

# The Ouroboric Singularity of Lexical Minimalism: A Fixed-Point Theory of Self-Hosting Single-Character Compilers in the Post-UCRT Epoch

## Introduction and Epistemological Context

The trajectory of programming language design over the past six decades has been characterized by a relentless and largely unquestioned march toward syntactic maximalism. Modern software engineering conventions heavily prioritize human readability, dictating the use of verbose identifiers, expansive standard libraries, and sprawling architectural paradigms. The mainstream engineering consensus adamantly argues that code must be written primarily for human comprehension rather than mere computational execution, a philosophical plague introduced to the discipline by foundational texts such as *The Elements of Programming Style* in 1974. This pedagogical dogma inherently restricts the computational canvas, treating verbosity as an unquestionable virtue while ignoring the theoretical and practical benefits of radical minimization. The overarching reliance on multisyllabic variables, explicit keyword reservations, and redundant control-flow structures drastically increases the cognitive overhead, byte-level footprint, and lexical redundancy of source files.

Within the prestigious, rigorously unorthodox, and occasionally hallucinatory annals of the Association for Computational Heresy (ACH), esoteric programming languages (commonly abbreviated as esolangs) have repeatedly challenged these base assumptions. These languages demonstrate that personal expression, algorithmic elegance, and pure logic can thrive within environments of extreme syntactical chaos and deliberate obfuscation. Historically, the SIGBOVIK proceedings have served as the premier venue for disseminating such paradigm-shifting research, ranging from the performance evaluations of Scratch to the derivation of mathematically pure anomalies. Esolangs favor individual personality over strict practicality, prioritizing expression over utilitarian efficiency, and challenging the foundational assumptions regarding who programming languages are designed for and how they should fundamentally operate.

The present research introduces py1, a strictly single-character token dialect of the Python programming language designed to push the boundaries of lexical parsimony to their absolute theoretical limits. Positioned at the complex intersection of extreme code golfing, cryptographic program obfuscation, and formal fixed-point compiler theory, py1 enforces an uncompromising architectural constraint. Following a brief definitional preamble, all logical, arithmetic, and control-flow identifiers within the execution body must be represented by exactly one character. The primary contribution of this work is fundamentally twofold. First, it formalizes the mathematical upper bounds of lexical parsimony through a surjective mapping of multisyllabic Python keywords onto single Unicode glyphs, specifically leveraging high-density Kanji characters to encapsulate entire logical subroutines. Second, it establishes a mathematically rigorous, three-stage ouroboric bootstrapping pipeline capable of generating a native x64 binary that completely and aggressively bypasses the Universal C Runtime (UCRT). By directly

addressing the Windows native application programming interface (kernel32.dll) through a custom intermediate representation (IR), the py1 compiler eliminates all external runtime dependencies, achieving a theoretically pure, Turing-complete execution environment. Through rigorous cryptographic hashing of the compiler's source code across successive generations of self-compilation , the architecture demonstrates strict idempotency, proving definitively that the compiler reaches a terminal fixed point in semantic space.

## The Theoretical Framework of Lexical Parsimony and Information Entropy

### The Shannon Entropy of Single-Character Identifiers

To fully appreciate the architectural necessity of py1, one must first examine the inherent redundancies of traditional programming languages through the rigorous lens of information theory. Claude Shannon's seminal theorems on source coding establish an absolute mathematical limit on how well data from a given source can be losslessly compressed over a perfectly noiseless channel. In conventional programming paradigms, the cross-entropy of typical English-based source code is highly sub-optimal. Historical estimates suggest that English text and corresponding high-level programming structures possess an entropy of roughly 1.75 to 2.3 bits per character. This indicates a massive volume of structural constraint and statistical predictability—such as the tendency for certain letters to invariably follow others, or the mandatory inclusion of whitespace and punctuation to appease the lexical scanner. Let  $X$  denote the discrete random variable representing the alphabet of valid lexical tokens in a given programming language, and let  $p(x_i)$  represent the probability mass function of each token. The theoretical entropy  $H(X)$  is defined by the classical equation:

