Mathematica Hw VI

Claus Ernst/Huanjing Wang Math/Cs 371 Spring 2016

Due: Wednesday, March 16 before class.

This assignment must be turned in electronically AND as a hard copy. The electronic copy must be submitted through blackboard **prior** to your class.

General Requirements - read before you start working

Form groups of two. Each member of the group must understand and be able to explain (in detail) all of the code turned in. We allow you to work in groups so that you have someone with whom you can talk about your work in detail. This is **not** meant for group members to only do half of the work.

Each group turns in one (stapled) printout at the beginning of class and submits an electronic copy (one file) **prior** to class.

Your electronic and hard copy must include:

a) The names of both group members on the print out (printed from the file, not added by hand). The name of both group members must be included in the name of the electronic file. For example if we (Claus and Huanjing) are a team then the file name should be **ErnstWang371Hw6.nb**. This will help to identify whose file we are looking at and it will also avoid having several different files with the same name.

b) A solution to each problem, which includes the code, as well as a couple of executions of your code which demonstrate that the code is (most likely) correct. Select different types of input for your example runs, to show that different parts of your code work correctly.

c) Your solutions must be well-documented *Mathematica* code, that is, you must use comments, meaningful variable names, proper indentations, hide local variables (if any are needed), etc. The comments in the code should be such that taken by themselves they describe the algorithm you used.

Note: Throughout this assignment, the lists of arguments for functions maybe given to clarify the order and meaning of the arguments; they do not follow Mathematica syntax. However, your solutions must follow the given order/structure of arguments as specified by the assignment. If you do deviate from this our test programs will not run.

Note: You can write any number of helper functions for each problem.

1. Cipher

A mixed-alphabet cipher is created by first writing a keyword followed by the remaining letters of the alphabet and then using the substitution (or cipher) text. For example, if the keyword is django, the cipher text alphabet would be:

djangobcefhiklmqrstuvwxyz

So, a is replaced with d, b is replaced with j, c is replaced with a, and so on. As an example, the piece of text

the sheik of araby
would then be encoded as
tcg scgeh mo dradjy

You can assume the text to be encoded only contains lower case letters and spaces.

(1) Write a function **encode[str, key, blocksize]** where *str* is the text to be encoded, *key* is the keyword, and *blocksize* is the block length to display the cipher text. This function returns the cipher text. Output the cipher text in blocks of length *blocksize* (text only, position of white space is displayed as is).

For example, **encode[“the sheik of araby”, “django”, 5];** returns
tcg scgeh mo dradjy
(do not display this message on the screen)
and displays the followings on the screen
tcgsc gehmo drdjy

(2) Write a function decode[str, key] where *str* is the text to be decoded and *key* is the keyword. This function display and returns original text.

2. Files

Bill’s Computer Parts is a discount computer store in Bowling Green KY. They keep their records in three files:

customers.txt, which contains the following information about their customers:
- cid (four digits)
- cname
- discount (YES or NO)

In this file, there is one line per customer, the items are separated by space in each line. The eligibility for discount indicates whether or not the customer will receive 10% discount in an order. If a customer is eligible for discount, the 10% discount applies to all orders made by this customer.

products.txt, which contains the following information about their products:
- pid (three digits)
- pname (no space between characters)
- price

In this file, there is one line per product, the items are separated by space in each line.

orders.txt, which keeps track of the order information at the store. Each order is represented on a separate line as follows:
- cid
- pid

• qty (quantity)

In this file, there is one line per order, the items are separated by space in each line.

You may assume that there is no space in customer's name and product's name, and customer id and product id listed in the order file will exist in the customers and products files.

Write a function **summary[ ]** which reads the information from the three files and outputs a detailed order information list and an order summary. The detailed order information should consist of customer name, product name, price, quantity, order amount, and discount amount. The list of the detailed order information should be consistent (in the same order) with the list in "orders.txt". The information should be displayed on the screen and be written to a file named "orderdetails.txt". This function should also generate an order summary. Each entry in the summary should consist of customer name, and the total order amount. The entries should be displayed on the screen in descending order based on the total order amount. The entries should also be written to a file named "ordersummary.txt".
The format of the two outputs files are listed below.

