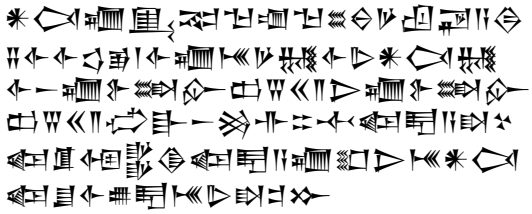
First Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Second Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Abstract

This paper announces the discovery of the use of neural nets almost 4,000 years before their use in the modern era. Newly discovered tablets preserve a perceptron used for calculating the numbers on Plimpton 322, the most important object in the history of mathematics. The native programming language used by the ancient Babylonian "cuneogrammers" uses sexagesimal numbering leading to some "weirdness".



1 Introduction

The history of math is long, but the history of programming is longer. Cuneiform, arguably the first writing in the world, is known for its sheep receipts and beer ration lists as well as complaints about substandard copper (Oppenheim, 1954) and women complaining at each other (Matuszak, 2020). This article adds neural network programming to that vaunted list of human achievements. A set of newly discovered cuneiform tablets preserve the mechanism for performing simple neural network calculations. These methods, it seems, were used to calculate the lengths of triangles; a well known example of this exercise is preserved on the tablet known as Plimpton 322[1]. It is remarkable that no historian of math or cuneiform scholar has ever consider this possibility. This paper covers the background, a description of the cuneogramming language, and includes a facsimile copy of the most important tablet as an appendix.

Unfortunately, the hardware required to execute these programs (i.e. a living Babylonian mathematician) has not been adequately preserved, but we have managed to write a Python library which emulates it.[2] The assumption is that these calculations were done by hand in their copious free time between inventing the wheel and the concept of zero. While the actual output of these tablets is relatively simple by modern standards, the implications of this discovery are profound. Future work will explore how these techniques could have been used in the realm of astronomical calculation and elucidate the full extent of Babylonian computational prowess.

2 Description of the Language

Programs in  (EME.ŠID.A "language of counting") follow a tabular structure with three main sections: (i) a header, denoted by (DUB "tablet"), (ii) a sequence of instructions, and (iii) a colophon detailing the tablet's authorship. Each instruction spans four columns, which we have taken to calling the *arguments*, *opcode*, *destination*, and *line number*. These columns are usually tab-separated, though in a few documents they are TAB-separated (using the cuneiform sign TAB). Instructions are grouped into blocks by means of horizontal lines.

Arguments An instruction's arguments may be numbers, register addresses, or a combination of the two. Numbers are encoded following standard Babylonian conventions, with  denoting the radix point which separates the integer part from a following fraction. There is

[1] An important and real description of this interesting object is found in Robson (2002).

[2] github.com/MrLogarithm/emeszida

an explicit representation for zero (𒁹), making these tablets some of the earliest unambigiuous examples of the mathematical concept of zero.

A register address is denoted by the phrase NIN_2.KAS_7 *n*-KAM 𒊭𒁕*n*𒌋 ("thing.account *n*-th", "the item in the *n*-th place"), where *n* is any number. 𒊭𒁕 expressions can be nested to perform a kind of pointer dereferencing: for example, if 𒊭𒁕𒁹𒌋 (register 1) contains the value 𒐖 (3), and 𒊭𒁕𒐖𒌋 (register 3) contains the value 𒐗 (7), then 𒊭𒁕𒊭𒁕𒁹𒌋𒌋 will evaluate to 7 (the value in the register pointed to by 𒊭𒁕𒁹𒌋).

If an instruction has multiple arguments, these must be delimited by a wide space (distinct from the short space used to separate groups of digits within a number), or by one of the phrases *a-na* 𒀀𒈾 "to" or *i-na* 𒄿𒈾 "from". By convention, the choice of delimiter depends on the instruction's opcode (see below), with multiplication operations preferring space delimiters, addition preferring 𒀀𒈾, and subtraction preferring 𒄿𒈾. There is no mechanism to enforce these conventions, but we recommend following them because 𒍑𒂗𒈫𒀀 is hard enough to read at the best of times.

Destination In most cases, the destination column of an instruction will be a register address where the output is to be stored, e.g. 𒊭𒁕𒁹𒌋. Some control-flow instructions (see below) instead expect the destination to be a line number. 𒋼𒌅 SUD ("distant, remote") can be used as a null destination, for statements which produce no output.

Opcodes Each instruction has a single opcode belonging to the following vocabulary:

- 𒋫𒄭 DAḪ.ḪA, "add"
- 𒁍𒌨 BA.ZI, "tear out"
- 𒀀𒊏 A.RA , "multiply"
- 𒅆 IGI, "reciprocal"
- 𒈫 ME, "to be"
- 𒅆𒄑𒌑 *ta-mar*, "you will see"
- 𒂗𒈾 NIGIN.NA, "start again"
- 𒂗𒄿𒋼𒍑 TUKUM.BI DIRIG, "if it exceeds"
- 𒂗𒄿𒋼𒋛 TUKUM.BI SIG, "if it it is weak"

The first three of these are binary operators for addition, subtraction, and multiplication respectively. The subtraction operator deserves special attention for its use in constructions of the following shape, which appear hundreds of times throughout the Babylonian programming corpus:

𒁹𒁹𒐖𒌋𒄿𒈾𒁹 𒁍𒌨 𒊭𒁕𒁹𒌋 𒁹

This instruction subtracts 𒁹𒁹𒐖𒌋 from zero and stores the result in 𒊭𒁕𒁹𒌋. This effectively negates the first argument, and seems to have been the primary way by which Babylonian cuneogrammers represented negative numbers, as their primitive and archaic notation otherwise lacked a means to encode such values. This curious practice gives definitive proof that the invention of negative numbers occurred centuries earlier than previously believed.

The language does not appear to have any kind of binary division operator. Rather, a unary 𒅆 operator was used to find the reciprocal of the denominator, which was then multiplied by the numerator using the binary 𒀀𒊏 operator.

𒈫 is a unary assignment operator which stores a value in a destination register. 𒅆𒄑𒌑 functions as a unary print operator.

𒍑𒂗𒈫𒀀 includes three types of control-flow instructions. 𒂗𒈾 functions like GOTO, and jumps the program counter to the specified line number. 𒂗𒄿𒋼𒍑 functions like the x86 jz instruction, and jumps to the specified line number if and only if its argument is zero. 𒂗𒄿𒋼𒋛 is similar, and jumps if the argument is greater than zero.

Line Numbers Every line of an 𒍑𒂗𒈫𒀀 program ends in a mandatory line number. However, these numbers are not generally sequential, and need not even be distinct. For example, most lines of the perceptron tablet are labeled with the number 𒁹 (zero); only lines that are the destination of some control-flow instruction receive non-zero identifiers.

2.1 Fractional Indexing

Both line numbers and register addresses in 𒍑𒂗𒈫𒀀 can have fractional parts. The original scribes seem to have exploited this fact

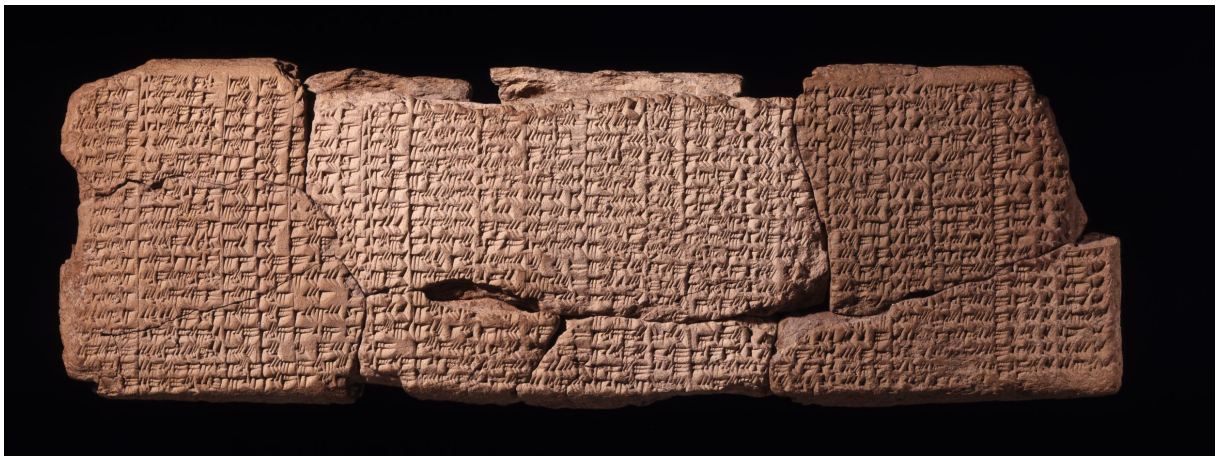


Figure 1: BM 34580, courtesy of the Trustees of the British Museum, CC BY-NC-SA 4.0

to establish non-overlapping "namespaces" for the different parts of their code. For example, in the perceptron tablet, all of the model parameters are stored in adresses with integer part 1; the program inputs all have integer part 2; the matrix multiplication subroutine uses addresses with integer part 3; and so on. The fractional parts of register addresses also appear to follow some standard conventions, with the X;0 register typically storing a subroutine's return address, while X;1 onward were used for its arguments.

