**Algorithm Practice**

**Project Description & Goal**

The goal of this project is to get some more practice developing interesting algorithms. We will be using samples from codequest as our basis.

**Project Specifications**

* Create a program that solves the 5 problems listed at the bottom of this document.
* Format the program however you feel comfortable doing so.
  + Menu with options 1-5 for each of the problems.
  + Linear program that just solves one after the other printing output.
  + Etc.

**Project Learning Objectives**

* Get more practice with complex algorithms.
* Learn how to consider edge cases that might not be present in the given test data.

**Project Demonstrated Competencies**

1. Problem # 1 developed and working as expected.
2. Problem # 2 developed and working as expected.
3. Problem # 3 developed and working as expected.
4. Problem # 4 developed and working as expected.
5. Problem # 5 developed and working as expected.

**Rubric**

|  |  |  |
| --- | --- | --- |
|  | **Description of perfect implementation** | **Score** |
| Competency #1 | Output is correct for any input, including input with multiple sets of the same type (as in the example input) as well as any input with an invalid type such as “ABC=5” | \_\_\_  12.5 |
| Competency #2 | Output is correct for any input, including saying that the input is invalid if it has any non-integer data in its string. | \_\_\_  12.5 |
| Competency #3 | Your program will read in the decoded key and use it to decode the messages in the input. | \_\_\_  12.5 |
| Competency #4 | Correct value of each factorial is output. | \_\_\_  12.5 |
| Competency #5 | The student’s grades are correctly reported to the console. | \_\_\_  +15 |

**Input File:** Prob01.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using Console.WriteLine())

**Introduction**

Little Johnny would like help determining the value of the money in his piggy bank. He will provide you with the number of quarters, dimes, pennies, nickels, and half dollars he has in the bank. You need to tell him the total value. And no cheating little Johnny! The input file will contain a list of coins found in the piggy bank and the number of those types of coins. Coins will be in no particular order. The task is to determine the amount of money represented by the group of coins in the bank.

**Program Input**

The file Prob01.in.txt will contain a list of items and their quantities. There will only be one item and quantity pair per line, although duplicate items are allowed on different lines. Items and quantities will be separated by an equal sign with no spaces. There may also be non-coin items in the list that you should ignore. Valid coin names are: HALFDOLLAR, QUARTER, DIME, NICKEL, and PENNY.

**Example Input:**

QUARTER=31

DIME=5

NICKEL=2

DIME=7

HALFDOLLAR=0

PENNY=157

**Program Output**

Your program’s output should display the total value of the money in the bank in standard US currency format, using a dollar sign before the dollar amount and two decimal places for the number of cents.

**Example Output:**

$10.62

**Input File:** Prob02.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using Console.WriteLine())

**Introduction**

Write a program that will take a list of integers and determine whether they are in ascending, descending or random order. Integers can be positive or negative. If you encounter a non-integer value, then the entire line should be considered invalid input.

**Program Input**

The file Prob02.in.txt will contain some number of lines of characters separated by spaces. Each line should be treated as a new input set.

**Example Input:**

5 6 9 13 17 15

1 2 3 4 5

-1 -2 -3 -4 -5

1 2 3 4 =5

**Program Output**

Your program should output one of the following messages:

* The numbers are in random order
* The numbers are in ascending order
* The numbers are in descending order
* The input was invalid

**Example Output:**

The numbers are in random order

The numbers are in ascending order

The numbers are in descending order

The input was invalid

**Input File:** Prob03.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using Console.WriteLine())

**Introduction**

You have been asked to decode a secret message. A General Decoder takes a set of input strings. The first string is the English alphabet encoded in the order needed to decode the subsequent strings of numbers.

For example, the sequence:

18-5-1-4 20-8-9-19 13-5-19-19-1-7-5

Can be decoded by applying the alphabet order:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

To read:

READ THIS MESSAGE

Your task is to write a program that will take as inputs a decoding key and an encoded message, and output the decoded message.

**Program Input**

The file Prob03.in.txt will contain the alphabet decoding key on the first line followed by any number of lines of encoded messages. Numbers correspond to the position of the letter they represent in the decoding key, and are separated by either a dash or a space. Dashes are used to separate letters within a word, and spaces denote word boundaries. There will be no numbers less than 1 or greater than 26.

**Example Input:**

AEIOUYBCDFGHJKLMNPQRSTVWXZ

22-24-4 20-4-1-9-21 9-3-23-2-20-11-2-9 3-17 1 24-4-4-9 1-17-9 3

3 22-4-4-14 22-12-2 4-17-2 15-2-21-21 22-20-1-23-2-15-2-9 7-6

1-17-9 22-12-1-22 12-1-21 16-1-9-2 1-15-15 22-12-2 9-3-10-10-2-20-2-17-8-2

**Program Output**

Your program should output the decoded message.

**Example Output:**

TWO ROADS DIVERGED IN A WOOD AND I

I TOOK THE ONE LESS TRAVELED BY

AND THAT HAS MADE ALL THE DIFFERENCE

**Input File:** Prob04.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using Console.WriteLine())

**Introduction**

A factorial is the product of a given positive integer (n) multiplied by all lesser positive integers and is represented as n! For example:

4! = 4 x 3 x 2 x 1 = 24

Your task is to write a program which will calculate factorials for numbers from 1 to 50.

**Program Input**

The file Prob04.in.txt will contain a list of numbers, one per line. All numbers will be integers between 1 and 12, inclusive. There will be no non-integer input.

**Example Input:**

5

10

**Program Output**

Your program should output the values of the factorials for the numbers encountered in the input file in the same order, one per line.

**Example Output:**

120

3628800

**Input File:** Prob05.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using Console.WriteLine())

**Introduction**

Scantrons! You know, sheets of paper used to mark your answers on multiple choice tests. You can select A, B, C, D, or E by filling in that little bubble. In this exercise, you will be writing a program to grade students’ scantron papers so the teacher doesn’t have to do it manually.

**Program Input**

The file Prob05.in.txt will contain three sections:

1. The first line of the file will be an integer number telling you how many points each question is worth.
2. The next set of lines will be the key to the test. Blank columns are indicated by an "O", and the bubbled in column is indicated by an equal sign. Columns are separated by a single space. Multiple answers may be bubbled in.
3. The student response section will consist of any number of students’ answers. Each student section will contain the following:
   1. The first line of each student’s section will be in the format "STUDENT #", where the pound sign will be an integer number indicating the student’s ID number. The word student and the ID number will be separated by a single space.
   2. The next set of lines will be the student’s responses to the questions on the test. Questions are answered in order, and only an exact match to the key will receive credit for the question. There is no partial credit.

**Example Input:**

10

O O O O =

O = O O O

O O O O =

= O O O O

= O O O O

O = O O O

O O = O O

O O O = O

O = O O O

O O O O =

STUDENT 12

O = O O =

O = O O O

O O O O =

= O O O O

= O O O O

O = O O O

O O = O O

O O O = O

O = O O O

O O O O =

STUDENT 13

O O O O =

O = O O O

O O O O =

= O O O O

O O O = O

O = O O O

O O = O O

O O O = O

O = O O O

O = O O O

**Program Output**

Your program should output each student’s id number along with their grade in the following format:

STUDENT #: Grade

**Example Output:**

STUDENT 12: 90

STUDENT 13: 80