Mainstream programming guidelines universally insist on descriptive variable names—a practice colloquially termed "meaningful identifiers"—to aid in program comprehension and maintenance. However, if the lexical scope of a variable is heavily constrained, the utility of this verbosity completely collapses. In py1, the language enforces a hard, uncompromising truncation of identifier length to  $L = 1$  character. By mapping complex control structures (e.g., while, print, elif) into single high-density Unicode glyphs, the semantic elasticity of the program is stretched to its absolute theoretical limit. Because each ideogram in the py1 syntax encapsulates an entire functional concept, the information density per character asymptotically approaches the theoretical upper bound of the source channel, completely eliminating the statistical redundancy inherent in English-keyword-based languages.

### Lexical Density and the Halliday Formulation

Lexical density is a formal metric utilized heavily in discourse analysis and computational linguistics to quantify the proportion of significant content-bearing words against the total token count of a document. Utilizing the revised Halliday formulation :

Where  $N_{\{lex\}}$  represents the number of lexical items (content words) and  $N_{\{total\}}$  represents total clauses or total words. Standard enterprise software rarely exceeds a lexical density of 56% due to the massive proliferation of syntactic sugar, mandatory whitespace, delimiter-separated words, and boilerplate modifiers. The py1 specification aggressively purges these non-content-bearing tokens. By collapsing entire operational concepts into a single

character, py1 code approaches a theoretical lexical density of  $L_d = 100\%$ , establishing an unprecedented paradigm of extreme parsimony that challenges the foundational assumptions of generalized parsing. Every single character parsed by the lexical analyzer carries maximum semantic weight, leaving no room for stylistic vanity.

## Obfuscation and the Eradication of the "Rodgular" Anti-Pattern

While usability is rarely the objective of esoteric programming languages , the cryptographic and system security implications of py1 are non-trivial. The practice of writing intentionally convoluted code is historically classified under various academic and industry pejoratives, including the "Rodgular" programming anti-pattern, defined as a software design technique characterized by the intentional combination of logic in a complex and convoluted manner. Code obfuscation transforms source material into a functionally equivalent but hermeneutically opaque state to deter reverse engineering, intellectual property theft, and automated static analysis.

The py1 architecture introduces a novel defense mechanism termed *Concatenative Opcode Entropy*. Static analysis tools, decompilers, and emerging artificial intelligence models frequently rely on heuristic pattern matching to identify standard opcodes and execution flows. To circumvent this, py1 developers initialize constants using dynamic string concatenations that evade strict static string checks. For example, the MOV instruction is defined not as a literal string in the binary, but dynamically derived at runtime as "M"+ "O"+ "V". When paired with the uniform assignment of independently sampled linguistic bases, the adversary's inference probability is driven to negligible values, fundamentally mirroring the theoretical models used to protect quantum circuits through compiler-resistant obfuscation.

## The Formal Specification and Syntax Mappings of py1

The formal syntax of py1 is deliberately segregated into two strictly enforced and mutually exclusive phases: the Definition Phase and the Body Phase. This strict bifurcation allows the compiler to maintain a minimal trusted computing base while simultaneously supporting arbitrary, user-defined token mappings that can adapt to any specific esoteric requirement.

### The Ontological Grounding of the Definition Phase

The epistemological grounding of a py1 program occurs entirely within the preamble. In this phase, developers must define the singular characters that will map to their underlying Python functional equivalents. This is achieved using the @v directive, representing an ontological declaration of variable instantiation.

For instance, the assignment of the standard output print function to the Kanji character 表 is achieved via the exact syntax: @v 表 'print'

This definitional sequence proceeds iteratively, allowing the programmer to build a complete lexicon of fundamental computational atoms , until the termination operator, denoted by the \$ symbol, is encountered. The inclusion of the \$ acts as a strict state-machine boundary condition, instantly transitioning the abstract syntax tree (AST) parser from a state of definitional absorption into a state of rigid structural enforcement.