The perceptron tablet also uses fractional register addresses to perform a kind of multi-dimensional array indexing. As an example, the first layer of the perceptron has a 50×2 weight matrix, and this is stored in registers 1;0,0,0 through 1;0,49,1. The integer part of these addresses denotes the "data" portion of memory; the first digit after the radix point identifies this as the 0th model parameter; and the second and third digits can be treated as a pair of indices ranging from 0–49 and 0–1 respectively. To access a specific element in this matrix, the scribes use repeated division by 60 to implement a kind of "bit-shift" instruction, in order to shift integer indices into the correct positions after the radix point. By adding the bit-shifted element indices (e.g. 0;0,4,7 for the element in row 5, column 8) to a pointer to the top corner of the matrix (1;0,0,0) they obtain the address of the desired element (1;0,4,7).

Notably, this practice limits the size of their model parameters to at most 60×60, as for larger values the addresses would carry over to higher digits and thus begin to overwrite one another. This limitation may explain why AI never made waves in Babylonian society, as their models were all too small to be truly revolutionary.

3 Description of the Texts

The cuneogramming corpus contains numerous fragments implementing recognizable procedures such as bit-shifting, populating an array, computing dot products, and so on. However, only a single text is known to have been preserved in its entirety.[3] Spanning close to 1700 lines, this impressive text is divided into five sections implementing what modern readers will immediately recognize to be a multilayer perception. The first section straightforwadly defines a matrix-multiplication subroutine. This is called by a subroutine defined in section two, which applies each layer of the perceptron to a given input, and applies a ReLU-style activation between each pair of layers. Section three implements the tablet's "main" method, which loads an input to memory, calls the perceptron subroutine, and prints the resulting output. Section four loads the model parameters, which appear to comprise weight matrices of sizes 50×2, 25×50, and 1×25, plus bias vectors of sizes 50, 25, and 1 respectively. The final section lists pairs of inputs, whose values (incredibly!) correspond to the second and third columns of Plimpton 322. Interestingly, this section very closely re-

[3] All of the known texts are reproduced in github.com/MrLogarithm/emeszida/tree/main/programs, and the long text is reproduced in facsimile in the appendix of this work.

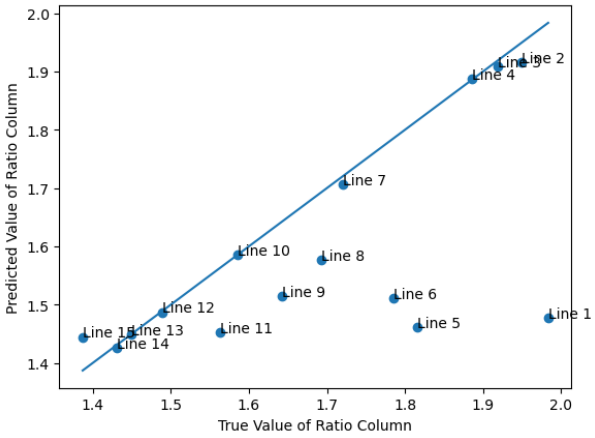


Figure 2: Outputs from the perceptron tablet vs. true value of each line from column 1 of Plimpton 322. If a point falls upon the line, the model output for that line exactly equals the value on Plimpton 322.

sembles the tables of parameters found in later astronomical calculations (see Figure 1).

When the program is executed, it produces a single numeric output for each input pair. These outputs correspond remarkably closely to the values in the first column of Plimpton 322, as demonstrated in Figure 2. The correspondence is not perfect, however, and the values which should match lines 1, 5, and 6 of Plimpton 322 are significantly larger than expected. This implies that, although the code on this tablet is clearly *related* to Plimpton 322, it could not have been used to directly populate the table in that text. Perhaps the outputs from this model were refined in some later step to produce the more exact ratios in the Plimpton text, or perhaps the Babylonians were disillusioned by the imprecision of their machine learning models and simply abandoned them for tried-and-true manual methods. Given how miniscule the cuneogramming corpus is relative to the larger body of Babylonian administrative writing, we lean towards the latter explanation.

4 Implications and Future Work

This completely rewrites the history of modern computing and artificial intelligence. Instead of focusing on figures such as Ada Lovelace and Grace Hopper we should be looking at pioneers thousands of years earlier like Enheduana the world's first known author (Helle, 2023) and now the world's first known programmer.

Other ancient corpora which have resisted decipherment, and which boast a similar numeric component, may represent additional examples of ancient programming traditions (Kelley et al., 2022).

Acknowledgments

The search for these tablets and our effort to understand them was inspired by Ramsey Nasser's قلب programming language (https://github.com/nasser/—).

References

Sophus Helle. 2023. *Enheduana: the complete poems of the world's first author*. Yale University Press, New Haven, CT.

Kathryn Kelley, Logan Born, M. Willis Monroe, and Anoop Sarkar. 2022. Image-aware language modeling for proto-elamite. *Lingue e linguaggio*, (2):261–294.

Jana Matuszak. 2020. *"Und du, du bist eine Frau?!": Editio princeps und Analyse des sumerischen Streitgesprächs 'Zwei Frauen B'*. De Gruyter.

A. L. Oppenheim. 1954. The seafaring merchants of Ur. *Journal of the American Oriental Society*, 74(1):6–17.

Eleanor Robson. 2002. Words and pictures: New light on Plimpton 322. *The American Mathematical Monthly*, 109(2):105–120.

A Appendix

The following pages reproduce the perceptron tablet in its entirety.

[illegible]

[illegible]

[illegible]

