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

```
int main() {
   int a[5] = {3,1,4,1,5};
   int x[2][3] = {{0,1,3},{2,4,8}};
   string s= "Hello";
   string t;

   cout << average(x, 2, 3) << endl;        // prints the average:  3.0
   t = doubleIt(s);  cout << t << endl;     // prints:  HelloHello
   reverseCols(x, 2, 3);                    // prints:  3 1 0, 8 4 2
   if (isPositive(a[0])) cout << "Positive" << endl;
                                            // prints:  Positive
   cout << midEntry(a, 5) << endl;          // prints:  4
   return 0;
}
```

(a) Title line for **average**

Answer:

```
double average(int x[][3], int r, int c)
```

(b) Title line for **doubleIt**

Answer:

```
string doubleIt(string s)
```

(c) Title line for **reverseCols**

Answer:

```
void reverseCols(int x[][3], int r, int c)
```

(d) Title line for **isPositive**

Answer:

```
bool isPositive(int x)
```

(e) Title line for **midEntry**

Answer:

```
int midEntry(int a[], int cap)
```

Each part gets 2 points. 1 point for the return type and 1 point for the parameters.

Problem 2 *(10 points)* Consider the following C++ program.

```
int mystery(int &a, int b, int c, int d[]) {
   int temp = a;
   a = b;
   b = temp;
   c = c + 1;
   d[b]=a;
   return c;
}

int main() {
    int x = 2, y = 3, z = 5;
    int array[6] = {1, 2, 3, 4, 5, 6};
    cout << z % y << endl;                                  // line (a)
    if (array[1] == 1 && x < y) cout << "Hello\n";          // line (b)
    else cout << array[array[2]] << endl;
    array[5] = mystery(y, z, x, array);
    cout << x << endl;                                      // line (c)
    cout << y << z << endl;                                 // line (d)
    for (int i = 0; i < 6; i++) cout << array[i];           // line (e)
    cout << endl;
    return 0;
}
```

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

Answer:

```
2
```

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

Answer:

```
4
```

(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:

```
55
```

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

Answer:

```
123553
```

Each part gets 2 points (no partial credit for any part).
If it's a tiny error, like spacing or an extra line give 1 point partial.

Problem 3 *(10 points)* Write a function called *absoluteArray* that replaces all negative elements in an array of decimal numbers by their absolute values. For example an entry of -2.5 would be replaced by 2.5.

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

```
int main() {
   double x[4] = {0, -2.5, -0.33, 1};
   absoluteArray(x, 4);
   for (int i = 0; i < 4; i++) cout << x[i] << " ";   // prints 0 2.5 0.33 1
   cout << endl;
   return 0;
}
```

Answer:

```
void absoluteArray(double a[], int c) {
   for (int i = 0; i < c ; i++)
      if (a[i] < 0) a[i] = - a[i];
}
```

Award partial credit for the following elements of a function:

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

3 points for a for loop.

2 point for an if statement

2 points for correctly changing array entries.

If a program follows a different (but reasonable) plan. Try to award
partial credit for meeting similar goals in the program. For example,
students might use the abs function.

Long answers with messy code cannot score more than 5 points.

Problem 4 *(10 points)* This problem considers a recursive function called *secondDigit* that calculates the second digit of an integer parameter (with at least two digits). For example *secondDigit(34566)* would give an answer of 4. The function should use a single parameter called x.

(a) Give a condition of x that detects the base case.

Answer: `x < 100`

(b) What answer should be returned when the condition in (a) applies?

Answer: `x % 10`

(c) Give a formula for *secondDigit(x)* that applies when x is not covered by the base case. This formula must make use of the result of an easier application of the *secondDigit* function.

Answer: `secondDigit(x / 10)`

Write a complete implementation of the *secondDigit* function. Excessively long solutions that use more than 10 lines of code may lose points.

Answer:

```
int secondDigit(int x) {
   if (x < 100) return x % 10;
   return secondDigit(x / 10);
}
```

Parts a, b and c get 2 points each. There are other options for the answers, but if they are to get credit they must be consistent with each other.

4 points for the function code:

2 of the 4 are for a title line 1 for a correctly placed base case 1 for a correctly placed recursive call.

A non-recursive implementation of the function can also get 4 points if it is correct.

If a, b or c is answered incorrectly but the correct answer appears in the function code, give credit for whichever of a, b or c that it is.