## The Rigid Enforcement of the Body Phase

Following the \$ boundary, the compiler algorithm actively rejects any identifier whose length exceeds exactly one character. String literals are technically permitted (utilizing double quotes), but they are subjected to an identical constraint: they must evaluate to a single character during runtime expansion inside the body. To fully illustrate this architectural constraint, the following mapping table delineates the core logic and control flow identifiers required to construct a Turing-complete application—specifically, the classical FizzBuzz algorithm—in py1 :

Kanji Token	Semantic Evaluation	Ontological Purpose
表	'print'	Standard output routing and visualization
字	'str'	Lexical type casting and string conversion
循	'while'	Indefinite cyclic execution and iteration
も	'if'	Primary conditional bifurcation
或	'elif'	Secondary conditional bifurcation
他	'else'	Fallback conditional routing
剩	'%'	Modulo arithmetic evaluation
等	'=='	Boolean equality assertion
足	'+'	Arithmetic summation
小	'<'	Strict inequality assertion
数	'i'	Ephemeral loop iterator variable
ド	'\$'	Phase transition terminator

## Numeric and Literal Representation Constraints

The language's commitment to single-character representation extends unconditionally to numeric literals and system constants. Multi-digit integers, which inherently violate the single-character constraint, must be aliased to single Kanji characters during the Definition Phase. This requires the programmer to explicitly define the mathematical bounds of their program before execution begins.

Kanji Token	Integer/Literal	Execution Context
零	0	Mathematical zero, boolean false baseline
壹	1	Mathematical one, iteration increment
三	3	Fizz constraint modulo divisor
五	5	Buzz constraint modulo divisor
拾	15	FizzBuzz combined constraint divisor
佰	101	Upper iterative bound for loop termination
泡	"Fizz"	String literal buffer representation

Kanji Token	Integer/Literal	Execution Context
譽	"Buzz"	String literal buffer representation

By enforcing this strict bijective mapping, the source file's token stream becomes highly visually symmetric but entirely incomprehensible to developers uninitiated in the specific character mapping dictionary. This satisfies the core criteria for computational heresy by alienating the reader while executing flawlessly on the machine level. The resulting code structure resembles an ancient manuscript more than a modern application , pushing the aesthetic boundary of what constitutes a valid executable text.

## Ouroboric Idempotency: A Fixed-Point Theory of Self-Hosting

A core metric of validity for any novel language within the broader computer science discipline is the realization of a self-hosting compiler—a compiler capable of processing its own source code to produce a functionally identical executable version of itself. This effectively solves the recursive "chicken-or-egg" dilemma inherent in systems programming, a rite of passage that proves the language is robust enough to handle its own complex abstract syntax trees. The py1 language establishes its supreme academic legitimacy by achieving a fully self-hosted, ouroboric state across a meticulously documented three-stage mathematical pipeline.

### The Three-Stage Bootstrap Generation Pipeline

The generation of the compiler operates across distinct chronological epochs, formalized in the deployment pipeline defined within the project's continuous integration YAML specifications. The objective of this process is to verify that the compiler logic reaches a steady-state equilibrium, meaning subsequent compilations yield a bit-for-bit identical artifact, a concept deeply rooted in fixed-point theory.

The first stage, representing the transition from the old world, utilizes an external, trusted Python 3.10.11 environment to evaluate the preliminary compiler (py1.py) against the source code of the compiler written in py1 itself (compiler.py1). This initial execution yields compiler\_gen2.py, representing the first native translation of the logic. To enforce environmental normalization and prevent transient entropy from polluting the output, the generated artifact is subjected to carriage-return normalization using the dos2unix utility, followed by AST structural normalization via the black code formatter. These formatting steps act as entropic stabilizers, collapsing the quantum wave function of the source code into a purely deterministic state.