𓀀𓀁𓀂𓀃𓀄𓀅𓀆𓀇𓀈𓀉𓀊𓀋𓀌𓀍𓀎𓀏𓀐𓀑𓀒𓀓𓀔𓀕𓀖𓀗𓀘𓀙𓀚𓀛𓀜𓀝𓀞𓀟𓀠𓀡𓀢𓀣𓀤𓀥𓀦𓀧𓀨𓀩𓀪𓀫𓀬𓀭𓀮𓀯𓀰𓀱𓀲𓀳𓀴𓀵𓀶𓀷𓀸𓀹𓀺𓀻𓀼𓀽𓀾𓀿𓁀𓁁𓁂𓁃𓁄𓁅𓁆𓁇𓁈𓁉𓁊𓁋𓁌𓁍𓁎𓁏𓁐𓁑𓁒𓁓𓁔𓁕𓁖𓁗𓁘𓁙𓁚𓁛𓁜𓁝𓁞𓁟𓁠𓁡𓁢𓁣𓁤𓁥𓁦𓁧𓁨𓁩𓁪𓁫𓁬𓁭𓁮𓁯𓁰𓁱𓁲𓁳𓁴𓁵𓁶𓁷𓁸𓁹𓁺𓁻𓁼𓁽𓁾𓁿𓂀𓂁𓂂𓂃𓂄𓂅𓂆𓂇𓂈𓂉𓂊𓂋𓂌𓂍𓂎𓂏𓂐𓂑𓂒𓂓𓂔𓂕𓂖𓂗𓂘𓂙𓂚𓂛𓂜𓂝𓂞𓂟𓂠𓂡𓂢𓂣𓂤𓂥𓂦𓂧𓂨𓂩𓂪𓂫𓂬𓂭𓂮𓂯𓂰𓂱𓂲𓂳𓂴𓂵𓂶𓂷𓂸𓂹𓂺𓂻𓂼𓂽𓂾𓂿𓃀𓃁𓃂𓃃𓃄𓃅𓃆𓃇𓃈𓃉𓃊𓃋𓃌𓃍𓃎𓃏𓃐𓃑𓃒𓃓𓃔𓃕𓃖𓃗𓃘𓃙𓃚𓃛𓃜𓃝𓃞𓃟𓃠𓃡𓃢𓃣𓃤𓃥𓃦𓃧𓃨𓃩𓃪𓃫𓃬𓃭𓃮𓃯𓃰𓃱𓃲𓃳𓃴𓃵𓃶𓃷𓃸𓃹𓃺𓃻𓃼𓃽𓃾𓃿𓄀𓄁𓄂𓄃𓄄𓄅𓄆𓄇𓄈𓄉𓄊𓄋𓄌𓄍𓄎𓄏𓄐𓄑𓄒𓄓𓄔𓄕𓄖𓄗𓄘𓄙𓄚𓄛𓄜𓄝𓄞𓄟𓄠𓄡𓄢𓄣𓄤𓄥𓄦𓄧𓄨𓄩𓄪𓄫𓄬𓄭𓄮𓄯𓄰𓄱𓄲𓄳𓄴𓄵𓄶𓄷𓄸𓄹𓄺𓄻𓄼𓄽𓄾𓄿𓅀𓅁𓅂𓅃𓅄𓅅𓅆𓅇𓅈𓅉𓅊𓅋𓅌𓅍𓅎𓅏𓅐𓅑𓅒𓅓𓅔𓅕𓅖𓅗𓅘𓅙𓅚𓅛𓅜𓅝𓅞𓅟𓅠𓅡𓅢𓅣𓅤𓅥𓅦𓅧𓅨𓅩𓅪𓅫𓅬𓅭𓅮𓅯𓅰𓅱𓅲𓅳𓅴𓅵𓅶𓅷𓅸𓅹𓅺𓅻𓅼𓅽𓅾𓅿𓆀𓆁𓆂𓆃𓆄𓆅𓆆𓆇𓆈𓆉𓆊𓆋𓆌𓆍𓆎𓆏𓆐𓆑𓆒𓆓𓆔𓆕𓆖𓆗𓆘𓆙𓆚𓆛𓆜𓆝𓆞𓆟𓆠𓆡𓆢𓆣𓆤𓆥𓆦𓆧𓆨𓆩𓆪𓆫𓆬𓆭𓆮𓆯𓆰𓆱𓆲𓆳𓆴𓆵𓆶𓆷𓆸𓆹𓆺𓆻𓆼𓆽𓆾𓆿𓇀𓇁𓇂𓇃𓇄𓇅𓇆𓇇𓇈𓇉𓇊𓇋𓇌𓇍𓇎𓇏𓇐𓇑𓇒𓇓𓇔𓇕𓇖𓇗𓇘𓇙𓇚𓇛𓇜𓇝𓇞𓇟𓇠𓇡𓇢𓇣𓇤𓇥𓇦𓇧𓇨𓇩𓇪𓇫𓇬𓇭𓇮𓇯𓇰𓇱𓇲𓇳𓇴𓇵𓇶𓇷𓇸𓇹𓇺𓇻𓇼𓇽𓇾𓇿𓈀𓈁𓈂𓈃𓈄𓈅𓈆𓈇𓈈𓈉𓈊𓈋𓈌𓈍𓈎𓈏𓈐𓈑𓈒𓈓𓈔𓈕𓈖𓈗𓈘𓈙𓈚𓈛𓈜𓈝𓈞𓈟𓈠𓈡𓈢𓈣𓈤𓈥𓈦𓈧𓈨𓈩𓈪𓈫𓈬𓈭𓈮𓈯𓈰𓈱𓈲𓈳𓈴𓈵𓈶𓈷𓈸𓈹𓈺𓈻𓈼𓈽𓈾𓈿𓉀𓉁𓉂𓉃𓉄𓉅𓉆𓉇𓉈𓉉𓉊𓉋𓉌𓉍𓉎𓉏𓉐𓉑𓉒𓉓𓉔𓉕𓉖𓉗𓉘𓉙𓉚𓉛𓉜𓉝𓉞𓉟𓉠𓉡𓉢𓉣𓉤𓉥𓉦𓉧𓉨𓉩𓉪𓉫𓉬𓉭𓉮𓉯𓉰𓉱𓉲𓉳𓉴𓉵𓉶𓉷𓉸𓉹𓉺𓉻𓉼𓉽𓉾𓉿𓊀𓊁𓊂𓊃𓊄𓊅𓊆𓊇𓊈𓊉𓊊𓊋𓊌𓊍𓊎𓊏𓊐𓊑𓊒𓊓𓊔𓊕𓊖𓊗𓊘𓊙𓊚𓊛𓊜𓊝𓊞𓊟𓊠𓊡𓊢𓊣𓊤𓊥𓊦𓊧𓊨𓊩𓊪𓊫𓊬𓊭𓊮𓊯𓊰𓊱𓊲𓊳𓊴𓊵𓊶𓊷𓊸𓊹𓊺𓊻𓊼𓊽𓊾𓊿𓋀𓋁𓋂𓋃𓋄𓋅𓋆𓋇𓋈𓋉𓋊𓋋𓋌𓋍𓋎𓋏𓋐𓋑𓋒𓋓𓋔𓋕𓋖𓋗𓋘𓋙𓋚𓋛𓋜𓋝𓋞𓋟𓋠𓋡𓋢𓋣𓋤𓋥𓋦𓋧𓋨𓋩𓋪𓋫𓋬𓋭𓋮𓋯𓋰𓋱𓋲𓋳𓋴𓋵𓋶𓋷𓋸𓋹𓋺𓋻𓋼𓋽𓋾𓋿𓌀𓌁𓌂𓌃𓌄𓌅𓌆𓌇𓌈𓌉𓌊𓌋𓌌𓌍𓌎𓌏𓌐𓌑𓌒𓌓𓌔𓌕𓌖𓌗𓌘𓌙𓌚𓌛𓌜𓌝𓌞𓌟𓌠𓌡𓌢𓌣𓌤𓌥𓌦𓌧𓌨𓌩𓌪𓌫𓌬𓌭𓌮𓌯𓌰𓌱𓌲𓌳𓌴𓌵𓌶𓌷𓌸𓌹𓌺𓌻𓌼𓌽𓌾𓌿𓍀𓍁𓍂𓍃𓍄𓍅𓍆𓍇𓍈𓍉𓍊𓍋𓍌𓍍𓍎𓍏𓍐𓍑𓍒𓍓𓍔𓍕𓍖𓍗𓍘𓍙𓍚𓍛𓍜𓍝𓍞𓍟𓍠𓍡𓍢𓍣𓍤𓍥𓍦𓍧𓍨𓍩𓍪𓍫𓍬𓍭𓍮𓍯𓍰𓍱𓍲𓍳𓍴𓍵𓍶𓍷𓍸𓍹𓍺𓍻𓍼𓍽𓍾𓍿𓎀𓎁𓎂𓎃𓎄𓎅𓎆𓎇𓎈𓎉𓎊𓎋𓎌𓎍𓎎𓎏𓎐𓎑𓎒𓎓𓎔𓎕𓎖𓎗𓎘𓎙𓎚𓎛𓎜𓎝𓎞𓎟𓎠𓎡𓎢𓎣𓎤𓎥𓎦𓎧𓎨𓎩𓎪𓎫𓎬𓎭𓎮𓎯𓎰𓎱𓎲𓎳𓎴𓎵𓎶𓎷𓎸𓎹𓎺𓎻𓎼𓎽𓎾𓎿𓏀𓏁𓏂𓏃𓏄𓏅𓏆𓏇𓏈𓏉𓏊𓏋𓏌𓏍𓏎𓏏𓏐𓏑𓏒𓏓𓏔𓏕𓏖𓏗𓏘𓏙𓏚𓏛𓏜𓏝𓏞𓏟𓏠𓏡𓏢𓏣𓏤𓏥𓏦𓏧𓏨𓏩𓏪𓏫𓏬𓏭𓏮𓏯𓏰𓏱𓏲𓏳𓏴𓏵𓏶𓏷𓏸𓏹𓏺𓏻𓏼𓏽𓏾𓏿

