(1) (10) Print the factorial of a given number. Recall $n! = n(n-1)\cdots 2 \cdot 1$ and $0! = 1$.

(2) Given a list of length N consisting of `int`s, return if a user-supplied `int` occurs in the list.

(3) (10) A phone number $(abc)def.ghij$ consists of an area code abc along with regional phone number $def.ghij$.

Assume Dallas has an area code of 469, Tallahassee has an area code of 850, Birmingham has an area code of 205, and Nashville has an area code of 615. Given a list of 10-digit numbers, return the quantity of phone numbers from each area in the order

Dallas, Tallahassee, Birmingham, Nashville, Other

(4) (10) A semester grade is composed of three components with particular weights: homework ($\overline{H}$, 20%), quizzes ($\overline{Q}$, 30%), and exams ($\overline{E}$, 50%). Given a tuple $(\overline{H}, \overline{Q}, \overline{E})$, return the semester grade.

(5) (15) A list `[x1, y1, x2, y2]` of four floats is given where the values correspond to the vertices of a diagonal of the rectangle whose sides are parallel to the coordinate axes. Return the area of the rectangle.

(6) (15) A triangle is right if its sides obey the Pythagorean theorem $a^2 + b^2 = c^2$. Given a list of six `ints`

$$\texttt{[x\_1, y\_1, x\_2, y\_2, x\_3, y\_3]} \equiv [(x_1, y_1), (x_2, y_2), (x_3, y_3)]$$

corresponding to the vertices of a triangle, return whether the triangle is right.

```
[2, 3, 3, 2, 6, 6] => False
[3, 2, 14, 2, 3, 62] => True
```

(7) (20) A 16-digit credit card number

$$a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}a_{11}a_{12}a_{13}a_{14}a_{15}a_{16}$$

consists of 15 digits $a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}a_{11}a_{12}a_{13}a_{14}a_{15}$ followed by a check digit a_{16}. Set F to be the number of digits in the odd positions that exceed 4. Then the check digit a_{16} is the smallest positive integer such that

$$2(a_1 + a_3 + \cdots + a_{15}) + (a_2 + a_4 + \cdots + a_{16}) + F$$

is a multiple of 10. Given a 15-digit number, return its check-digit so that a valid credit card number has been created.

```
314159265358979 => 6
```

(8) (17) Return a given binary number in its decimal form.

(9) (15) Suppose you are given two *sorted* lists L_1, L_2 each containing five not-necessarily-distinct positive integers . Return the list L containing all of the elements from L_1 and L_2 such that L is sorted in increasing order.

(10) (12) Suppose an integer $N \geq 2$ and a list L of size $N-1$ such that L is a subset of $\{1, 2, 3, \ldots N\}$ are given. Return the integer k that is missing from $\{1, 2, 3, \ldots N\}$ that L does not possess.

```
5; 1, 3, 4, 5 => 2
5; 1, 2, 3, 4 => 5
```

(11) (15) In a list of `floats`, a *peak* is a value that is greater than all of the entries to its right. Return the peaks from a given list. Assume the rightmost character is always a peak.

(12) (15) Suppose a positive integer N and a list L of distinct positive integers are given. Return the number of pairs from L that sum to N.

(13) (25) Given a string S of alphabetic characters, return a string S' wherein any adjacent characters of S that are the same are eliminated. Your program should recurse so that all duplicates are eliminated in each step.

```
abbc => ac
abbcbbcdbccbd => accdbbd => add => a
```

(14) (15) The Collatz Sequence is formed by starting with a fixed positive integer n. If n is even, return $n/2$. If n is odd, return $3n+1$. This process is repeated until 1 is reached. A seed of 22 yields

$$22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

we get a *cycle* of length 16. Given a pair of integers i, j, return the longest cycle from the set of seeds $\{i, i+1, \ldots, j\}$

```
1, 10 => 20
100, 200 => 125
```

(15) (15) Given a string consisting of grouping characters

`( ) [ ] { }`

in *any* order, return if they could form a correct grouping of a mathematical expression.

