4.7 Problems

1) Write a program that prompts the user to enter two strings and prints out the match score to indicate how similar the words are. If the two strings are different lengths, then you should print a score of 0. Otherwise, if the two strings are the same length, the score printed should be the number of characters in corresponding locations that are the same. For example, the match score between "home" and "host" should be 2 and the match score between "paper" and "caper" should be 4.

2) Write a program that asks the user to enter 10 words and prints out the word that comes first alphabetically.

3) Write a program that simulates using a Magic Eight Ball. Your program should have a list of pre-programmed responses to questions stored in a list of strings. Ask the user to enter a question and the present the user with a randomly chosen response from the list you have created.

4) Write a program that reads in a text file of test scores and prints out a histogram of the scores. The file format is as follows: the first line of the file contains a single positive integer, *n*, representing the number of test scores. The following *n* lines will contain one test score each. Each of these test scores will be a non-negative integer less than or equal to 100. The histogram should have one row on it for each distinct test score in the file, followed by one star for each test score of that value. For example, if the tests scores in the file were 55, 80, 80, 95, 95, 95 and 98, the output to the screen should look like:

```
55*
80**
95***
98*
```

5) Write a simulation where the user collects packs of baseball cards. Each pack contains a set of 10 cards, where each card is numbered from 1 to 100. In the simulation, have the user continue to buy packs of cards until she has collected all 100 distinct cards. Each set must contain distinct cards, but two different sets may contain the same card.

6) Generate a set containing each positive integer less than 1000 divisible by 15 and a second set containing each positive integer less than 1000 divisible by 21. Create a set of integers that is divisible by either value, both values and exactly one value. Print out the contents of each of these resultant sets.

7) Write a program that allows the user to add telephone book entries, delete people from a telephone book, allows the user to change the number of an entry, and allows the user to look up a person’s phone number. Put in the appropriate error checking.

8) Write a program that asks the user to enter a list of censored words, along with their approved replacements. (For example, “jerk” might be replaced with “mean person”.) Read in a sentence from the user and write a modified sentence to the screen where each censored word in the sentence was replaced with the approved replacement. In the given example, the sentence “he is such a jerk” would be converted to “he is such a mean person”.

9) Rewrite program #8 so that it reads in the list of censored and replacement words from the file "censordictionary.txt", reads the sentence to convert from the input file "message.txt" and outputs the converted message to the file "safemessage.txt". Create appropriate file formats for all three files.

10) The following problem is taken from a programming contest. The input for the problem is to be read from the file "idnum.in" and the output is to be printed to the screen. The exact file format is given as well as some sample input and the corresponding output for those cases.

The Problem

There are many new summer camps starting up at UCF. As new departments try to start up their summer camps, word got around that there was a computer science summer camp at UCF that was already established. One of the tools that these other summer camps need is a tool to create identification numbers for all the campers. These summer camps have kindly asked Arup to create a computer program to automate the process of allocating identification numbers. Naturally, Arup has decided that this would be an excellent exercise for his BHCSIers. For each summer camp in question, identification numbers will be given in the order that students sign up for the camp. Each camp will have a minimum number for which to start their student identification numbers. Each subsequent number will be generated by adding 11 to the previously assigned number so that each number is sufficiently spaced from the others. After assigning all the identification numbers, your program will need to print an alphabetized list of each student paired with his/her identification number.

The Input

The first line of the input file will consist of a single integer *n* representing how summer camps for which you are assigning identification numbers. For each summer camp, the first line of input will contain one integer *k* $(0 < k \leq 200)$, representing the number of students in that summer camp. The second line of input for each summer camp will contain a single positive integer, *minID*, which represents the minimum identification number for all the students in the camp. (This is the number that will be given to the first student to sign up for the camp.) The following *k* lines will each contain a single name consisting of only 1 to 19 upper case letters. These names are the names of all the students in the class, in the order in which they signed up for the camp. The names within a single summer camp are guaranteed to be unique.

The Output

For every summer camp, the first line will be of the following format:

Summer camp #m:

where m is the number of the summer camp starting with 1.

The following *k* lines should list each student in the summer camp in alphabetical order and his/her identification number, separated by a space.

Put a blank line of output between the output for each summer camp.

Sample Input

```
2
8
2000
SARAH
LISA
ARUP
DAN
JOHN
ALEX
CONNER
BRIAN
10
100001
JACK
ISABELLA
HAROLD
GARY
FRAN
EMILY
DANIELLE
CAROL
BOB
ADAM
```

Sample Output

```
Summer camp #1:
ALEX 2055
ARUP 2022
BRIAN 2077
CONNER 2066
DAN 2033
JOHN 2044
LISA 2011
SARAH 2000

Summer camp #2:
ADAM 100100
BOB 100089
CAROL 100078
DANIELLE 100067
EMILY 100056
FRAN 100045
GARY 100034
HAROLD 100023
ISABELLA 100012
JACK 100001
```