16.216: ECE Application Programming

Practice Problems: 2-D Arrays
Solution

1. What does each of the following programs print?

a.
```
int main() {
   int arr[3][3] = {{1, 3, 5},
                    {2, 4, 6},
                    {-1, -5, -8}};
   int i, j;
   for (i = 0; i < 3; i++) {
      for (j = 0; j < 3; j++) {
         arr[i][j] = arr[i][j] + arr[j][i];
         printf("%d   ", arr[i][j]);
      }
      printf("\n");
   }
   return 0;
}
```

Solution:
```
2   5   4
7   8   1
3   -4   -16
```

b.
```
int main() {
    int x[3][5];
    int v = 0;
    int i, j;

    for (j = 0; j < 5; j++) {
       for (i = 0; i < 3; i++) {
          x[i][j] = v;
          v++;
       }
    }

    for (i = 0; i < 3; i++) {
       for (j = 0; j < 5; j++) {
          printf("%d ", x[i][j]);
       }
    }
    return 0;
 }
```

Solution:

```
0 3 6 9 12 1 4 7 10 13 2 5 8 11 14
```

1 (cont.) What does the following program print? Assume the user enters the following:

```
1 1 5.5
1 1 2.2
3 2 -0.2
5 0 -5
2 -2 3
0 1 2
5 1 1.7
```

c.
```
int main() {
   double arr[10][3];
   double v;
   int i, j;

   for (i = 0; i < 10; i++) {
      for (j = 0; j < 3; j++) {
         arr[i][j] = 0;
      }
   }

   do {
      scanf("%d %d %lf", &i, &j, &v);
      if ((i > -1) && (j > -1)) {
         arr[i][j] += v;
      }
   } while ((i > -1) && (j > -1));

   for (i = 0; i < 10; i++) {
      for (j = 0; j < 3; j++) {
         printf("%lf ", arr[i][j]);
      }
      printf("\n");
   }
   return 0;
}
```

Solution:

```
0.000000 0.000000 0.000000
0.000000 7.700000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 -0.200000
0.000000 0.000000 0.000000
-5.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
```

2. Write a function to do each of the following tasks:

a. `add4by4()`: Given three 4x4 arrays of double-precision variables—`dest`, `m1`, and `m2`—add the corresponding elements of `m1` and `m2` and store the results in `dest`. For example, if `m1[0][0] = 3` and `m2[0][0] = 4.5`, then `dest[0][0]` should equal 3+4.5 = 7.5.

Solution:

```
void add4by4(double dest[][4], double m1[][4], double m2[][4]) {
     int i, j;

     for (i = 0; i < 4; i++) {
          for (j = 0; j < 4; j++) {
               dest[i][j] = m1[i][j] + m2[i][j];
          }
     }
}
```

b. `MACrow()`: Given a 2-D integer array, `x[][3]`, a 1-D integer array, `y[]`, and an integer, `n`, that represents the number of rows in `x` and elements in `y`, perform a multiply-accumulate operation (MAC) on each row of `x` and store the result in `y`. For example, if row 0 of `x` contains the values `{2, 3, 4}`, then `y[0] = 2 x 3 + 4 = 6 + 4 = 10`.

Solution:

```
void MACrow(int x[][3], int y[], int n) {
     int i;

     for (i = 0; i < n; i++)
          y[i] = x[i][0] * x[i][1] + x[i][2];
}
```

6 (cont.)

c. `countZeroColumns()`: Given a 2-D integer array, `arr[][10]`, and the number of rows in the array, `n`, return the number of columns in this array in which most of the values are 0. For example, if the array has three rows, you count the number of columns that contain two or more zeroes.

```
int countZeroColumns(int arr[][10], int n) {
     int totalCols = 0;       // Total # of columns with
                              // mostly zero values
     int zeroesPerCol;        // # of zeroes in a given column
     int i, j;

     // For each column, go through all rows and count # zeroes
     // If majority of column is zeroes, increment totalCols
     for (j = 0; j < 10; j++) {
          zeroesPerCol = 0;
          for (i = 0; i < n; i++) {
               if (arr[i][j] == 0)
                    zeroesPerCol++;
          }

          if (zeroesPerCol > (n / 2))   // Majority of
               totalCols++;             // column is zeroes
     }

     return totalCols;
}
```