Problem 1 Write the best title lines for the functions that are called by the following main program. Do not supply the blocks for the functions.

```cpp
int main() {
    int x = 12, y = 36, w = 1024;
    double z[4] = {1.1, 1.11, 2.5, 5.7};

    // a. The function percentage reports x as a percentage of y, eg as 33.3.
    cout << percentage(x, y) << endl; // (a)

    // b. The function nearTo reports whether two numbers are within 0.1. Here it says yes.
    if (nearTo(z[0], z[1])) cout << "Very close\n"; // (b)

    // c. The function cutZeros removes any zeros from a number.
    cutZeros(w); // (c)
    cout << w << endl; // print 124

    // d. The function unitsDigits makes a number (here 1125) from the units digits of the array elements.
    cout << unitsDigits(z, 4) << endl; // (d)

    // e. A mystery function.
    mystery(mystery(nearTo(z[0], percentage(x,y)))); // (e)

    return 0;
}
```

(a) Title line for `percentage` as called at the line marked (a).
**Answer:** `double percentage(int x, int y)`
(b) Title line for `nearTo` as called at the line marked (b).
**Answer:** `bool nearTo(double x, double y)`
(c) Title line for `cutZeros` as called at the line marked (c).
**Answer:** `void cutZeros(int &x)`
(d) Title line for `unitsDigits` as called at the line marked (d).
**Answer:** `int unitsDigits(double z[], int cap)`
(e) Title line for `mystery` as called at the line marked (e).
**Answer:** `bool mystery(bool b)`
Problem 2  Write C++ statements to carry out the following tasks. Do not write complete programs. Each answer should be at most 3 lines of C++. Assume the following variables have been declared and initialized with positive values.

```cpp
int x, y;
```

(a) Print 53 copies of the word Hello on a single (long) line of output.

**Answer:**

```cpp
for (int c = 1; c <= 53; c++) cout << "Hello ";
cout << endl;
```

(b) Make C++ calculate and print the remainder when variable y is divided by variable x.

**Answer:**

```cpp
cout << y % x << endl;
```

(c) Print the square root of 2020. Use a C++ function for the calculation.

**Answer:**

```cpp
cout << sqrt(2020.0) << endl;
```

(d) Print a random number in the range 100 to 999, inclusive. Use a C++ function.

**Answer:**

```cpp
cout << rand() % 900 + 100 << endl;
```

(e) Print the digits of the variable y backwards. So if y is 19683, print 38691.

**Answer:**

```cpp
while (y > 0) {
    cout << y % 10;
    y = y / 10;
}
```
Problem 3  Write a function called swapRows that swaps two rows of a 2-dimensional array of integers with 3 columns. The function should use 5 parameters as follows: the array name, the number of rows, the number of columns and the numbers of the two rows to be swapped.

Excessively long solutions that use more than 12 lines of code may lose points. A program that uses the function swapRows follows.

```cpp
int main() {
    int x[4][3] = {{0,1,2}, {3,4,5}, {6,7,8}, {9,10,11}};
    swapRows(x, 4, 3, 0, 1); // swaps the first two rows of x
    cout << x[0][0] << endl; // prints 3
    return 0;
}
```

Answer:

```cpp
void swapRows(int a[][3], int rows, int cols, int i, int j) {
    for (int c = 0; c < cols; c++) {
        int temp = a[i][c];
        a[i][c] = a[j][c];
        a[j][c] = temp;
    }
}
```
Problem 4  The recursive function `removeRepeats` removes repeated digits from a positive integer. For example `removeRepeats(1002011003)` returns 1020103. As the example shows, a repeat is a digit that matches the previous digit.

An implementation of this function with parts of the code covered up is given below. There is also a main program that uses it.

Some pieces of code have been replaced by PART (a), PART (b), and so on. To answer the parts of this question you should supply the C++ code that was replaced. Each answer must fit on a single line.

```cpp
int removeRepeats(PART (a)) {
    if (x < 10) PART (b);
    int y = PART (c);
    if (PART (d)) return y;
    else return PART (e);
}

int main() {
    cout << removeRepeats(1002011003) << endl; // prints 1020103
    cout << removeRepeats(1234) << endl; // prints 1234
    cout << removeRepeats(1111) << endl; // prints 1
    return 0;
}
```

(a) Give a replacement for PART (a) to declare the parameter `x`:

**Answer:** PART (a) is `int x`

(b) Give a replacement for PART (b) as the base case of recursion:

**Answer:** PART (b) is `return x`

(c) Give a replacement for PART (c) to remove repeats from `x / 10`:

**Answer:** PART (c) is `removeRepeats( x / 10 )`

(d) Give a replacement for PART (d) to detect when `y` is the answer we want:

**Answer:** PART (d) is `x % 10 == y % 10`

(e) Give a replacement for PART (e) to return an answer that is different from `y`, if that is needed:

