COP 2930 - Introduction to Computing
Input Files - Suggested Exercises

Objective:

1. Practice writing programs that read input from files.

1) Consider a file called "apscores.txt", where the first line of the file as a single positive integer, *n*, representing the number of AP test scores in the file. The following *n* lines each have a single score, an integer in between 1 and 5 inclusive. Read in the data from the file, compute the following results and print them to the screen:

a) The number of students who achieved each score from 1 to 5.
b) The average score on the exam for the group of students.

2) Read input from any text file and count the number of a's, b's, c's, …, z's and output a tally of how many of each letter was in the input file. Recall that ord(letter) will give the letter's Ascii value and chr(number) will convert the integer number to the corresponding character with the Ascii value number.

3) 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
```