**Lab Exercise 1 — Rolling Dice**

**Note:** If you have difficulties with these problems, you can look up “Template & Tips” for help. If you don't need it, that's great.

Write a program that simulates the rolling of two dice. The program should call rand to roll the first die, and should call rand again to roll the second die. The sum of the two values should then be calculated. [Note: Each die has 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.1 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 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).

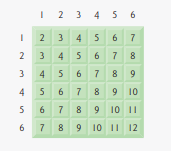
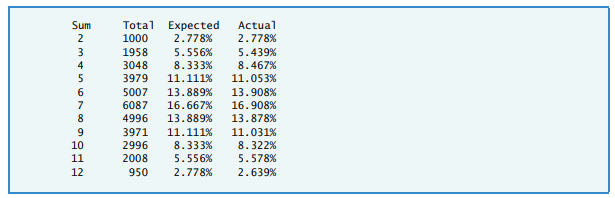


Fig.1 | 36 possible outcomes of rolling two dice

**Sample Output:**



**Lab Exercise 2 — Array**

**Task 1:**

Create an array with N integers. Input the values of each element. Then output the size of the array and values of each element.

**Sample input:**

6

1 2 3 4 5 6

**Sample output:**

6

1 2 3 4 5 6

**Task 2:**

Modify the program to use function *input* to input the value to an array, and function *output* to output an array. Sample input and output are same with Task 1.

**Task 3:**

Add functions to the program.

1. Function *search* finds the position of “key value” in the array. If not found, return -1.
2. Function *minimum* gives the minimum value of the array.
3. Function *maximum* gives the maximum value of the array.
4. Function *minipos* gives the position of minimum value in the array.
5. Function *maxipos* gives the position of maximum value in the array.

Write some codes to test your functions. Maybe you can try replacing keyboard input with random numbers

**Task 4:**

Add functions to the program.

1. Function *sum* sums up the values of every elements.
2. Function *average* calculates the average value of elements, calling function *sum* in the code.
3. Function *even* counts the number of even numbers.
4. Function *odd* counts the number of odd numbers.

Write some codes to test your functions.

**Task 5:**

Add functions to the program.

1. Function *split* divides the array into two arrays: odd array and even array.

**Sample input:**

6

1 2 3 4 5 6

**Sample output:**

3

1 3 5

3

2 4 6

1. Function *combine* merges two arrays into one array, one array following another.

**Sample input:**

3

6 5 4

3

1 2 3

**Sample output:**

6

6 5 4 1 2 3

1. (\*)Function *combineordered* merges two ordered arrays into one array, The merged array is still in order.

**Sample input:**

3

1 2 3

3

2 5 6

**Sample output:**

6

1 2 2 3 5 6

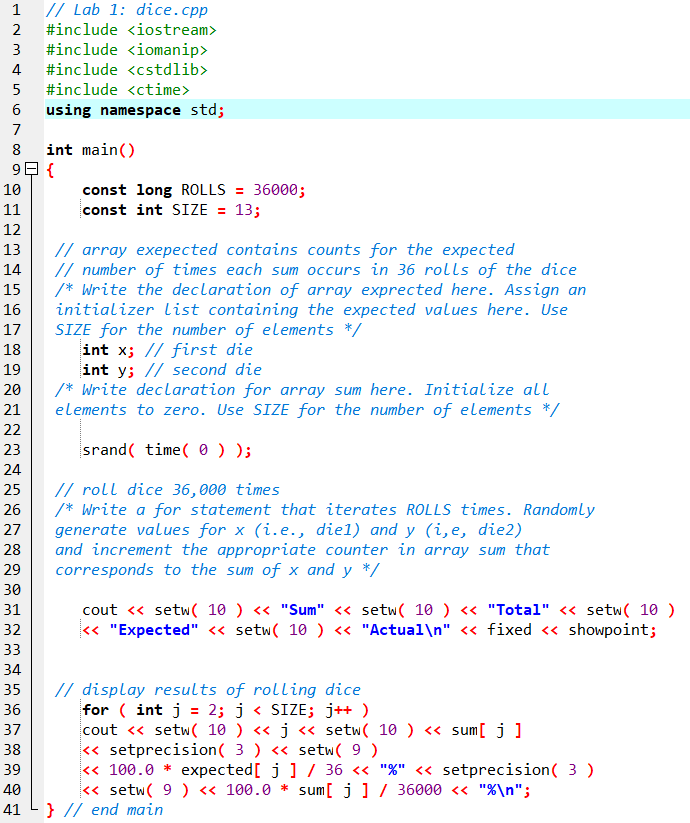
**(\*) Task 6:**

1. Try to create your own ARRAY and use all the code just now.

2. Add more interesting features to your ARRAY.

## Template & Tips

**Lab Exercise 1 — Rolling Dice**



**Problem-Solving Tips**

1. Remember that array subscripts always begin with zero. This is also true for each dimension of a multiple-subscripted array (which this lab does not use).

2. The actual percentage is the likelihood, based on the results of your program, that a dice roll produced a certain result. In other words, if you roll the dice 36,000 times the actual percentage will be the (number of times a result occurred / 36000) \* 100.

3. The expected percentage is the statistical probability that a dice roll will produce a certain result. This can be calculated from the diagram “36 possible outcomes of rolling two dice,” shown in the problem description. For example, there is only one combination that will produce the sum of 2 and there are 36 total combinations that occur with equal likelihood. Therefore, the expected percentage of rolling a 2 is 1/36 or 2.778%.