A Grab-bag of Drills

In this set of drills we will be tying up any loose ends. There will be three major topics here: Pass-by-reference, 2D arrays, and advanced Robot C applications.

Pass-by-reference: Concepts

1. When do we use pass-by-reference?
2. Write a function that receives two integers and returns both their product and their sum. Hint: There are at least two ways to do this.
3. When should you **not** use pass-by-reference?

##Pass-by-reference: applications

Question 1

Below are two functions that appear to do the same thing, and a main function that uses both functions. Predict the output of the program **without running the code**. Why do you predict what you do?

```
#include <iostream>
using namespace std;

void swapOne(int a, int b)
{
  int temp = a;
  a = b;
  b = temp;
}

void swapTwo(int &a, int &b)
{
  int temp = a;
  a = b;
  b = temp;
}

int main()
{
  int a = 4, b = 3;
  swapOne(a,b);
  cout << a << " " << b << endl;
  swapTwo(a,b);
  cout << a << " " << b << endl;
}
```

Question 2

Write a function that takes in **at least** two integers. These two integers, R1 and R2, will be resistance values. The function should return the equivalent resistance of the two resistors if they are connected in parallel, as well as the equivalent resistance of the two resistors if they are connected in series. Test your function appropriately.

Question 3

A vector in 3D is given by three components:

$$\vec{v} =< x_1, x_2, x_3 >$$

Write a function that receives the components of a 3D vector, $\vec{v}$ and a scalar, α, and returns the vector $\vec{v}_2 = \alpha\vec{v}$. Do not use arrays for this question. You may choose to return an entirely new vector or to overwrite the values of the original one.

Question 4

A vector in 2D is given by two components:

$$\vec{v} =< x, y >$$

Write a function that receives the components of two vectors, $\vec{v}_1$ and $\vec{v}_2$. The function should return both the dot product and the angle between the two vectors.

2D Arrays: Concepts

1. Create a 2D array of integers, and initialize them all to zero. The array must be at least 2x2.
2. If I create the following array, how many rows and columns does it have? `double AR[5][6];`
3. Create a 2x2 array of integers. Have the user enter the values of the array using loops. **Note:** Strictly speaking, you could avoid using loops, since there are only 4 values to read in. The purpose of the question is to get you to practice using loops with 2D arrays, though, so please use them. If you'd prefer, make the arrays 10x10 instead.
4. As in question 3, create a 2x2 array of integers, and initialize them however you want (you may even re-use your code from Q3). Print them to the screen in a nicely formatted way.

2D Arrays: Applications

Note: due to the sheer number of elements in a 2D array, it can be difficult to test your program. For instance, if you have a 10x10 array and you ask the user to enter all of the values, they must enter 100 things! All questions below **must** be solved using 2D arrays, but for testing purposes you can use smaller arrays than those needed to solve the problem in order to make testing manageable.

Question 1

There are five players in a card game that lasts 10 rounds. After each round, players count the remaining cards in their hand an receive a score out of 10. The winner of the game has the fewest points at the end of the 10 rounds. Write a program that asks each user in turn to enter their score for a given round. Keep track of the scores in a 2D array. Once all scores have been entered at the end of the game, print out who won.

Question 2

There are 10 students in a class and they take 5 quizzes. Their final grade is the average of the highest 4 quizzes, with the lowest quiz being dropped. Write a program that has the user enter all grades in a loop (see the note above about making this easier to test!). Compute and output the average for each student.

Robot C: Full programs

1. Assume the robot is in standard lab configuration and placed in an empty room **at a random location in the middle of the floor**. You may assume the room is rectangular and the robot is placed so that it is facing one of the walls. Determine how big the room is.
2. One of the best ways to "condition" sensor data is to take multiple measurements and average them. The sonar sensor is notoriously unreliable. In fact, sometimes the sensor might read 0 or 255 even though there is an object in front of the sensor. Write a program that takes 10 measurements from the sonar sensor. If **all** of the values are 255 or 0, then the distance should be set to -1 to indicate that nothing is in front of the sensor. Otherwise, you should compute the average sensor value, ignoring any values of 255 or 0 when computing it.
3. The bug algorithm: For this question, assume that we've modified the standard lab configuration so that the sonar sensor points 90 degrees to the right, rather than straight ahead. The robot is in a room that contains a single rectangular box, and is oriented so that it will move perpendicular to and towards the box if it moves forward. The robot should move forward until it bumps into the box, back up slightly, turn, and then measure how big the box is.