4. Write a program that reads in an array of characters. You may assume that there are fewer than 50 entries in the array. Your program determines how many entries are used. The output is to be a two-column list. The first column is a list of vowels in the English alphabet; the second column is the count of the number of occurrences of each vowel. The list should be sorted on entries in the first column in alphabetical order.

   For the array values

   b r e a d , b u t t e r, a n d j a m o n y o u r p l a t e

   the output should be

Vowel	*Count*
a	*4*
e	*3*
i	*0*
o	*2*
u	*2*

8. The birthday paradox is that there is a surprisingly high probability that two people in the same room happen to share the same birthday. By birthday, we mean the same day of the year (ignoring leap years), but not the exact birthday including the birth year or time of day. Write a program that approximates the probability that two people in the same room have the same birthday, for 2 to 50 people in the room.

   The program should use simulation to approximate the answer. Over many trials (say, 5000), randomly assign birthdays to everyone in the room. Count up the number of times at least two people have the same birthday on a trial, and then divide by the number of trials to get an estimated probability that two people share the same birthday for a given room size.

   Your output should look something like the following. It won't be exactly the same due to the random numbers:

   For 2 people, the probability of two birthdays is about 0.002
   For 3 people, the probability of two birthdays is about 0.0082
   For 4 people, the probability of two birthdays is about 0.0163
   …
   For 49 people, the probability of two birthdays is about 0.9654
   For 50 people, the probability of two birthdays is about 0.969

10. 10. Write a program that displays the list of flavors available at an ice-cream parlor with a unique code for each flavor, as shown below.

Code	*Flavor*
1	*Chocolate*
2	*Vanilla*
3	*Strawberry*
4	*Raspberry*
5	*Butterscotch*

The program then displays the code and quantity available for customers to purchase as shown below.

1 20

2 15

3 15

4 20

5 20

The program should read the user's choice of flavors and quantity and display the products remaining in the stock, until the stock is exhausted or the user signals that the program should end. Finally, the user's cart should be displayed as shown below

Chocolate 3

Strawberry 1

Raspberry 1

14. You have collected reviews from four movie reviewers where the reviewers are numbered 0-3. Each reviewer has rated six movies where the movies are numbered 100-105. The ratings range from 1 (terrible) to 5 (excellent). The reviews are shown in the following table:

	100	101	102	103	104	105
0	3	1	5	2	1	5
1	4	2	1	4	2	4
2	3	1	2	4	4	1
3	5	1	4	2	4	2

Write a program that stores this data using a 2D array. Based on this information your program should allow the user to enter ratings for any three movies. The program should then find the reviewer whose ratings most closely match the ratings input by the user. It should then predict the user's interest in the other movies by outputting the ratings by the reviewer for the movies that were not rated by the user. Use the Cartesian distance as the metric to determine how close the reviewer's movie ratings are to the ratings input by the user. This technique is a simple version of the nearest neighbor classification algorithm.

For example, if the user inputs a rating of 5 for movie 102, 2 for movie 104, and 5 for movie 105, then the closest match is reviewer 0 with a distance of *sqrt((5-5)^2 + (2-1)^2 + (5-5)^2) = 1*. The program would then predict a rating of 3 for movie 100, a rating of 1 for movie 101, and a rating of 2 for movie 103.

17. Programming Project 2.12 asked you to explore Benford's Law. An easier way to write the program is to use an array to store the digit counts. That is, count[0] might store the number of times 0 is the first digit (if that is possible in your data set), count[1] might store the number of times 1 is the first digit, and so forth. Redo Programming Project 2.12 using arrays.

※ 2.12 This problem is based on a "Nifty Assignment" by Steve Wolfman (http://nifty.stanford.edu/2006/wolfman-pretid). Consider lists of numbers from real-life data sources, for example, a list containing the number of students enrolled in different course sections, the number of comments posted for different Facebook status updates, the number of books in

different library holdings, the number of votes per precinct, etc. It might seem like the leading digit of each number in the list should be 1-9 with an equally likely probability. However, Benford's Law states that the leading digit is 1 about 30% of the time and drops with larger digits. The leading digit is 9 only about 5% of the time.

Write a program that tests Benford's Law. Collect a list of at least one hundred numbers from some real-life data source and enter them into a text file. Your program should loop through the list of numbers and count how many times 1 is the first digit, 2 is the first digit, etc. For each digit output the percentage it appears as the first digit.