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

```
int main() {
   int a[5] = {3,1,4,1,5};
   int x[2][3] = {{0,1,3},{2,4,5}};
   string s= "Hello";
   string t;

   cout << average(a, 5) << endl;           // prints the average:  2.8
   t = reverse(s);  cout << t << endl;      // prints:  olleH
   reverseRows(x, 2, 3);                    // prints:  2 4 5, 0 1 3
   if (hasRepeat(a, 5)) cout << "Has repeat" << endl;
                                            // prints:  Has repeat
   t = entries(a, 5); cout << t << endl;    // prints:  3,1,4,1,5
   return 0;
}
```

(a) Title line for **average**

Answer:

```
double average(int a[], int cap)
```

(b) Title line for **reverse**

Answer:

```
string reverse(string s)
```

(c) Title line for **reverseRows**

Answer:

```
void reverseRows(int x[][3], int r, int c)
```

(d) Title line for **hasRepeat**

Answer:

```
bool hasRepeat(int a[], int cap)
```

(e) Title line for **entries**

Answer:

```
string entries(int a[], int cap)
```

Each part gets 2 points. 1 point for the return type and 1 point for the parameters.

Problem 2 *(10 points)* Consider the following C++ program.

```
string mystery(int a, int &b, int c, string d[]) {
   int temp = a;
   a = b;
   b = temp;
   c = c + 1;
   d[b]=d[c];
   return d[a];
}

int main() {
    int x = 5, y = 2, z = 3;
    string array[6] = {"CS111","Midterm","2","today","is","easy"};
    cout << y % y << endl;                                    // line (a)
    if (x < y || z < y) cout << array[x - y - z] << endl;  // line (b)
    else cout << x * y + z << endl;
    array[0] = mystery(x, y, z, array);
    cout << x << endl;                                        // line (c)
    cout << y << z << endl;                                   // line (d)
    for (int i = 0; i < 6; i++) cout << array[i] << " ";   // line (e)
    cout << endl;
    return 0;
}
```

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

Answer:

```
0
```

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

Answer:

```
13
```

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

Answer:

```
5
```

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

Answer:

```
53
```

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

Answer:

```
2 Midterm 2 today is is
```

Each part gets 2 points (no partial credit for any part).
If it's a tiny error, like spacing or an extra line give 1 point partial.

Problem 3 *(10 points)* Write a function called *oddCount* that counts the number of odd elements in an array of integers. For example, if the array contains 0, 1, 4, 5, 7 then the function should return an answer of 3. This is because the three entries 1, 5 and 7 are odd (and the others are even).

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

```
int main() {
   int x[5] = {0, 1, 4, 5, 7};
   cout << oddCount(x, 5) << endl;                    // prints 3
   return 0;
}
```

Answer:

```
int oddCount(int a[], int c) {
   int count = 0;
   for (int i = 0; i < c ; i++)
      if (a[i] % 2 != 0) count++;
   return count;
}
```

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

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

1 point for initializing a counter

2 points for a for loop.

2 points for an if statement

1 point for changing the counter.

1 point a return statement.

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

Long answers with messy code cannot score more than 5 points.
```

Problem 4 *(10 points)* This problem considers a recursive function called *number2s* that calculates the number of digits **equal to 2** in an integer parameter (that is not negative). For example *number2s(123121)* would give an answer of 2. The function should use a single parameter called *x*.

(a) Give a condition of *x* that detects the base case.

Answer: `x == 0`

(b) What answer should be returned when the condition in (a) applies?

Answer: `0`

(c) Give a formula for *number2s(x)* that applies when *x* is not covered by the base case and has a last digit of 2. This formula must make use of the result of an easier application of the *number2s* function.

Answer: `1 + number2s(x / 10)`

(d) Give a formula for *number2s(x)* that applies when *x* is not covered by the base case and does not have a last digit of 2. This formula must make use of the result of an easier application of the *number2s* function.

Answer: `number2s(x / 10)`

Write a complete implementation of the *number2s* function. Excessively long solutions that use more than 10 lines of code may lose points.

Answer:

```
int number2s(int x) {
   if (x == 0) return 0;
   if (x % 10 == 2) return 1 + number2s(x / 10);
   return number2s(x / 10);
}
```

Parts a, b and c get 2 points each. Part d gets 1 point. There are other options for the answers, but if they are to get credit they must be consistent with each other.

3 points for the function code:

1 of the 3 are for a title line 1 for a correctly placed base case 1 for a correctly placed recursive call.

A non-recursive implementation of the function can also get 3 points if it is correct.

If a, b or c is answered incorrectly but the correct answer appears in the function code, give credit for whichever of a, b or c that it is.