CD Module Exam Practice Questions

1. What is the value of result when the following code is executed?

```processing
int[] data = {10, 70, 20, 90, 30, 80};
int result = data.length;
```

2. What is the value of result when the following code is executed?

```processing
int[] data = {10, 70, 20, 90, 30, 80};
int result = data[data.length-1];
```

3. What is the value of result when the following code is executed? Yes, I am very annoying.

```processing
int[] data = {10, 70, 20, 90, 30, 80};
int result = data[data.length-1] * data[data.length - data.length];
```

4. Create an array named ages that can hold ages of 5000 people. Choose the most appropriate data type.

5. Create an array named salaries that can hold salaries of 80000 people. Choose the most appropriate data type.

6. Create an array named letters that can hold all letters of the English alphabet. Choose the most appropriate data type and the minimum size.

7. Create an array named switches that can hold status of each individual switch in the house (on/off). Assume there are 200 switches in the house. Choose the most appropriate data type.

8. Create an array named data to hold the values 10, 70, 20, 90. Only one statement is allowed.

9. Create an array named data to hold the values -5, 20, 1, 0, 8, 1.5. Only one statement is allowed.

10. Draw the memory diagram for each of the arrays created in the previous two questions.

11. Complete the following code that displays each item of the array data.

```processing
int i = _____;
while(i < ___________) {
    print(data[i]+" ");
    i++;
}
```

12. Complete the following code that displays each item of the array data, but from last item to first item.

```processing
int i = _____;
while(i > ___________) {
    print(data[i]+" ");
    i______;
}
```

13. Complete the following code that multiplies all the items of the array data, and stores in variable result.

```processing
int result = _______;
for(________; _________; _______) {
    result*=data[i];
}
```

14. Complete the following code that checks if there exists any item in the array data that is greater than the item after it.

```processing
boolean foundSuchItem = false;
for(________; _________; _______) {
    if(data[i] > data[i+1]) {
        foundSuchItem = true;
    }
}
```

15. (Advanced) How can you make the above code more efficient (if not already done).

16. Complete the following code that checks if there exists any item in the array data that is equal to the item before it.

```processing
boolean foundSuchItem = false;
for(________; _________; _______) {
    if(data[i] == data[i-1]) {
        foundSuchItem = true;
    }
}
```

17. Complete the following code that checks if there exists any item in the array data that is greater than

the item after it.

```processing
boolean foundSuchItem = false;
for(________; _________; _______) {
    if(data[i] > data[i+1]) {
        foundSuchItem = true;
    }
}
```

18. Given an integer array foo containing somewhere between 2 and 1000 items, write a statement to store the second-last item into a variable result.

19. Given an integer array foo containing somewhere between 2 and 1000 items, write a statement to store the index of the last item into a variable result.

20. Given an integer array foo containing somewhere between 2 and 1000 items, write a statement to store the median (middle item) in a variable result. It's clear which one is the middle item if there are an odd number of items. If there are an even number of items, assuming the second of the two contending items is the middle item (to make things easy). If you want a slightly more interesting problem, assume the first of the two contending items is the middle item for an array containing even number of items.

21. Create an array named foo that holds 3 sub-arrays, each containing 5 integers.

22. Create an array named bar that holds 3 sub-arrays. First sub-array can hold up to 3 integers, second sub-array can hold up to 5 integers, third sub-array can hold up to 2 integers.

23. Create an array named dip that holds the sub-arrays containing the following values:

    1. first sub-array: 10, 70, 20, 90
    2. second sub-array: 30, 50
    3. third sub-array: 70, 90, -10

24. If you solved the previous question using more than one statements, solve it again using just one statement.

25. Create an array named dip that holds the sub-arrays containing the following values:

    1. first sub-array: 1, 2, 3
    2. second sub-array: 4, 5, 6
    3. third sub-array: 7, 8, 9
    4. fourth sub-array: 10, 11, 1.2

26. Draw the memory diagram for each of the arrays created in the previous 5 questions.

27. Explain the difference between a reference copy and an instance copy. Give an example of each.

28. What is the output when the following code is executed? Explain why that is the case?

```processing
int[] data1 = {10, 70, 20, 90, 30, 80};
int[] data2 = {10, 70, 20, 90, 30, 80};
if(data1 == data2) {
    println("Ronaldo");
}
else {
    println("Messi");
}
```

Processing

29. Assuming the existence of an array named foo that holds at least (but not necessarily) one item, create a reference copy of foo into dip and instance copy of foo into bar.