𒑲𒑰𒌍 𒌍𒐈 𒐊 𒐎 𒌍𒐍 𒐐𒐍 𒐏𒐋 𒌋 𒌋𒐍 𒎙𒐋 𒐋 𒌋𒐊 𒌍𒐈 𒐌 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒌋𒐎𒆠	𒑲
𒑲𒑰𒎙𒐊 𒌍𒐎 𒎙𒐎 𒐏𒐎 𒐐𒐊 𒐈 𒎙𒐈 𒐏𒁹 𒌋𒁹 𒎙𒐋 𒌍𒐈 𒌍𒁹 𒐏𒐊 𒌍𒐍 𒌋𒐍 𒌋𒐉 𒎙𒐉𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒎙𒆠	𒑲
𒑲𒑰𒐋 𒐏𒈫 𒌋𒐈 𒎙𒐋 𒌍𒈫 𒑲 𒌋𒁹 𒑲 𒐏𒐎 𒌍𒐎 𒐏𒈫 𒌍𒐋 𒎙𒐉 𒎙𒐊 𒌍𒐍 𒐐𒈫 𒐏𒐍	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒎙𒁹𒆠	𒑲
𒑲𒑰𒎙𒁹 𒌋𒐈 𒌍𒈫 𒑲 𒐏𒐋 𒐋 𒐋 𒌍𒐈 𒐏𒐈 𒎙𒐈 𒐐𒁹 𒌍𒐎 𒌍𒐉 𒌋𒐋 𒌋𒐎 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒎙𒈫𒆠	𒑲
𒑲𒑰𒐏𒐉 𒌋𒐊 𒌋𒐎 𒐊 𒐐𒐎 𒌋𒐉 𒎙𒁹 𒐋 𒌍 𒎙𒐍 𒐐𒐋 𒌍𒐉 𒌋𒐊 𒌋 𒐉 𒐏𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒎙𒐈𒆠	𒑲
𒑲𒑰𒌋𒈫 𒎙 𒐈 𒐏𒐈 𒐊 𒌋𒈫 𒌍𒐋 𒐋 𒈫 𒎙𒐎 𒎙𒐈 𒌋𒐍 𒐐𒐋 𒌍𒐈 𒐏𒐌 𒌍𒁹 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒎙𒐉𒆠	𒑲
𒁹𒑰𒌋𒁹 𒐈 𒐌 𒐌 𒐐𒐎 𒌍𒐉 𒐐 𒎙𒐍 𒐏𒈫 𒎙𒐋 𒌍𒐌 𒌋𒐈 𒎙𒐎 𒐐𒁹 𒎙𒁹 𒌍𒐋𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒎙𒐊𒆠	𒑲
𒑲𒑰𒌍𒐎 𒐐𒐈 𒎙𒐈 𒐐𒐉 𒌋𒐊 𒌋𒐌 𒎙𒐉 𒐏𒐌 𒐐𒐎 𒐈 𒎙𒐉 𒌋𒐋 𒎙𒁹 𒌍 𒌋𒐉 𒎙𒐉𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒎙𒐋𒆠	𒑲
𒑲𒑰𒐋 𒐊 𒐐𒐍 𒐏𒈫 𒌋 𒌋𒐍 𒌋𒐉 𒐐𒐊 𒎙𒐎 𒐏𒐈 𒌋𒐍 𒐏𒈫 𒑲 𒐋 𒐐𒐉 𒐏𒐈 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒎙𒐌𒆠	𒑲
𒑲𒑰𒎙𒈫 𒐉 𒐏𒐎 𒎙𒐋 𒐐𒐉 𒐏𒐋 𒌋𒐈 𒐏𒐎 𒐏𒁹 𒌋𒐊 𒐏𒐊 𒎙 𒐐𒐍 𒑲 𒌍 𒐏𒐈 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒎙𒐍𒆠	𒑲
𒁹𒑰𒎙𒐈 𒎙𒁹 𒐐𒐋 𒑲 𒐐𒈫 𒐌 𒐏𒐍 𒎙𒐌 𒎙𒐉 𒐐𒐈 𒌋𒐉 𒎙𒐋 𒐐𒐎 𒐏𒈫 𒐏𒐈 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒎙𒐎𒆠	𒑲
𒁹𒑰𒈫 𒐎 𒌍𒐎 𒌍𒐍 𒑲 𒑲 𒐏𒐊 𒌍𒐈 𒎙𒐈 𒑲 𒈫 𒐐𒐎 𒐏𒈫 𒐏𒐈 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒌍𒆠	𒑲
𒑲𒑰𒌋𒐋 𒑲 𒐐𒐎 𒌍𒐊 𒐐𒈫 𒑲 𒐐𒐊 𒌍𒐋 𒌍𒐍 𒐏𒐊 𒎙 𒐊 𒑲 𒐐𒐎 𒌍𒁹 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒌍𒁹𒆠	𒑲
𒑲𒑰𒌋𒐋 𒎙𒐌 𒐋 𒑲 𒌋𒐈 𒐏𒐋 𒐏𒐋 𒐐𒈫 𒐈 𒐏𒁹 𒐐𒁹 𒐐𒐊 𒌍𒈫 𒌍𒈫 𒌍𒐋 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒌍𒈫𒆠	𒑲
𒑲𒑰𒎙𒐋 𒌋𒐌 𒐏𒁹 𒌍𒐈 𒐏𒐍 𒌋𒐈 𒌍𒐋 𒈫 𒌍𒁹 𒎙𒐉 𒌋𒐉 𒐐𒐈 𒁹 𒐏𒐌 𒎙𒐋 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒌍𒐈𒆠	𒑲
𒑲𒑰𒌋𒐋 𒑲 𒐏𒁹 𒌍𒐌 𒐌 𒐐𒐊 𒐏𒐊 𒐏𒐋 𒌍𒐊 𒐊 𒎙𒐍 𒐋 𒐋 𒎙𒐋 𒐐𒈫 𒐏𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒌍𒐉𒆠	𒑲
𒁹𒑰𒌋 𒐐𒐋 𒐐𒈫 𒐏𒐋 𒎙𒐎 𒐐𒐈 𒌍𒐌 𒑲 𒌍𒐈 𒌋𒐋 𒐋 𒐐𒐊 𒎙𒐋 𒐐𒐋 𒌍𒐈 𒎙𒐋	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒌍𒐊𒆠	𒑲
𒑲𒑰𒐐𒐋 𒁹 𒌍𒐈 𒐏𒐈 𒌋𒐊 𒐐𒐋 𒎙𒐈 𒌍𒈫 𒎙𒐈 𒐐𒐎 𒌋𒐈 𒌍𒐈 𒐉 𒐐𒐈 𒐏𒐊 𒌍𒐋	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒌍𒐋𒆠	𒑲
𒑲𒑰𒌋𒐋 𒌍𒐋 𒐐 𒌍 𒐏𒐈 𒐐𒐈 𒑲 𒌍𒐎 𒐋 𒎙𒐋 𒎙𒐎 𒎙𒐎 𒑲 𒑲 𒌋𒐎 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒌍𒐌𒆠	𒑲
𒑲𒑰𒌍𒐈 𒐏𒁹 𒑲 𒐐𒐎 𒐏𒐊 𒐋 𒐏𒐋 𒐏𒁹 𒎙𒐋 𒌍𒐉 𒎙𒐌 𒐏𒈫 𒐋 𒎙 𒐎 𒌍𒐋𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒌍𒐍𒆠	𒑲
𒑲𒑰𒐐𒐉 𒌍𒐋 𒎙𒐋 𒈫 𒌋𒐌 𒌋𒈫 𒌋𒐎 𒐏𒐎 𒐎 𒌋𒁹 𒐏𒐊 𒐏𒈫 𒐐𒐉 𒐏𒐈 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒌍𒐎𒆠	𒑲
𒁹𒑰𒁹 𒐐𒁹 𒐌 𒑲 𒐐𒐊 𒌋𒐈 𒐐𒐊 𒎙𒐉 𒁹 𒐐𒐈 𒌋𒁹 𒎙𒐌 𒌋𒐋 𒐐𒐎 𒌍𒁹 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒐏𒆠	𒑲
𒑲𒑰𒐊 𒎙𒐊 𒌍𒐈 𒐉 𒌋𒐉 𒌋𒐊 𒐏𒐊 𒌋𒈫 𒐏𒐎 𒎙𒐍 𒌍𒐊 𒐋 𒎙𒐉 𒌍𒈫 𒌋𒐊 𒎙𒁹 𒌍𒐋𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒐏𒁹𒆠	𒑲
𒑲𒑰𒎙𒐋 𒑲 𒌋𒐎 𒌋𒐌 𒐐𒐎 𒌍𒐊 𒐐𒐋 𒎙𒐈 𒐏𒐊 𒎙𒁹 𒌋𒐊 𒌋 𒐐𒁹 𒎙 𒎙𒐋 𒐐𒈫 𒐏𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒐏𒈫𒆠	𒑲
