

# Tutorial 8

Md. Tanvir Hassan

**Email:** [md.hassan@mail.concordia.ca](mailto:md.hassan@mail.concordia.ca)

# Exercise 1

**Use a one-dimensional array to solve the following problem.**

A company pays its salespeople on a commission basis. The salespeople each receive \$200 per week plus 9 percent of their gross sales for that week. For example, a salesperson who grosses \$5000 in sales in a week receives \$200 plus 9 percent of \$5000, or a total of \$650. Write a program (using an array of counters) that determines how many of the salespeople earned salaries in each of the following ranges (assume that each salesperson's salary is truncated to an integer amount):

- a) \$200–299
- b) \$300–399
- c) \$400–499
- d) \$500–599
- e) \$600–699
- f) \$700–799
- g) \$800–899
- h) \$900–999
- i) \$1000 and over

# Exercise 2

**(Duplicate Elimination with array)** Use a one-dimensional array to solve the following problem. Read in 20 numbers, each of which is between 10 and 100, inclusive. As each number is read, validate it and store it in the array only if it isn't a duplicate of a number already read. After reading all the values, display only the unique values that the user entered. Provide for the "worst case" in which all 20 numbers are different. Use the smallest possible array to solve this problem.

# Exercise 3

**(Dice Rolling)** Write a program that simulates the rolling of two dice. The sum of the two values should then be calculated. [Note: Each die can show an integer value from 1 to 6, so the sum of the two values will vary from 2 to 12, with 7 being the most frequent sum and 2 and 12 being the least frequent sums.] Figure 7.26 shows the 36 possible combinations of the two dice. Your program should roll the two dice 36,000 times. Use a one-dimensional array to tally the numbers of times each possible sum appears. Print the results in a tabular format. Also, determine if the totals are reasonable

(i.e., there are six ways to roll a 7, so approximately one-sixth of all the rolls should be 7).

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12