**Answer:** PART (e) is `10 * y + x % 10`
Problem 1 Write the best title lines for the functions that are called by the following main program. Do not supply the blocks for the functions.

```c
int main() {
    double x = 12.0, y = 24.0, w = 1024.3;
    int z[5] = {1, 1, 2, 5, 8};

    // a. The function percentage reports x as a percentage of y. Here it gives 50.
    z[4] = percentage(x, y); // (a)
    cout << z[4] << "%" << endl; // prints 50%

    // b. The function dividesInto reports whether a first number divides a second. Here it says no.
    if ( dividesInto(z[2], z[3]) ) cout << "Is a factor\n"; // (b)

    // c. The function round is used to round a number to the nearest integer.
    round( w ); // (c)
    cout << w << endl; // print 1024.0

    // d. The function median reports the median (the middle value) from an array of integers.
    cout << median(z, 5) << endl; // (d)

    // e. A mystery function.
    mystery(mystery(dividesInto(1,2), z[2]), z[2]); // (e)
    return 0;
}
```

(a) Title line for `percentage` as called at the line marked (a).
**Answer:** `int percentage(double x, double y)`

(b) Title line for `dividesInto` as called at the line marked (b).
**Answer:** `bool dividesInto(int x, int y)`

(c) Title line for `round` as called at the line marked (c).
**Answer:** `void round(double &x)`

(d) Title line for `median` as called at the line marked (d).
**Answer:** `int median(int z[], int cap)`

(e) Title line for `mystery` as called at the line marked (e).
**Answer:** `bool mystery(bool b, int x)`
Problem 2  Write C++ statements to carry out the following tasks. Do not write complete programs. Each answer should be at most 3 lines of C++. Assume the following variables have been declared and initialized with positive values.

```
int x, y;
```

(a) Print x copies of the word Hello on a single (long) line of output.

**Answer:**
```
for (int c = 1; c <= x; c++) cout << "Hello ";
cout << endl;
```

(b) Make C++ calculate and print the exact quotient when the variable y is divided by the variable x.

**Answer:**
```
cout << y / (x * 1.0) << endl;
```

(c) Print the square root of 2021. Use a C++ function for the calculation.

**Answer:**
```
cout << sqrt(2021.0) << endl;
```

(d) Print a random number in the range 2000 to 2029, inclusive. Use a C++ function.

**Answer:**
```
cout << rand() % 30 + 2000 << endl;
```

(e) Print the last digits of the variables x and y.

**Answer:**
```
cout << x % 10 << y % 10;
```
Problem 3  Write a function called \textit{countMatch} that counts the number of matching entries between two arrays of integers. Two entries match if they have the same index and the same value. The function should use 4 parameters as follows: the first array name, the first array capacity, the second array name, the second array capacity. (Do not assume that the arrays have the same capacity.)

Excessively long solutions that use more than 12 lines of code may lose points. A program that uses the function \textit{swapRows} follows.

```c
int main() {
    int x[7] = { 3, 1, 4, 1, 5, 9, 2};
    int y[6] = { 1, 1, 2, 3, 5, 8};

    cout << countMatch(x, 7, y, 6) << endl; // prints 2: the matches are
    // the value 1 at index 1
    // the value 5 at index 4

    return 0;
}
```

Answer:

```c
int countMatch(int a[], int aCap, int b[], int bCap) {
    int ans = 0;
    for (int c = 0; c < aCap && c < bCap; c++) {
        if (a[c] == b[c]) ans++;
    }
    return ans;
}
```
The recursive function `printBlocks` splits a number into blocks, which it prints. The blocks are separated by the 0 digits of the number. For example, if the number is 1002011003 the function would print 1 2 11 3. Note that printed blocks are separated by single spaces.

An implementation of this function with parts of the code covered up is given below. There is also a main program that uses the function.

Some pieces of code have been replaced by `PART (a)`, `PART (b)`, and so on. To answer the parts of this question you should supply the C++ code that was replaced. Each answer must fit on a single line.

```cpp
PART (a) printBlocks(int x) {
    if (x == 0) PART (b);
    PART (c);
    if (x % 10 == 0) PART (d);
    if (PART (e)) cout << " ";
    cout << x % 10;
}

int main() {
    printBlocks(1002011003); cout << endl; // prints 1 2 11 3
    printBlocks(1000); cout << endl; // prints 1
    printBlocks(1011001); cout << endl; // prints 1 11 1
    printBlocks(0); cout << endl; // prints
    return 0;
}
```

(a) Give a replacement for `PART (a)` to specify the answer type:

**Answer:** `PART (a)` is `void`

(b) Give a replacement for `PART (b)` as the base case of recursion:

**Answer:** `PART (b)` is `return`

(c) Give a replacement for `PART (c)` to process everything except the last digit of `x`:

**Answer:** `PART (c)` is `printBlocks( x / 10 )`

(d) Give a replacement for `PART (d)` to deal with numbers that end with a 0:

**Answer:** `PART (d)` is `return`

(e) Give a replacement for `PART (e)` to decide whether we need to insert a space:

**Answer:** `PART (e)` is `x / 10 % 10 == 0`