𒑲𒑰𒌋𒐊 𒐉 𒌍𒁹 𒐋 𒎙 𒐐𒐊 𒌋𒐊 𒐐𒈫 𒌋𒐊 𒑲 𒌋𒁹 𒌋𒐌 𒌋𒐈 𒐌 𒐊 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒐏𒐈𒆠	𒑲
𒑲𒑰𒌋𒐉 𒌋𒐊 𒌍 𒎙𒐉 𒁹 𒐏𒁹 𒐐𒐈 𒐉 𒐈 𒎙𒐉 𒐏𒁹 𒌍𒐊 𒐏𒐈 𒌋𒐈 𒐌 𒐊 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒐏𒐉𒆠	𒑲
𒑲𒑰𒐌 𒌍𒐋 𒐏𒐉 𒐐𒐉 𒎙𒈫 𒌍𒐉 𒐐𒐋 𒌍𒐊 𒐐𒁹 𒐋 𒌍𒐈 𒌋𒐈 𒐏𒐊 𒎙𒁹 𒐐𒐌 𒈫 𒎙𒐉𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒐏𒐊𒆠	𒑲
𒑲𒑰𒎙𒐎 𒐐𒁹 𒐐𒐈 𒐏𒐊 𒐊 𒐋 𒎙𒐋 𒑲 𒎙𒐋 𒐌 𒎙𒐈 𒎙𒈫 𒎙𒐋 𒐏𒁹 𒌋𒐋 𒐏𒐋	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒐏𒐋𒆠	𒑲
𒑲𒑰𒌋𒐎 𒌋𒐉 𒐐𒐋 𒑲 𒐎 𒐏𒐊 𒎙𒐎 𒌍𒐋 𒁹 𒐐𒁹 𒐐𒐉 𒈫 𒎙𒐊 𒐌 𒐊 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒁹 𒐏𒐌𒆠	𒑲
𒑲𒑰𒐐𒈫 𒐎 𒎙𒐈 𒌍𒐎 𒐋 𒌍𒐎 𒐊 𒐐𒐎 𒐐𒈫 𒌍𒐉 𒐐𒐋 𒑲 𒐈 𒐉 𒌋𒐎 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒐏𒐍𒆠	𒑲
𒑲𒑰𒐎 𒌋𒁹 𒑲 𒌍 𒎙𒐊 𒎙𒐉 𒐏𒐊 𒐐𒐉 𒐈 𒐋 𒌋 𒑲 𒌋𒐎 𒐏𒐋 𒐐𒁹 𒐐 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒁹 𒐏𒐎𒆠	𒑲
𒑲𒑰𒐊 𒑲𒐉 𒐎 𒐊 𒌍 𒎙𒐉 𒌍𒁹 𒌋𒈫 𒎙𒐉 𒑲 𒁹 𒐐𒐋 𒐌 𒌋𒐊 𒐏𒐊 𒌍𒐋	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒑲𒆠	𒑲
𒑲𒑰𒐊 𒌋𒁹 𒁹 𒎙𒁹 𒌍𒐎 𒐊 𒐏𒐈 𒐐𒐊 𒌍𒐌 𒌋𒁹 𒐐𒐎 𒑲 𒌍𒐎 𒌋𒐉 𒐏𒁹 𒌋𒐋 𒐏𒐍	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒁹𒆠	𒑲
𒑲𒑰𒐋 𒌋 𒑲𒐊 𒎙𒐈 𒐐𒐈 𒌋𒐊 𒐏𒐎 𒐉 𒐏𒐋 𒌍𒁹 𒎙𒈫 𒐐𒐊 𒎙𒐍 𒐎 𒐐𒐎 𒈫 𒎙𒐉𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒈫𒆠	𒑲
𒑲𒑰𒈫 𒌍𒐋 𒐐 𒐏𒐈 𒎙𒐉 𒐐𒐊 𒐐𒐋 𒎙𒐋 𒎙 𒐐𒐉 𒎙𒐎 𒐋 𒌋 𒐐𒐋 𒐐𒐋 𒌍𒐋 𒎙𒐉𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐈𒆠	𒑲
𒑲𒑰𒐍 𒑲 𒌍𒈫 𒐏𒐈 𒐋 𒐏𒈫 𒐐𒐊 𒌋𒐋 𒌋𒐋 𒌍𒐋 𒐐 𒌍𒐈 𒌍𒐊 𒐐 𒎙𒐊 𒎙𒁹 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐉𒆠	𒑲
𒑲𒑰𒐊 𒎙𒐊 𒑲 𒌍𒐈 𒌋𒐉 𒎙𒈫 𒌋𒐈 𒌍𒐌 𒎙𒐋 𒐋 𒐐𒐉 𒎙 𒌍𒐊 𒌋𒐉 𒐐𒈫 𒐏𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐊𒆠	𒑲
𒑲𒑰𒁹 𒐐𒐌 𒐋 𒐏𒐉 𒐐𒐊 𒐐𒁹 𒐐𒐊 𒎙𒐉 𒎙𒐉 𒎙 𒌍𒐊 𒌍𒐌 𒐊 𒐊 𒐊 𒌋𒐋 𒐏𒐍	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐋𒆠	𒑲
𒑲𒑰𒐈 𒎙𒐉 𒈫 𒌍𒈫 𒐐𒈫 𒌍𒁹 𒎙𒐋 𒌋𒐍 𒑲 𒐐 𒐐𒐈 𒐉 𒌍𒐊 𒌋𒐌 𒑲 𒐏𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐌𒆠	𒑲
𒑲𒑰𒌋 𒌋𒈫 𒌍𒐋 𒐐𒐈 𒑲 𒎙𒐊 𒌍𒈫 𒐐𒈫 𒈫 𒐏𒐋 𒌋𒐉 𒌋𒐌 𒌍𒈫 𒐐𒐊 𒑲 𒐏𒐍	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐍𒆠	𒑲
𒑲𒑰𒐈 𒐊 𒈫 𒌍 𒌋𒈫 𒎙𒐉 𒐏𒐊 𒎙𒐋 𒎙𒁹 𒑲𒐊 𒐊 𒎙𒐉 𒌋𒐊 𒐏𒐈 𒐏𒁹 𒎙𒁹 𒎙𒐋𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐎𒆠	𒑲
𒑲𒑰𒐊 𒑲𒐊 𒐏𒐌 𒐈 𒌋𒐊 𒑲 𒌍𒐋 𒌋𒈫 𒌋𒐊 𒑲𒐋 𒑲𒐋 𒎙𒐌 𒎙𒐋 𒌋𒁹 𒌋𒐌 𒑲 𒑲𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌋𒆠	𒑲
𒑲𒑰𒐋 𒎙𒐈 𒐏𒈫 𒐏𒐊 𒐏𒐊 𒐊 𒐏𒁹 𒌋𒐊 𒌋𒐋 𒌋𒐉 𒐏𒁹 𒎙 𒐉 𒑲𒐊 𒑲𒁹 𒑲𒐊 𒎙𒐋	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌋𒁹𒆠	𒑲
𒑲𒑰𒐋 𒎙𒐌 𒑲𒐌 𒐊 𒎙𒐋 𒐌 𒐊 𒌋𒐋 𒑲𒐉 𒐌 𒌋𒐉 𒐈 𒌋𒐍 𒎙𒐋 𒌋𒐊 𒌋𒐋 𒑲𒐍	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌋𒈫𒆠	𒑲
𒑲𒑰𒈫 𒑲𒐍 𒌋𒐊 𒐏𒐍 𒐋 𒐏𒈫 𒈫 𒑲𒐉 𒎙𒐉 𒌍𒐊 𒐏𒈫 𒐈 𒐏𒐉 𒐊 𒌋𒁹 𒈫 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌋𒐈𒆠	𒑲
𒑲𒑰𒌋𒈫 𒁹 𒐏 𒁹 𒎙𒐊 𒌍𒐌 𒎙𒐍 𒐏𒐉 𒐉 𒐏𒐈 𒎙𒈫 𒌍𒈫 𒎙𒐌 𒌍𒐋 𒑲𒁹 𒌋𒐋 𒑲𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌋𒐉𒆠	𒑲
𒑲𒑰𒈫 𒌍𒐌 𒌋𒐋 𒑲𒐈 𒌋𒐉 𒌋𒐌 𒐌 𒌋𒐊 𒌍𒐋 𒑲𒁹 𒑲𒁹 𒌋𒐍 𒐏𒈫 𒎙𒐉 𒐏𒐊 𒌍𒐋𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌋𒐊𒆠	𒑲
𒑲𒑰𒐈 𒌋𒐍 𒑲𒁹 𒌍𒐉 𒎙𒐊 𒐋 𒑲 𒎙𒁹 𒐊 𒌍𒐌 𒐏𒈫 𒑲 𒎙𒐌 𒎙 𒌋𒐊 𒎙𒁹 𒌍𒐋	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌋𒐋𒆠	𒑲
