tity on hand to a file. (Don't save the `next` pointer; it won't be valid once the program terminates.) As it reads parts from a file, the `r` operation will rebuild the list one node at a time.

11. Write a program that reads a date from the command line and displays it in the following form:

    ```
    September 13, 2010
    ```

    Allow the user to enter the date as either `9-13-2010` or `9/13/2010`; you may assume that there are no spaces in the date. Print an error message if the date doesn't have one of the specified forms. *Hint:* Use `sscanf` to extract the month, day, and year from the command-line argument.

12. Modify Programming Project 2 from Chapter 3 so that the program reads a series of items from a file and displays the data in columns. Each line of the file will have the following form:

    *item*, *price*, *mm*/*dd*/*yyyy*

    For example, suppose that the file contains the following lines:

    ```
    583,13.5,10/24/2005
    3912,599.99,7/27/2008
    ```

    The output of the program should have the following appearance:

    ```
    Item            Unit            Purchase
                    Price           Date
    583             $  13.50        10/24/2005
    3912            $ 599.99        7/27/2008
    ```

    Have the program obtain the file name from the command line.

13. Modify Programming Project 8 from Chapter 5 so that the program obtains departure and arrival times from a file named `flights.dat`. Each line of the file will contain a departure time followed by an arrival time, with one or more spaces separating the two. Times will be expressed using the 24-hour clock. For example, here's what `flights.dat` might look like if it contained the flight information listed in the original project:

    ```
    8:00 10:16
    9:43 11:52
    11:19 13:31
    12:47 15:00
    14:00 16:08
    15:45 17:55
    19:00 21:20
    21:45 23:58
    ```

14. Modify Programming Project 15 from Chapter 8 so that the program prompts the user to enter the name of a file containing the message to be encrypted:

    ```
    Enter name of file to be encrypted: message.txt
    Enter shift amount (1-25): 3
    ```

    The program then writes the encrypted message to a file with the same name but an added extension of `.enc`. In this example, the original file name is `message.txt`, so the encrypted message will be stored in a file named `message.txt.enc`. There's no limit on the size of the file to be encrypted or on the length of each line in the file.

15. Modify the `justify` program of Section 15.3 so that it reads from one text file and writes to another. Have the program obtain the names of both files from the command line.