1 Review

1. Find and correct the bugs of the following program? What is the output?

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

double func1(double x) {
  return x * x;
}

double& func2(double& x) {
  double y;
  y = x * x;
  return y;
}

void func3(double& x, double y) {
  x = 100.;
  y = x * x;
}

void func4(const double& x, double& y) {
  x = 100.;
  y = x * x;
}

void func5(double x, double* y) {
  y = x * x;
}

int main()
{
  double x = 10.0;
  double y = 0;
  double* py = &y;

  cout << "x = " << x << endl;
  cout << "func 1: x^2 = " << func1(x) << endl;
  cout << "func 2: x^2 = " << func2(x) << endl;
  cout << "func 2: y = " << y << endl;
  func3(x, y);
  cout << "func 3: x^2 = " << y << endl;
  func4(x, y);
  cout << "func 4: x^2 = " << y << endl;
  func5(&x, py);
  cout << "func 5: x^2 = " << *py << endl;
}
```

2. Consider the following 3 source files. Find and correct all the bugs so that the program works with

$$x = 10,$$
$$y = 50$$

printed out to the console.

```
    double x(10.0);
    double y(0.0);
    const double c = 5.0;
```

Listing 1: extern_var.cpp

```
void mult(const double& x, const double c) {
  y = c * x;
}
```

Listing 2: extern_func.cpp

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

int main()
{
  {
    double x;
    x = 100.0;
  }

  cout << "x = " << x << endl;
  mult(x, c);
  cout << "y = " << y << endl;
}
```

Listing 3: extern_main.cpp

3. Consider the following 3 files including 2 header files and one main program source file. What is wrong here with the code?

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

double totalArea(0.0);
double areaTri(const double& h, const double& b);
double areaRec(const double& w, const double& h);
void printTotal() {
  cout << "The total area = " << totalArea << endl;
}
```

Listing 4: func_declare.h

```
#include "func_declare.h"

double areaTri(const double& h, const double& b) {
  return 0.5 * h * b;
}

double areaRec(const double& w, const double& h) {
  return w * h;
}
```

Listing 5: func_define.h

```
#include <iostream>
#include "func_declare.h"
#include "func_define.h"
using namespace std;
```

```
5
6   int main()
7   {
8     double h = 5.0, b = 2.0;
9     double w = 3.0;
10    double triArea, recArea;
11    triArea = areaTri(h, b);
12    recArea = areaRec(w, h);
13    totalArea = triArea + recArea;
14
15    cout << "Triangle␣area:␣" << triArea << endl;
16    cout << "Rectangle␣area:␣" << recArea << endl;
17    printTotal();
18  }
```

Listing 6: func_main.cpp

4. Let's modify the previous `func_define.h` to `func_define.cpp` and `func_main.cpp` as below. Does the correction in the previous exercise still work in this case? Why is that? How to correct it?

```
1   #include "func_declare.h"
2
3   double areaTri(const double& h, const double& b) {
4     return 0.5 * h * b;
5   }
6
7   double areaRec(const double& w, const double& h) {
8     return w * h;
9   }
```

Listing 7: func_define.cpp

```
1   #include <iostream>
2   #include "func_declare.h"
3   using namespace std;
4
5   int main()
6   {
7     double h = 5.0, b = 2.0;
8     double w = 3.0;
9     double triArea, recArea;
10    triArea = areaTri(h, b);
11    recArea = areaRec(w, h);
12    totalArea = triArea + recArea;
13
14    cout << "Triangle␣area:␣" << triArea << endl;
15    cout << "Rectangle␣area:␣" << recArea << endl;
16    printTotal();
17  }
```

Listing 8: func_main.cpp

2 Programming

Write the C++ code which satisfies the following requirements and such that the main program below should work.

Programming requirements:

1. *Declare* all functions relating to *vector* operations in a separate header file and name it `vector.h`.
2. *Define* all functions relating to *vector* operations in a separate source file and name it `vector.cpp` with `vector.h` included.
3. *Declare* all functions relating to *matrix* operations in a separate header file and name it `matrix.h`.
4. *Define* all functions relating to *matrix* operations in a separate source file and name it `matrix.cpp` with `matrix.h` included.
5. Remember to use header guards for the header files.
6. Remember to use `assert` to check whether the sizes of the vectors and matrices involving in the calculations are correct.
7. For all functions, use the interface and input arguments given in the `main()` function in Listing 9 below.
8. Use the following `main` file to test your implementation. Use MATLAB to double check the output results.