30. What are the contents of the three arrays after the following code is executed?

```processing
int[] a = {10, 20, 30, 40};
int[] b = a;
int[] c = {50, 60, 70};
a = c;
b[2] = 80;
c = b;
```

Processing

31. What is the value of result when the following code is executed?

```processing
int[] data = {10, 70, 20, 90};
int result = 0;
for(int i=1; i < data.length; i++) {
    result+=i;
}
```

Processing

32. What is the value of result when the following code is executed?

```processing
int[] data = {10, 70, 20, 90};
int result = 0;
for(int i=1; i < data.length; i++) {
    result+=data[i];
}
```

Processing

33. What is the value of result when the following code is executed?

Processing

```processing
int[] data = {10, 70, 20, 90};
int result = 0;
for(int i=0; i < data.length; i++) {
    result+=data[i];
}
```

34. What is the value of result when the following code is executed?

Processing

```processing
int[] data = {10, 70, 20, 90, 30, 80};
int result = 0;
for(int i=0; i < data.length; i+=2) {
    result+=data[i];
}
```

35. What is the value of result (if any) when the following code is executed?

Processing

```processing
int[] data = {10, 70, 20, 90, 30, 80};
int result = 0;
for(int i=data.length-1; i > 0; i++) {
    result+=data[i];
}
```

36. What is the value of result (if any) when the following code is executed?

Processing

```processing
int[] data = {10, 70, 20, 90, 30, 80};
int result = 0;
for(int i=data.length-1; i > 0; i--) {
    result+=data[i];
}
```

37. What is the value of result when the following code is executed?

Processing

```processing
int[] data = {10, 70, 20, 90, 30, 80};
int result = 0;
for(int i=data.length-1; i >= 0; i--) {
    result+=data[i];
}
```

38. What is the value of idx when the following code is executed?

```processing
int[] data = {10, 70, 20, 0, 90, -30, 80};
int idx;
for(idx=0; data[idx] > 0; idx++) {
}
```

39. What is the value of idx when the following code is executed? This should expose the problem with the code from the previous (and current) question. Run it in Processing if you can't figure out what the problem will be.

```processing
int[] data = {10, 70, 20, 90};
int idx;
for(idx=0; data[idx] > 0; idx++) {
}
```

40. What is the value of idx when the following code is executed?

```processing
int[] data = {10, 70, 20, 90};
int idx;
for(idx=0; idx < data.length && data[idx] > 0; idx++) {
}
```

41. Define a function that when passed an array of integers, returns the sum of all odd numbers. Call the function with an array of your choice and store the returned value in a variable named result, which you must declare with a compatible data type.

42. Define a function that when passed an array of floating-point values, returns true if the array contains any negative number, false otherwise. Call the function with an array of your choice and store the returned value in a variable named result, which you must declare with a compatible data type.

43. Define a function that when passed an array of integers, returns true if all the items of the array are divisible by 3 and also by 5, false otherwise. Note: for an empty array it should return true (this is known as the vacuous truth), but don't worry - if your logic is right, you won't have to handle this scenario separately. Call the function with an array of your choice and store the returned value in a variable named result, which you must declare with a compatible data type.

44. Define a function that when passed two integer arrays, returns if they are identical (same items in same order), false otherwise. Call the function with arrays of your choice and store the returned value in a variable named result, which you must declare with a compatible data type.

45. Define a function that when passed an integer array, returns true if every single item of the array occurs exactly once, false otherwise. This is based on something known as the handshake algorithm. Call the function with an array of your choice and store the returned value in a variable named result, which you must declare with a compatible data type.

46. Define a function that when passed a two-dimensional array, returns the highest value in the array. Call the function with an array of your choice and store the returned value in a variable named result, which you must declare with a compatible data type.

47. Define a function that when passed a two-dimensional array such that each sub-array holds two values, representing the x and y coordinates of a point, draw the points on the screen. If you are keen, draw lines from one point (say at index i) to the next point (at index i+1). Call the function with an array of your choice.

48. Define a function that when passed an integer array and two integers (store them in formal parameters `low, high`), returns an array containing all items in the passed array that are in the range `[low, high]`. For example, if the array passed is `{10, 70, 20, 90, 60}`, `low = 20`, `high = 70`, return the array `{70, 20, 60}`.

49. Define a function that when passed an integer, returns an array containing all prime numbers from 2 to that number (inclusive on that side). For example, if the number passed is 10, return the array `{2, 3, 5, 7}`. If the number passed is -8, return the empty array `{}`. You can define another function that checks if a passed integer is prime or not, and then call it in your function.

50. Create an array of arrays of arrays that holds 5 sub-arrays, each containing 10 sub-arrays, each containing 4 items.