Problem 1
Write a complete C++ program that does the following:
1. Asks the user to enter 2 integers, x and y. Both should be between 2 and 10 (inclusive), and if either is illegal then the program terminates.
2. Fills a table with characters entered by the user. The number of characters should be as many rows as x and as many columns as double of y.
3. Prints out the characters in the last row.

For example, the following represents one run of the program:

Enter 2 integers: 3 2
Enter 12 characters: M i d t e r m 2 C S C I
The characters in the last row: C S C I

#include <iostream>
using namespace std;

int main ()
{
    int x, y, r, c;
    char table [10] [20];
    cout << "Enter 2 integers: ";
    cin >> x >> y;
    if (x < 2 || x > 10 || y < 2 || y > 10) exit (1);
    cout << "Enter " << x * y * y << " characters: ";
    for (r = 0; r < x; r++)
        for (c = 0; c < 2 * y; c++)
            cin >> table [r] [c];
    cout << "The characters in the last row: ";
    for (c = 0; c < 2 * y; c++)
        cout << table [x - 1] [c] << " ";
    return 0;
} //main

Problem 2
Consider the following C++ program. What is the output from the program?

#include <iostream>
using namespace std;

int func1 (double &d, string s)
{
    s = "midterm";
    d = 13.14 - 3.14;
    cout << "s";
    return 13 + 1;
}
int func2 (int &a, int &b, int c)
{
    a = b + c;
    b = 1;
    return c;
}
int main ()
{
    double piDoubled = 3.14 + 3.14;  //line A
    string str = "CSCI";
    func1 (piDoubled, str);
    piDoubled = 3.14 + 3.14;  //line B
    cout << func1 (piDoubled, str);
    cout << piDoubled << piDoubled;

    int x = 1, y = 11;
    cout << 2 * (func2 (x, y, x));  //line D
    cout << x << y;  //line E

    return 0;
}

(a) s
(b) s14
(c) 1010
(d) 2
(e) 121

Problem 3
Write a recursive function called findDigitalRoot. The function both takes in and returns a positive integer. In particular, the returned integer is a single digit and it is obtained by repeatedly summing the digits of an integer. For example, the digital root of 2048 is 5, since $2 + 0 + 4 + 8 = 14$ and $1 + 4 = 5$. Additionally, the digital root of 19683 is 9, since $1 + 9 + 6 + 8 + 3 = 27$ and $2 + 7 = 9$. Lastly, the digital root of 123456789 is 9, since $1 + 2 + \ldots + 9 = 45$ and $4 + 5 = 9$.

A program that uses the function follows. Your function must complete this program.

int main ()
{
    cout << findDigitalRoot (111); //prints: 3
    cout << findDigitalRoot (2048); //prints: 5
    cout << findDigitalRoot (19683); //prints: 9
    cout << findDigitalRoot (123456789); //prints: 9
    return 0;
} //main

Alternatively, you may write two recursive functions. The first, findDigitalRoot, will be called by the main function above. The second will be a helper function whose goal is to simply perform a computation needed by findDigitalRoot.

int findDigitalRoot (int n)
{
    if (n < 10) return n;
    int sum = 0;
    sum = n % 10 + findDigitalRoot (n / 10);
    if (sum >= 10) sum = findDigitalRoot (sum);
    return sum;
} //findDigitalRoot
OR:

```c
int sumDigits (int n)
{
    if (n < 10) return n;
    else return sumDigits (n / 10) + n % 10;
}  //sumDigits

int findDigitalRoot (int n)
{
    if (n < 10) return n;
    return findDigitalRoot (sumDigits (n));
}  //findDigitalRoot
```

**Problem 4**
Write header lines (prototypes) for the five functions that are called by the following main program. Do not supply the blocks for the functions.

```c
int main() {
    int i = 123, arr1 [3] = {1, 2, 3}, arr2 [2][2] = {{1, 0}, {2, 4}};
    double d1 = 1.23, d2 = 12.3;

    printfLine (arr2, 2, 2);  //prints: 1 0 2 4
    printFancy (arr1, 3);  //prints: 1 * 2 * 3 ***
    arr1 [0] = 4;
    printFancy (arr1, 3);  //prints: 4 **** 2 ** 3 ***
    cout << doNothing (i, (int) d1);  //prints: This is a useless function
    switchValues (d1, d2);  //switches the values: now, d1 = 12.3 and d2 = 1.23
    goodDayWishes ();  //prints: Have a good day

    return 0;
}
```

(a) void printfLine (int array [][]2, int rowCap, int colCap);
(b) void printFancy (int array [], int capacity);
(c) string doNothing (int i1, int i2);
(d) void switchValues (double &d1, double &d2);
(e) void goodDayWishes ();