        int total = a + b + c;
        printf("Result is %d", total);
}
```

4. A. `int numbers[SIZE] = { 1, 2, 3, 4, 5 };`
   B. `int *ptr;`
   C. `ptr = numbers;`
   D.
```
for(int i=0; i<5; i++){
        printf("%d", *(ptr+i));
}
```

E.
```
for(int i=0; i<5; i++){
        printf("%d", *(numbers+i));
}
```

F. 1. numbers[1]

2. *(numbers+1)

3. ptr[1]

4. *(ptr+1)

G. The address of ptr+2 is the address of the third element in the numbers array. The value is 3.

5. A. **No error in this line.**
   B. **num = *xp;** instead of num = xp;
   C. **No need to put an asterisk sign to xp since you are using an array notation.**
   D. **You cannot use an increment operand before an array name.**

III.

1.

```c
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>

//Functions cannot return an array so I used a pointer here
int *scan_word(int occurences[26]){
    char c;
    //Gets an input from the user until a space is detected
    while((c = getchar()) != '\n'){
        //checks if the input are letters
        if(isalpha(c)){
            //Increments the value of the index located by subtracting the position of the letter minus the position of 'A'
            occurences[toupper(c) - 'A']++;
        }
    }
    //returns the address of the array occurences
    return occurences;
}

//Function for checking if anagram
bool isAnagram(int occurences1[26],int occurences2[26]){
    //same variable is used as an indicator
    int same = 1;
    //goes through every element of occurences1 array and compares it with every element of occureces2 array
    for (int i = 0; i < 26; i++){
        //if an element is not the same, the value of same is changed to 0
        if (occurences1[i] != occurences2[i]){
            same = 0;
            break;
        }
    }
    //returns a bool type value (0 or 1)
    return same;
}

int main(void){
    //declaration of variables and arrays
    int *fWord, *sWord, letters[26] = {0};
    int occurences1[26], occurences2[26];

    //Asks the user the first word but the process of storing input is presented in the function scan_word
    printf("Enter the first word: ");
    //calls the function scan_word while passing the letters array as the argument and stores the returned value onto fWord
    fWord = scan_word(letters);
    printf("\n");
    //every element of occurences1 array is changed accordingly to the elements of the fWord array which was returned from the function
    for (int i = 0; i < 26; i++){
        occurences1[i] = fWord[i];
    }

    //resets the values of the elements inside the letters array
    for (int i = 0; i < 26; i++){
        letters[i] = 0;
    }

    //Asks the user the second word but the process of storing input is presented in the function scan_word
    printf("Enter the second word: ");
    //calls the function scan_word while passing the letters array as the argument and stores the returned value onto sWord
    sWord = scan_word(letters);
    printf("\n");
    //every element of occurences2 array is changed accordingly to the elements of the fWord array which was returned from the function
    for (int i = 0; i < 26; i++){
        occurences2[i] = sWord[i];
    }

    //calls the function isAnagram
    if(isAnagram(occurences1, occurences2)){        //if returned value is 1, which is true, it is an anagram
        printf("The words are anagrams\n");
        return 0;
    }
    printf("The words are not anagrams\n");         //else, returned value is 0, which is false
    return 0;
}
```

2.

```c
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>

//Functions cannot return an array so I used a pointer here
int *scan_word(int occurences[26]){
    char c;
    //Gets an input from the user until a space is detected
    while((c = getchar()) != '\n'){
        //checks if the input are letters
        if(isalpha(c)){
            //Increments the value of the index located by subtracting the position of the letter minus the position of 'A'

            occurences[toupper(c) - 'A']++;
        }
    }
    //returns the address of the array occurences
    return occurences;
}

//Function for checking if anagram
bool isAnagram(int occurences1[26],int occurences2[26]){
    //same variable is used as an indicator
    int same = 1;
    //goes through every element of occurences1 array and compares it with every element of occureces2 array
    for (int i = 0; i < 26; i++){
        //if an element is not the same, the value of same is changed to 0
        if (*(occurences1 + i) != *(occurences2 + i)){
            same = 0;
            break;
        }
    }
    //returns a bool type value (0 or 1)
    return same;
}

int main(void){
    //declaration of variables and arrays
    int *fWord, *sWord, letters[26] = {0};
    int occurences1[26], occurences2[26];

    //Asks the user the first word but the process of storing input is presented in the function scan_word
    printf("Enter the first word: ");
    //calls the function scan_word while passing the letters array as the argument and stores the returned value onto fWord
    fWord = scan_word(letters);
    printf("\n");
    //every element of occurences1 array is changed accordingly to the elements of the fWord array which was returned from the function
    for (int i = 0; i < 26; i++){
        *(occurences1 + i) = *(fWord + i);
    }

    //resets the values of the elements inside the letters array
    for (int i = 0; i < 26; i++){
        *(letters + i) = 0;
    }

    //Asks the user the second word but the process of storing input is presented in the function scan_word
    printf("Enter the second word: ");
    //calls the function scan_word while passing the letters array as the argument and stores the returned value onto sWord
    sWord = scan_word(letters);
    printf("\n");
    //every element of occurences2 array is changed accordingly to the elements of the fWord array which was returned from the function
    for (int i = 0; i < 26; i++){
        *(occurences2 + i) = *(sWord + i);
    }

    //calls the function isAnagram
    if(isAnagram(occurences1, occurences2)){        //if returned value is 1, which is true, it is an anagram
        printf("The words are anagrams\n");
        return 0;
    }
    printf("The words are not anagrams\n");         //else, returned value is 0, which is false
    return 0;
}
```

Github Link: https://github.com/ainzzcutie/CMSC21.git