𒑲𒑰𒁹 𒌍𒁹 𒐊 𒐏𒐌 𒌋𒈫 𒎙𒐋 𒐋 𒑲𒈫 𒌋𒈫 𒎙𒐊 𒎙𒐉 𒎙𒐉 𒌋𒈫 𒌍𒐌 𒌍𒐈 𒌋𒐍 𒑲𒐈 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌋𒐌𒆠	𒑲
𒑲𒑰𒐋 𒑲𒐊 𒑲𒐈 𒑲𒐌 𒌍𒐌 𒑲𒐍 𒎙𒐍 𒐏𒐌 𒌍𒐊 𒑲𒐍 𒑲 𒐏𒁹 𒌋𒐋 𒌋𒐍 𒈫 𒐏𒈫 𒑲𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌋𒐍𒆠	𒑲
𒑲𒑰𒐈 𒐏𒈫 𒈫 𒐋 𒐍 𒌍𒐈 𒎙 𒐏𒈫 𒐏𒐊 𒌋𒐈 𒑲𒐊 𒑲𒈫 𒎙𒐊 𒐉 𒌋𒐌 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌋𒐎𒆠	𒑲
𒑲𒑰𒈫 𒌍𒐊 𒐏𒐊 𒌋𒐊 𒌍𒁹 𒐏𒈫 𒐊 𒐉 𒎙𒐌 𒌋 𒌋 𒌍𒐋 𒐏𒐊 𒐈 𒐏𒐋 𒐌 𒌍𒐋𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒎙𒆠	𒑲
𒑲𒑰𒐈 𒐏 𒌋𒐋 𒁹 𒌋𒈫 𒈫 𒌍𒐉 𒌋𒐉 𒐏𒈫 𒐏𒐌 𒐌 𒐏𒁹 𒎙 𒑲 𒑲𒈫 𒌋𒐉 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒎙𒁹𒆠	𒑲
𒑲𒑰𒈫 𒐌 𒑲𒐊 𒑲𒐈 𒐉 𒑲𒁹 𒐋 𒌋𒈫 𒐊 𒐏𒐌 𒌋𒐉 𒑲𒁹 𒌍𒐋 𒐏𒐊 𒐋 𒌋𒐉 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒎙𒈫𒆠	𒑲
𒑲𒑰𒐉 𒌍𒐋 𒐏𒐈 𒐏𒐍 𒐏𒐉 𒐏 𒐏𒐊 𒑲𒁹 𒌋𒐉 𒎙𒐊 𒌋𒈫 𒌍𒐈 𒐏𒐉 𒐏𒐈 𒌍𒐉 𒐉 𒑲𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒎙𒐈𒆠	𒑲
𒑲𒑰𒌋 𒑲𒁹 𒐋 𒌋𒐌 𒐏𒁹 𒌋𒁹 𒎙𒁹 𒐊 𒐏 𒑲𒐋 𒌍𒐌 𒌍𒁹 𒑲 𒎙𒐍 𒑲𒐍	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒎙𒐉𒆠	𒑲
𒑲𒑰𒐈 𒎙𒐋 𒎙𒈫 𒁹 𒎙𒐊 𒑲 𒎙𒐌 𒑲𒐌 𒑲 𒎙𒈫 𒌍𒈫 𒑲𒐍 𒐋 𒑲𒐌 𒌋𒁹 𒎙𒐊 𒎙𒐋 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒎙𒐊𒆠	𒑲
𒑲𒑰𒐊 𒑲𒐊 𒌋𒐉 𒎙 𒌋𒐈 𒌍𒐉 𒌋𒐊 𒌍𒐉 𒎙𒐍 𒐈 𒌍𒐌 𒐏𒐊 𒐏 𒌋𒐉 𒑲𒐊 𒈫 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒎙𒐋𒆠	𒑲
𒑲𒑰𒐈 𒐏𒐋 𒑲𒐍 𒎙𒐊 𒌋𒈫 𒑲𒐋 𒐏𒈫 𒑲𒐍 𒐌 𒌍𒐌 𒑲𒐋 𒎙 𒌍𒐌 𒐈 𒌋 𒐉 𒑲𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒎙𒐌𒆠	𒑲
𒑲𒑰𒁹 𒐏𒐊 𒌋𒐊 𒎙𒈫 𒎙𒐍 𒑲𒐍 𒑲𒐌 𒑲𒐊 𒐊 𒑲𒐈 𒎙𒐋 𒎙𒐋 𒎙𒐌 𒌋𒐊 𒐏𒐊 𒐊 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒎙𒐍𒆠	𒑲
𒑲𒑰𒌋𒐈 𒑲𒐋 𒑲𒐍 𒌍𒐍 𒌍𒐋 𒌋𒐊 𒐊 𒌋 𒑲 𒌋𒐋 𒌍𒁹 𒌍𒐊 𒎙𒐈 𒎙𒐉 𒌋𒐊 𒌋𒐋 𒑲𒐍	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒎙𒐎𒆠	𒑲
𒑲𒑰𒌋𒈫 𒑲𒁹 𒎙𒐋 𒑲𒐍 𒎙𒐊 𒈫 𒐏𒁹 𒌍𒐈 𒐊 𒌍𒈫 𒑲𒐊 𒐏𒐋 𒌋𒐍 𒑲𒐍 𒐏𒐊 𒌍𒐋𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌍𒆠	𒑲
𒑲𒑰𒐉 𒎙𒐋 𒌍𒐍 𒌍𒈫 𒌍𒐊 𒌋𒐉 𒑲𒐌 𒐊 𒑲𒐊 𒐏𒈫 𒑲𒐊 𒐏𒐋 𒑲𒁹 𒎙𒐈 𒑲𒈫 𒑲𒐈 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌍𒁹𒆠	𒑲
𒑲𒑰𒐉 𒎙 𒑲𒐈 𒈫 𒐏𒐉 𒑲𒁹 𒎙𒈫 𒐍 𒎙𒐊 𒐏𒐊 𒐏𒐋 𒎙𒐌 𒌋𒐈 𒑲𒐍 𒌋𒐊 𒌋𒐋 𒑲𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌍𒈫𒆠	𒑲
𒑲𒑰𒁹 𒑲𒁹 𒎙 𒎙𒐊 𒌍𒐉 𒐋 𒑲𒈫 𒑲𒈫 𒌋𒐈 𒐏𒈫 𒐏𒐊 𒐏𒈫 𒎙𒐋 𒎙𒐍 𒐏 𒌋𒐍 𒌋𒐉 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌍𒐈𒆠	𒑲
𒑲𒑰𒐉 𒐍 𒐍 𒌋𒐊 𒌍𒐌 𒌋𒁹 𒌍 𒑲𒐌 𒌋𒐈 𒐋 𒎙𒁹 𒐏𒐍 𒌍𒁹 𒐏𒐌 𒌋𒐈 𒐏𒐊 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌍𒐉𒆠	𒑲
𒑲𒑰𒌋𒐉 𒎙𒐌 𒐈 𒌍𒐊 𒑲𒐊 𒌍 𒎙𒐍 𒎙𒐈 𒐏𒐍 𒑲𒐍 𒌋 𒐈 𒑲𒐌 𒐏𒐋 𒎙𒁹 𒐌 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌍𒐊𒆠	𒑲
𒑲𒑰𒁹 𒈫 𒌍𒐉 𒌋𒁹 𒎙𒐊 𒐋 𒌋𒐈 𒐏 𒌋𒐉 𒌋𒈫 𒐏𒐊 𒎙𒐌 𒐏𒐊 𒈫 𒑲𒐊 𒈫 𒎙𒐉𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌍𒐋𒆠	𒑲
𒑲𒑰𒐊 𒌍𒐉 𒈫 𒎙𒐍 𒌍𒐈 𒐌 𒐏𒐉 𒌋𒐌 𒐏𒐍 𒎙𒐋 𒐉 𒑲 𒎙𒐋 𒌍𒐉 𒎙𒈫 𒐉 𒑲𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌍𒐌𒆠	𒑲
𒑲𒑰𒈫 𒌋𒐌 𒐏𒐊 𒑲𒐉 𒐈 𒑲𒐉 𒑲𒐉 𒌍𒐈 𒁹 𒑲𒈫 𒎙𒐊 𒌍𒐌 𒐌 𒐏 𒐏𒐍 𒌍𒐈 𒌍𒐋𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌍𒐍𒆠	𒑲
𒑲𒑰𒌋𒐊 𒌋𒐉 𒐏 𒌍𒐋 𒈫 𒎙𒐊 𒐌 𒑲𒈫 𒑲𒈫 𒐏𒐊 𒌍𒐈 𒑲𒐊 𒑲𒐊 𒁹 𒎙𒐋 𒎙𒐉𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒌍𒐎𒆠	𒑲
𒑲𒑰𒈫 𒑲 𒎙𒐊 𒌍𒐉 𒌋𒐍 𒌋𒐍 𒎙 𒌍𒐌 𒐏𒐊 𒑲 𒐏𒐉 𒑲 𒐏𒐈 𒐏𒐋 𒑲𒐍 𒐏𒐊 𒌍𒐋𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐏𒆠	𒑲
𒑲𒑰𒐈 𒌋𒐋 𒁹 𒌋𒐍 𒑲 𒐏𒐍 𒌍𒁹 𒎙𒐋 𒌋𒁹 𒌋𒐊 𒑲 𒌍 𒑲 𒎙𒐋 𒐋 𒑲𒐈 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐏𒁹𒆠	𒑲
