Problem 1 *(10 points)* Write the best **title lines** for the functions that are called by the following main program. **Do not supply the blocks for the functions.**

```
int main() {
   int x = 12, y = 36, w = 21331;
   double z[4] =  {1.1, -1.11, 2.5, 5.7};

   // a. The function ratio returns x/y rounded to 2 decimal places eg as 0.33.
   cout << ratio(x, y) << endl;  // (a)

   // b. The function sameSign reports whether two numbers have the same sign. Here it returns false.
   if ( !sameSign(z[0], z[1]) ) cout << "Opposite signs\n";  // (b)

   // c. The function cutDuplicates removes all duplicate digits from a number.
   cutDuplicates( w );      // (c)
   cout << w << endl;       // prints 2

   // d. The function inWords makes the word for a number (here twelve) from an integer parameter.
   cout << inWords(x) << endl;   // (d)

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

   return 0;
}
```

(a) Title line for **ratio** as called at the line marked (a).

Answer: `double ratio(int x, int y)`

(b) Title line for **sameSign** as called at the line marked (b).

Answer: `bool sameSign(double x, double y)`

(c) Title line for **cutDuplicates** as called at the line marked (c).

Answer: `void cutDuplicates(int &x)`

(d) Title line for **inWords** as called at the line marked (d).

Answer: `string inWords(int x)`

(e) Title line for **mystery** as called at the line marked (e).

Answer: `bool mystery(bool b)`

2 points per part.
1 point for completely correct parameters.
1 point for completely correct return type.

Problem 2 *(10 points)* Consider the following C++ program. The program makes use of a function `reverseArray` that reverses the entries in an array. So that if the array has capacity 5 and entries 1, 2, 3, 4, 5 in this order, the function reverseArray moves the entries so that they are ordered as 5, 4, 3, 2, 1.

Make sure to use your own 8-digit CUNY ID number as the number entered as input to the program. It would be a very bad idea to give answers based on another student's ID number!

```
int main() {
   int id, a[10] = {3, 1, 4, 1, 5, 9, 2, 6, 5, 3};

   cout << "Enter your 8-digit CUNY id number: ";
   cin >> id;     // assume that the user types YOUR OWN CUNY ID number

   cout << id << endl;                                          // line (a)
   cout << a[ id % 10 ] << endl;                                // line (b)

   reverseArray(a, 10);
   cout << a[ a[0] ] << endl;                                   // line (c)
   cout << a[ 4 ]  % a[ 5 ] << endl;                            // line (d)
   cout << a[ 0 ]  + a[ 2 ] % a[ 3 ] << endl;                   // line (e)

   return 0;
}
```

Enter your 8-digit CUNY id number: (a) What is the output from the instruction beginning on line (a)?

Answer:

```
12345678

This answer is based on the ID number 12345678.  Actual answers will be different.
```

(b) What is the output from the instruction beginning on line (b)?

Answer:

```
5

The answer will be the array entry indexed by the last digit of the answer to (a).
```

(c) What is the output from the instruction beginning on line (c)?

Answer:

```
2
```

(d) What is the output from the instruction beginning on line (d)?

Answer:

```
4
```

(e) What is the output from the instruction beginning on line (e)?

Answer:

2 points per part. One point partial credit for answers that are almost correct except for a tiny error. (A tiny error might be extra or missing new lines or spaces.)

Check carefully that the 8 digit CUNY ID number is correct. If not email me what would be correct and what was used.

Note that the answer to part b depends on the ID number.

Problem 3 *(10 points)* Write a function called *sumAbsolute* that returns the sum of the absolute values of the entries in an array with base type double. (The absolute value $|x|$ of a number is obtained by ignoring its sign, so for example $|-4| = |4| = 4$.) The function should use 2 parameters as follows: the array name, the capacity.

Excessively long solutions that use more than 10 lines of code may lose points. A program that uses the function *sumAbsolute* follows.

```
int main() {
   double x[4] = { -1, -2, 3, 0};
   cout << sumAbsolute(x, 4) << endl;  // prints 6 (this is found as 1 + 2 + 3 + 0).
   return 0;
}
```

Answer:

```
double sumAbsolute(double a[], int cap) {
   double answer = 0.0;
   for (int c = 0; c < cap; c++) {
      if (a[c] < 0) answer -= a[c];
      else answer += a[c];
   }
   return answer;
}
```

```
Award partial credit for the following elements of a program:

3 points for title line: 1 for return type, 1 for array parameter, 1 for other parameter.

1 point for declaring and initializing an answer variable.

2 points for a for loop.

1 point for an if statement

2 points for correctly changing the answer in each case.

1 point for returning the answer.

If a program follows a different (but reasonable) plan. Try to award
partial credit for meeting similar goals in the program.

If you feel that a solution is too long and messy to grade properly
(longer than about 10 lines if they were normally spaced out)
judge a rough score for partial credit by the above milestones and
then adjust the score down by the following guidelines.

Very long and messy but looks probably correct --- max allowed score is 7/10.
Very long and messy and looks partially correct --- max allowed score is 5/10.
Very long and messy and looks probably badly wrong --- max allowed score is 3/10.
```

Problem 4 *(10 points)* The recursive function `changeDigits` has 3 parameters `x, a, b`. The parameter `x` is a positive integer and the other parameters are single digit integers. The function considers the digits of `x` and returns a result obtained by changing any copy of `a` to become `b`. For example `changeDigits(1331, 1, 9)` returns 9339 and `changeDigits(1331, 1, 0)` returns 330.

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.

```
int changeDigits(PART (a)) {
   if ( PART (b) ) return b;
   if ( PART (c) ) return x;
   int y = PART (d);
   int z = changeDigits(PART (e));
   return PART (f);
}

int main() {
   cout << changeDigits(1331, 1, 2) << endl;  // prints 2332
   cout << changeDigits(1331, 1, 0) << endl;  // prints  330
   cout << changeDigits(1331, 2, 5) << endl;  // prints 1331
   return 0;
}
```

(a) Give a replacement for PART (a) to declare the parameters `x, a, b` :

Answer: PART (a) is `int x, int a, int b`

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

Answer: PART (b) is `x == a`

(c) Give a replacement for PART (c) as a second base case of recursion:

Answer: PART (c) is `x < 10`

(d) Give a replacement for PART (d) to change digits in `x / 10`:

Answer: PART (d) is `changeDigits( x / 10 , a, b)`

(e) Give a replacement for PART (e) to change the last digit, if necessary:

Answer: PART (e) is `x % 10, a, b`

(f) Give a replacement for PART (f) to return the answer:

Answer: PART (f) is `10 * y + z`

1 point each for parts b and c. 2 points for each other part.
Do not penalize answers that give the whole line instead of the required part.
Accept any answer that works identically, but if it works in any way differently it will be wrong and gets 0 points.