(16) (20) Given a string containing only alphabetic characters and spaces, return the reverse of it. (You may not use negative slices, etc.)

```
Bob is your uncle => uncle your is Bob
```

(17) (10) Given a string that contains a sentence, return the quantity of each uppercase letter, each lowercase letter, and the quantity of non-alphabetic characters.

(18) (12) Suppose you are given two strings S_1, S_2 consisting of alphabetic characters. If you can rotate S_1 by two places to form S_2, then we call the strings *2-rotated.* Return whether two given strings are 2-rotated or not.

```
math, thma => True
laptop, oplapt => True
pencils, cilspen => False
```

(19) (20) Suppose a circle is given by its center and radius. Given two circles, return whether they intersect in zero, one, or two points.

(20) (20) Suppose an positive odd integer N is given. Return a $*$-diamond with center of length N.

```
Sample:

                  1:     *                     5:     *
                                                     ***
                  3:     *                          *****
                        ***                          ***
                         *                            *
```

(21) (25) Suppose the score for a team in a game is always a [non-negative] multiple of 11. A team's mascot does a series of push-ups after every score equal to the total team score. Hence the first touchdown garners 11 push-ups, the second 22, and so on. Given a final score of N, return the *total* number of push-ups the mascot performed during the game.

(22) (35) Given a fraction in the form (n, d), return a list of tuples

$$[(1, d_1), (1, d_2), \ldots, (1, d_k)]$$

such that

$$\frac{n}{d} = \frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3} + \cdots + \frac{1}{d_k} \qquad \text{and} \qquad d_1 < d_2 < \cdots < d_k$$

(23) (30) A message is encrypted via a shift cipher by replacing each character with another character in the alphabet (a-z, space with 0 as space and a as 1, and so on) a fixed shift k from the original with rollover at the extreme. (E.g., "ab z" shifted right 2 is "cdba") One way to attack an encrypted message is to map the words in the message to a dictionary.

Implement a decryption method that breaks messages encrypted with a shift cipher. Given a list of strings (dictionary of plain words) and an encrypted string, use the provided dictionary to map each lines words to words in the provided dictionary. If successful, the program prints the shift k used to encrypt the original messaged followed by the decoded message . If decryption fails, return -1.

```
['alabama', 'blue', 'birmingham', 'color', 'story', 'street', 'world']
zcdlatsvt iwpi lxaa rwpcvt ndjg ldgas
=> 15, 'knowledge that will change your world'
```

(24) (32) Suppose L is a list of `int`s. Return the greatest sum of consecutive entries. We allow sums of one element.

```
1  2  3  4  =>  10
1  −2  3  4  =>  7
1  2  −2  4  =>  5
```

(25) (20) Return whether a given string is *valid* or not if we define a string to be *valid* provided

- its characters are elements of the character set

$$\{\texttt{a},\texttt{b},\texttt{c},\ldots,\texttt{z},\texttt{0},\texttt{1},\texttt{2},\ldots,\texttt{9},\texttt{\_}\},$$

- its first character is a letter
- no two adjacent characters nor the last character may be an underscore

(26) (28) Return the longest palindrome in a given string of alphabetic characters. If the palindrome is not unique, return the first occurring palindrome.

(27) (32) Some virtual machines are based on the notion of a run-time stack to hold data. This is like a stack of plates where data may only be removed from or added to the top. Code is of the form:

```
PUSH  1
PUSH  2
ADD
RESULT
=>  3                 (stack output)
```

PUSH 1 pushes a 1, PUSH 2 pushes a 2 and ADD removes the top 2 values on the stack, adds them and pushes the result (so the 1 and 2 would be replaced by 3). RESULT is a special operation that will display the current value at the top of the stack.