orderdetails.txt

```
        Order Details

CName   PName    Price  Qty Total   Discount
Austin  Printer  200.0  2   360.0   40.0
Alex    Toner    40.0   3   120.0   0.0
Ashby   Computer 500.0  1   450.0   50.0
Austin  Computer 500.0  1   450.0   50.0
```

ordersummary.txt

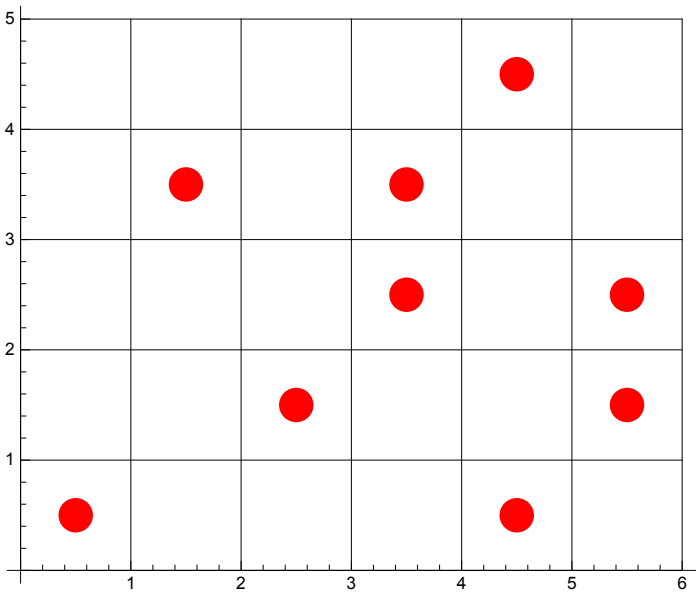
```
Order Summary

CName   Total
Austin  810.0
Ashby   450.0
Alex    120.0
```

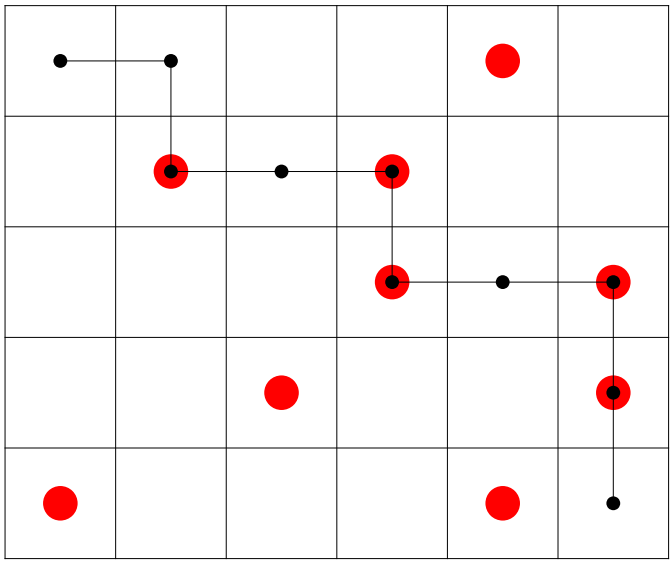
3. Maximum Number of Coins

Several coins are placed in cells of an n by m board, no more than one coin per cell. A robot, located in the upper left cell of the board, need to collect as many of the coins as possible and bring them to the bottom right cell. On each step, the robot can move either one cell to the right or one cell down from its current location. When the robot visits a cell with a coin, it always picks up that coin.

(1) Write a function **initializeBoard[n, m, k]** where n is the number of rows and m is the number of columns of the board. Randomly place k coins in the board. Displays and returns the board. For example, **initializeBoard[5, 6, 9]** may generates the following board.



(2) Write a function **maxNumberofCoins[n,m]** where n is the number of rows and m is the number of columns of the board. **maxNumberofCoins** first generates a random integer k which is less than half of n×m, and then calls **initializeBoard[n,m,k]** to generate the initial board with coins. This function displays one of the optimal paths to get the maximum number of coins and returns the maximum number of coins the robot can pick up (Note: there maybe more than one optimal path.) You will not get full credit if you search all possible paths.



Extra credit:

(1) There may be more than one path to collect the maximum number of coins. Draw all possible optimal paths and display each path differently.

(2) Any type of animation will result in extra credit. Describe clearly what animation you make in docu-

mentation (comments). Below is an example to show two different optimal paths.