𒑲𒑰𒐈 𒐏𒐌 𒐊 𒌍𒐈 𒌋 𒌋𒈫 𒌍𒐈 𒑲𒐊 𒐏𒐊 𒐏𒐍 𒐋 𒐌 𒐏 𒌋𒐊 𒐊 𒌋𒐋 𒑲𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐏𒈫𒆠	𒑲
𒑲𒑰𒐊 𒑲𒐌 𒐏𒐊 𒐏 𒐋 𒐏𒐋 𒁹 𒑲𒐊 𒎙 𒌋𒐈 𒐊 𒌋𒐉 𒎙𒐉 𒌋𒐈 𒑲𒐌 𒎙𒐋 𒎙𒐉𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐏𒐈𒆠	𒑲
𒑲𒑰𒐉 𒎙𒐊 𒎙 𒑲𒐈 𒐊 𒐏𒐊 𒎙𒐋 𒐏𒐋 𒐊 𒎙𒁹 𒎙𒁹 𒐈 𒐊 𒐏𒐊 𒐏𒐍 𒐉 𒑲𒐍	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐏𒐉𒆠	𒑲
𒑲𒑰𒐌 𒌋𒐊 𒐏𒐊 𒐉 𒌍𒈫 𒐋 𒐌 𒑲𒐊 𒑲𒁹 𒌋𒐊 𒌍𒐊 𒑲 𒎙𒈫 𒌋𒁹 𒑲𒈫 𒑲𒐈 𒌋𒈫𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐏𒐊𒆠	𒑲
𒑲𒑰𒐊 𒐏𒐌 𒐏𒐋 𒌋𒐍 𒐏𒐈 𒐏𒈫 𒑲𒐍 𒐌 𒌋𒐊 𒌍𒐊 𒌍𒐌 𒐊 𒌍𒐍 𒌋𒐊 𒐏𒐋 𒐌 𒌍𒐋	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐏𒐋𒆠	𒑲
𒑲𒑰𒐉 𒌍𒐉 𒑲𒐋 𒎙 𒑲𒐋 𒑲𒁹 𒑲𒐊 𒌋𒐉 𒎙𒐈 𒌍𒈫 𒎙𒈫 𒑲𒐋 𒑲𒐈 𒐏𒐍 𒐉 𒑲𒐍	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐏𒐌𒆠	𒑲
𒑲𒑰𒌋𒐈 𒈫 𒌍𒁹 𒌋𒐊 𒎙𒐈 𒌍𒐋 𒑲𒈫 𒐏𒈫 𒐊 𒐋 𒌋𒐉 𒌋𒐉 𒈫 𒑲𒐍 𒌋𒁹 𒌍𒁹 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐏𒐍𒆠	𒑲
𒑲𒑰𒐈 𒑲𒐊 𒌋 𒐊 𒎙𒁹 𒌍𒐈 𒐏𒈫 𒎙𒐉 𒐏𒐋 𒑲𒐋 𒐋 𒌍𒐈 𒐈 𒑲𒐉 𒎙𒐍 𒎙𒐉𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒑲 𒐏𒐎𒆠	𒑲
𒑲𒑰𒐌 𒌋 𒐉 𒐈 𒌋𒐍 𒌍 𒑲𒐊 𒌍𒈫 𒌋𒐍 𒑲 𒐈 𒐏𒈫 𒌋𒐉 𒌍𒐊 𒌍𒁹 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒑲𒆠	𒑲
𒑲𒑰𒐌 𒌍𒐍 𒑲𒐋 𒐏𒐍 𒎙𒐈 𒎙𒐋 𒑲 𒐏𒐌 𒌍𒈫 𒑲𒐋 𒌋𒐉 𒌋𒐊 𒌍𒈫 𒐏𒐊 𒑲 𒑲𒐍𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒁹𒆠	𒑲
𒑲𒑰𒈫 𒌋𒐍 𒌋𒈫 𒌋𒐊 𒌋𒈫 𒑲𒐌 𒌍𒐍 𒐊 𒌋𒐉 𒌍𒐊 𒌋𒐍 𒌋𒐌 𒑲𒐌 𒑲𒁹 𒌍𒐋 𒎙𒐈 𒈫 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒈫𒆠	𒑲
𒑲𒑰𒐋 𒐏𒐊 𒐋 𒑲𒐈 𒌍𒈫 𒎙 𒎙𒁹 𒐏 𒑲 𒐌 𒌋𒐌 𒌍 𒌋𒈫 𒐊 𒑲𒐊 𒌍𒐋𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒐈𒆠	𒑲
𒑲𒑰𒐊 𒐋 𒑲 𒐉 𒌍𒐍 𒑲 𒎙 𒐌 𒎙𒐊 𒑲𒐈 𒌍𒐋 𒐌 𒐊 𒑲𒐋 𒑲𒐊 𒐌 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒐉𒆠	𒑲
𒑲𒑰𒑲 𒌋𒐊 𒐌 𒎙𒐊 𒑲𒐈 𒌍𒐊 𒐉 𒎙𒐊 𒁹 𒑲𒐉 𒌋𒐊 𒐊 𒑲 𒎙𒐌 𒐋 𒎙𒐊 𒐏𒐊 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒐊𒆠	𒑲
𒑲𒑰𒐊 𒑲 𒐏 𒑲 𒌋𒐈 𒎙𒐍 𒌍𒁹 𒎙 𒌋𒐍 𒎙𒐊 𒑲𒐈 𒁹 𒌍𒐊 𒐏𒐊 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒐋𒆠	𒑲
𒑲𒑰𒑲 𒌋 𒎙𒐋 𒑲 𒌋𒐍 𒑲𒐈 𒎙𒈫 𒐊 𒌋𒐌 𒐋 𒑲𒐋 𒑲𒐊 𒐏𒈫 𒌍𒈫 𒐋 𒐋 𒌍𒐍 𒎙𒐉𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒐌𒆠	𒑲
𒑲𒑰𒐉 𒐌 𒐌 𒑲𒁹 𒑲𒈫 𒌋𒐊 𒎙𒐌 𒌋𒁹 𒑲𒐋 𒌋𒐊 𒁹 𒑲 𒑲𒁹 𒌋𒐉 𒎙𒐌 𒑲𒐊 𒌍𒐋	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒐍𒆠	𒑲
𒑲𒑰𒈫 𒌍𒁹 𒌍𒐍 𒌋𒈫 𒑲𒈫 𒎙𒁹 𒌋𒐍 𒐍 𒐏𒐊 𒐏𒈫 𒌍𒈫 𒎙𒐈 𒐏 𒌋𒐌 𒎙𒐈 𒌍𒁹 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒐎𒆠	𒑲
𒑲𒑰𒈫 𒐏𒐋 𒐈 𒌍𒐋 𒌋𒐌 𒐏𒐍 𒐍 𒎙𒐌 𒌋𒐌 𒌍𒐉 𒐋 𒐏𒈫 𒎙𒐉 𒎙𒐌 𒐏𒐊 𒐊 𒌋𒈫	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒌋𒆠	𒑲
𒑲𒑰𒐌 𒌋 𒐏𒐉 𒌍𒐊 𒑲𒁹 𒎙𒈫 𒌍𒐉 𒑲 𒌍𒐊 𒐏𒐌 𒑲𒐍 𒎙𒐈 𒌋𒐋 𒌋𒐈 𒎙𒐋 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒌋𒁹𒆠	𒑲
𒑲𒑰𒐉 𒐏𒐊 𒑲𒈫 𒐏𒐍 𒑲𒈫 𒌋𒐈 𒐋 𒌍𒐍 𒐏𒐉 𒑲𒐌 𒐏𒐋 𒐌 𒎙 𒎙𒐍 𒌋𒐈 𒎙𒐋 𒎙𒐉𒄩𒑲	𒁲𒈦	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒌋𒈫𒆠	𒑲
𒑲𒑰𒐈 𒎙𒐋 𒐍 𒎙𒐍 𒐏𒐊 𒑲 𒐉 𒐉 𒑲𒐈 𒌋𒐉 𒐏𒈫 𒌋𒐍 𒐍 𒎙𒐉 𒌍𒐉 𒌍𒐈 𒎙𒐋	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒌋𒐈𒆠	𒑲
𒑲𒑰𒑲 𒐏𒈫 𒌍𒐍 𒑲𒐌 𒐏𒈫 𒌋𒐋 𒌋𒐍 𒌋𒐉 𒎙𒐈 𒎙𒁹 𒎙 𒑲𒁹 𒌋𒈫 𒐊 𒐏𒐌 𒎙𒐊 𒎙𒐋 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒌋𒐉𒆠	𒑲
𒑲𒑰𒑲 𒐏𒐉 𒐏𒐋 𒎙𒐌 𒌍𒐉 𒎙𒈫 𒌍𒁹 𒌍𒐌 𒐏𒐉 𒎙𒐊 𒐉 𒎙𒐌 𒌋𒐋 𒐏𒐋 𒌋𒐊 𒌍𒐌 𒐏 𒎙𒐉	𒀸	𒉼𒁲 𒁹𒑰𒈫 𒁹 𒌋𒐊𒆠	𒑲