Write an interpreter for such stack machine code. In addition to PUSH which will always be followed by an unsigned integer, implement ADD, SUB, MUL, DIV, and RESULT operations. The operators may be abbreviated to A, S, M, D, and R. All operands will be integers. You may assume integer division when implementing DIV.

```
Sample:

Enter virtual machine commands:   |   Enter virtual machine commands:
P 1                               |   P 1
P 2                               |   P 2
A                                 |   P 3
P 3                               |   P 4
M                                 |   P 5
P 1                               |   M
S                                 |   A
P 2                               |   P 6
D                                 |   S
R                                 |   A
=> 4                              |   A
                                  |   P 10
                                  |   D
                                  |   R
                                  |   => 2
```

(28) (37) Given a list of positive `int`s, return the largest possible `int` formed via concatenation of the provided numbers.

```
3, 30, 43, 1 => 433301
```

(29) (38) Given a string S and a positive integer $N < \texttt{len}(N)$, break S into rows of length N. We perform two operations on the resulting rectangular array of characters: swap two rows (r, i, j) or swap two columns (c, i, j). For a provided list of swaps $[(r/c, i_1, j_1), (r/c, i_2, j_2), \ldots, ((r/c), i_k, j_k)]$, return the string S' after the swaps have been performed.

```
                             12345
bob is your uncle          1 bob_i    s_you    suyo_
N = 5                   => 2 s_you => bob_i => bib_o => suyo_bib_orcun_l___e
[(r,1,2),(c,2,5)]          3 r_unc    r_unc    rcun_
                           4 le___    le___    l___e
(Underscores indicate spaces.)
```

(30) (40) Suppose you are given a list of positive ints of which a single, unique element is the greatest and each list value indicates the height of stackable cups placed in the particular row position. (E.g., [3,1,2] means a stack of three cups, then one cup, and finally two cups all in a row) Water is poured into the stack with greatest height (and filling only its topmost stacked cup), and water overflows "nicely" into all remaining cups. If a valley occurs (3, 1, 2), then water will fill the cup containing the valley, overflow the valley, and then begin to fill adjacent higher cups. Return the number of cups and total valley areas filled for a particular list.

```
[1, 2, 1]; [1, 3, 1]; [1, 1, 2;] [1, 2, 3] => 3
[2,1,3,1,4] => 8
[3,1,2,1,4] => 10
```

(31) (45) The Hamming distance between two non-empty strings of equal length is the number of characters in which they differ taking position into account

$d(\text{drat}, \text{drop}) = 2 \quad d(\text{bob}, \text{rob}) = 1 \quad d(\text{abc}, \text{xyz}) = 0 \quad d(\text{abc}, \text{abc}) = 0 \quad d(\text{abcd}, \text{dcba}) = 4$

We say two words are *close* if their Hamming distance is 1, and given a collection of words $\{w_1, w_2, \ldots, w_k\}$ we say they form a *word ladder* if w_i and w_{i+1} are close for $1 \leq i \leq k-1$.

Given a collection of words, return if a word ladder exists in the collection.

```
['cord', 'ward', 'cold', 'warm', 'card'] => True
['stone', 'shone', 'aloof', 'chine', 'chins', 'coins'] => False
['coins', 'chine', 'chins', 'shone', 'shine', 'stone'] => True
```

(32) (45) Suppose the seven houses $A, B, C, \ldots, G$ are connected via sidewalks:

Sidewalk	Distance (ft)
AB	24
AC	71
BE	59
BD	103
CE	134
CF	101
CG	169
EF	65
FG	141

Given a pair of houses H_1, H_2, return the length of the shortest path between them.

(33) (50) You are given a dictionary for an unknown language. The characters of the alphabet are the same, but it appears that the ordering may be different. Your task is to write a program that can determine the ordering of the alphabet given a list of words in sorted order.

The input will be a "sorted" list of strings. You may assume that the characters received will be uppercase. Not all the characters A-Z will necessarily be used in the dictionary.

Your program must output the alphabet used to write these strings, in "sorted" order. You may assume that the sorted strings are sufficient to determine a unique sorted alphabet.

```
Sample:

A, B, C => A, B, C
AB, AC, BE, BD, BA => E, D, A, B, C
```