```
#include <iostream>

#include "constants.h"
#include "vector.h"
#include "matrix.h"

using namespace std;

int main()
{
  //== matrices
  int numRowsA, numColsA;
  int numRowsB, numColsB;
  int numRowsC, numColsC;
  int numRowsD, numColsD;
  int numRowsE, numColsE;
  int** A, ** B, ** C, ** D, **E;

  // from inputs
  cout << "Input numRowsA and numColsA:" << endl;
  cin >> numRowsA >> numColsA;
  cout << "Input numRowsB and numColsB:" << endl;
  cin >> numRowsB >> numColsB;

  // compute the size of C = A + B

  // compute the size of D and E such that
  // E = A * D

  // allocation
  A = allocate(numRowsA, numColsA);
  B = allocate(numRowsB, numColsB);
  C = allocate(numRowsC, numColsC);
  D = allocate(numRowsD, numColsD);
  E = allocate(numRowsE, numColsE);

  // initialization
```

```
38  ramdom(A, numRowsA, numColsA);
39  ramdom(A, numRowsA, numColsA);
40  ramdom(D, numRowsD, numColsD);
41
42  // set C and E to be zero
43  zeros(C, numRowsC, numColsC);
44  zeros(E, numRowsE, numColsE);
45
46  // linear algebra
47  // C = A + B
48  add(A, numRowsA, numColsA,
49    B, numRowsB, numColsB,
50    C, numRowsC, numColsC);
51
52  // E = A * D
53  mult(A, numRowsA, numColsA,
54    B, numRowsB, numColsB,
55    D, numRowsD, numColsD);
56
57  //=== vectors
58  int sizeV1, sizeV2, sizeW, sizeT;
59  int* v1, *v2, *w, *t;
60
61  // compute size v1, v2, and w such that
62  // w = v1 + v2, t = E*w
63
64  // allocate v1, v2, w, and t
65  v1 = allocate(sizeV1);
66  v2 = allocate(sizeV2);
67  w = allocate(sizeW);
68  t = allocate(sizeT);
69
70  // initialization
71  random(v1, sizeV1);
72  random(v2, sizeV2);
73
74  // set zero
75  zeros(w, sizeW);
76  zeros(t, sizeT);
77
78  // linear algebra
79  // w = v1 + v2
80  add(v1, sizeV1,
81      v2, sizeV2,
82      w, sizeW);
83
84  // t = E*w
85  mult(E, numRowsE, numColsE,
86     w, sizeW,
87     t, sizeT);
88
89  //=== print results
90  print(v1, sizeV1, "v1");
91  print(v2, sizeV2, "v2");
92  print(w, size_w, "w␣=␣v1␣+␣v2");
93  print(A, numRowsA, numColsA, "A");
94  print(B, numRowsB, numColsB, "B");
95  print(C, numRowsC, numColsC, "C␣=␣A␣+␣B");
96  print(D, numRowsD, numColsD, "D");
97  print(E, numRowsE, numColsE, "E␣=␣A␣*␣D");
98  print(t, sizeT, "t␣=␣E␣*␣w");
99
100 // deallocation
101 deallocate(v1);
102 deallocate(v2);
103 deallocate(w);
```

```
104   deallocate(t);
105   deallocate(A, numRowsA);
106   deallocate(B, numRowsB);
107   deallocate(C, numRowsC);
108   deallocate(D, numRowsD);
109   deallocate(E, numRowsE);
110
111   return 0;
112 }
```

Listing 9: func_main.cpp

3 Some hints on programming

- **Memory allocation/deallocation for vectors**

```
1  #include <iostream>
2  using namespace std;
3
4  double* allocate(const int& numCols)
5  {
6    double* v;
7    v = new double[numCols];
8    return v;
9  }
10
11 void deallocate(double* v)
12 {
13   delete[] v;    // for arrays
14 }
```

- **Print vectors:**

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  void print(const double* v, const int& size, string name)
6  {
7    cout << "Vector " << name << ": " << endl;
8    for (int j = 0; j < size; ++j)
9      cout << v[j] << ", ";
10   cout << endl;
11 }
```

- **Add vectors:**

```
1  void add(const int& length, const double* v1, const double* v2, double* w)
2  {
3    for (int i = 0; i < length; ++i)
4      w[i] = v1[i] + v2[i];
5  }
```