𒄑𒉿𒄑𒉿𒄑𒉿

𒄿𒈬𒋛𒄿𒈬𒋛𒄿𒈬𒋛

𒄑𒀀𒄑𒀀𒄑𒀀

𒄑𒋻𒄑𒋻𒄑𒋻

[illegible]

𒑱𒐈 𒌍𒐉 𒌍𒐍 𒎙𒐋 𒌍𒐈 𒈫 𒐏𒐋 𒎙 𒐏 𒐐𒐋 𒌋𒐋 𒐍 𒐏𒐍 𒌍𒁹 𒌋𒈫
𒑱𒈫 𒎙𒐋 𒎙 𒐏𒐌 𒐐𒁹 𒎙𒐉 𒐐𒁹 𒎙𒐋 𒌍𒐌 𒌋 𒐈 𒐐 𒎙𒐍 𒌍𒁹 𒐏𒐋 𒌍𒐈 𒌍𒐋
𒑱 𒐏𒁹 𒐐𒐈 𒌍𒐌 𒐐 𒌋𒐋 𒐐𒈫 𒌋𒐋 𒐏𒈫 𒎙𒐈 𒐐 𒎙𒁹 𒎙𒐍 𒌍𒐌 𒐏𒁹 𒌍𒐉 𒐉 𒐏𒐋
𒑱𒐌 𒐏𒐋 𒎙𒁹 𒎙𒐋 𒐐𒐋 𒐐𒐉 𒈫 𒐐𒐋 𒌋𒐋 𒎙𒐌 𒌍𒐈 𒎙𒁹 𒁹 𒎙𒐋 𒎙𒐉
𒑱 𒐏𒐍 𒐏𒐍 𒐐𒈫 𒎙𒐍 𒐏𒐍 𒌍𒁹 𒐐𒐋 𒌍𒐋 𒐏 𒌋𒐋 𒎙𒐋 𒌍𒈫 𒌋𒐉 𒎙𒐋 𒌋𒐋 𒌋𒐉 𒎙𒐉
𒑱 𒐈 𒌋𒈫 𒐏𒐈 𒌋𒁹 𒎙𒐉 𒌍𒁹 𒐏𒐋 𒐏𒐌 𒎙𒐋 𒈫 𒎙𒐋 𒐍 𒐐𒐋 𒐏𒐉 𒐍 𒌍𒐋
𒑱𒁹 𒌍𒈫 𒐏𒁹 𒐐𒈫 𒎙𒈫 𒌋𒐍 𒐐𒁹 𒐏𒈫 𒈫 𒐏 𒎙𒐌 𒐏𒁹 𒌋𒐉 𒐏𒐈 𒌍𒐌 𒈫 𒎙𒐉
𒑱𒐉 𒐐𒐈 𒐏𒁹 𒌋𒐋 𒐌 𒐐𒐍 𒐏𒈫 𒐏𒐋 𒐏𒐉 𒐍 𒌋𒐉 𒐈 𒌋𒐋 𒌍𒐋 𒌋𒐌 𒌋𒐋 𒐏𒐋
𒑱𒁹 𒌋𒐋 𒐏𒐍 𒌋 𒌋𒁹 𒐏𒐉 𒐐𒐌 𒌍𒐋 𒐏𒐌 𒐐𒐍 𒐐𒐍 𒁹 𒐐𒐋 𒎙𒁹 𒐌 𒌋𒈫
𒑱𒐌 𒐐𒐋 𒌍𒐍 𒐏𒁹 𒌍𒐋 𒐋 𒐐𒐌 𒌋𒐌 𒌋𒐉 𒁹 𒐏 𒌋𒐌 𒌋𒐋 𒌍𒐋 𒌍𒐌 𒌍𒁹 𒌋𒈫
𒑱𒐉 𒐌 𒁹 𒐌 𒌍𒐈 𒌍𒐉 𒌍𒐍 𒐋 𒐋 𒎙𒐌 𒌍𒐌 𒌋𒐈 𒌋𒐋 𒁹 𒐐𒐌 𒌋𒈫
𒑱𒈫 𒌍𒐉 𒌋 𒎙𒁹 𒎙𒐉 𒐉 𒌋𒐉 𒐋 𒐏𒐍 𒌍𒐉 𒐋 𒐐𒈫 𒎙𒐉 𒎙𒐍 𒐐𒐌 𒐌 𒌋𒈫
𒑱𒐋 𒐐𒐋 𒌍𒐍 𒐏 𒌋𒐍 𒐋 𒎙𒐋 𒐐𒐋 𒌋𒁹 𒐐𒐉 𒎙𒈫 𒎙𒁹 𒐏𒐌 𒎙𒐋 𒐏𒐌 𒈫 𒎙𒐉
𒑱𒌋 𒐌 𒎙𒈫 𒌍 𒎙𒐌 𒎙𒐋 𒐐𒐌 𒐍 𒌋𒐋 𒐌 𒎙𒐌 𒐐𒐌 𒌍𒐋 𒐉 𒎙𒐉 𒐐𒐌 𒌍𒐋

[illegible]

𒐕 𒁹𒌋𒁕𒄿	𒁲	𒀸𒌋𒁕𒄿𒁹𒄿	𒁹
𒁹𒌋𒁕 𒁹𒄿	𒁲	𒀸𒌋𒁕𒄿𒁹𒄿	𒁹
𒁹𒌋𒁕𒄿 𒁹𒌋𒁕𒄿	𒁲	𒀸𒌋𒁕𒄿𒁹𒄿	𒁹
𒁹𒌋𒁕𒄿	𒁲	𒀸𒌋𒁕𒄿𒁹𒄿	𒁹
𒁹𒌋𒁕𒄿	𒁲	𒀸𒌋𒁕𒄿𒁹𒄿	𒁹
𒌋𒌋	𒁲𒁹	𒁹𒁹	𒁹
𒌋	𒁲𒁹𒁹𒁹	𒁹𒁹	𒁹

𒁹𒌋𒁕𒄿𒁹𒄿𒁹𒄿𒁹𒄿𒁹𒄿𒁹𒄿𒁹𒄿𒁹𒄿𒁹𒄿𒁹𒄿