In the second stage, the newly minted compiler\_gen2.py is tasked with compiling the original compiler.py1 source code. This is the critical moment of self-hosting; the compiler must prove it can accurately parse and evaluate its own esoteric syntax without introducing semantic drift. The output is rendered as compiler\_gen3.py, which is subsequently cleansed through the exact same dos2unix and black stabilization routines.

The third and final stage is the cryptographic verification of semantic immutability. To ascertain that the language has reached a structural fixed-point , a strict idempotency validation is performed. The verification relies on the generation of SHA-256 cryptographic hashes for both compiler\_gen2.py and compiler\_gen3.py.

The empirical logs demonstrate the flawless execution of this verification: sha256sum

```
compiler_gen2.py > gen2.sha256 sha256sum compiler_gen3.py > gen3.sha256 SUCCESS:  
Byte-level reproducibility achieved.
```

This byte-level convergence empirically proves that  $\text{Compiler}(\text{Compiler}(\text{Source})) \equiv \text{Compiler}(\text{Source})$ . In the lexicon of continuous mathematics and complete partial orders (CPOs), the py1 compiler has successfully traversed the infinite chain of nested abstractions and arrived safely at its least upper bound. While formal verification of a realistic compiler typically requires vast specifications and thousands of lines of proofs in frameworks like Coq , py1 relies on brutalist, direct hash collisions to definitively assert that its operational semantics have maintained complete fidelity across generations.

## The "Trusting Trust" Implications of the Golden Chain

Ken Thompson's seminal Turing Award lecture, "Reflections on Trusting Trust" , posited that a compiler could harbor a sourceless backdoor completely undetectable by source-level audits, injecting malicious payloads during the compilation phase. By repeatedly bootstrapping py1 through an ouroboric cycle—termed the "Final Golden Chain" in the system logs —and guaranteeing fixed-point convergence, the architecture actively defends against such transient injection attacks. The structural normalization ensures that no hidden anomalies propagate through the lexical nodes , neutralizing rogue logic that might rely on whitespace manipulation or carriage return permutations to hide malicious intent. The compiler has reached a fixed point where Stage 2 and Stage 3 are bit-perfectly identical, proving the ecosystem is sealed and pure.

## Transpilation and Windows Native Intermediate Representation (IR)

Having irrefutably established the self-hosting credibility of the compiler, the py1 framework shifts its paradigm from high-level Python transpilation to native low-level machine execution. Transpilation idempotency ensures that translating from the high-level esoteric domain to an Intermediate Representation (IR) does not alter the underlying functional logic, a requirement strictly mandated in formal verification literature.

### Lexical Translation to Abstract Opcodes

The `win_ir_spec.py1` module is responsible for translating the abstract syntax tree into a highly optimized, flat linear sequence of operations reminiscent of traditional assembly code, yet entirely formatted within py1's single-character ideological constraints. Standard x64 hardware registers are aggressively obfuscated using Kanji aliases to maintain the philosophical purity of the language and obscure the program's intent from passive observation.

Kanji Token	x64 Register Equivalency	Architectural Function
蓄	RAX	Primary Accumulator / Return Value storage
壱	RBX	Base Register / Primary Iteration state
繰	RCX	Counter Register / Microsoft ABI Argument 1
夕	RDX	Data Register / Microsoft ABI

Kanji Token	x64 Register Equivalency	Architectural Function
		Argument 2
蜂	R8	General Purpose / Microsoft ABI Argument 3
苦	R9	General Purpose / Microsoft ABI Argument 4
式	R12	General Purpose Callee-Saved Register
肆	R13	General Purpose Handle Storage

To actively prevent automated signature analysis and subvert decompiler heuristics, opcodes are constructed dynamically rather than declared statically. Instead of a bare, easily identifiable string like "MOV", the language initializes the variable 置 by concatenating strings at runtime: @v 置 "M"+ "O" + "V". This concatenative opcode entropy mechanism deliberately fragments the literal pool, forcing any reverse-engineering attempt to perform full symbolic execution simply to resolve the static instruction set. The generation of IR instructions—such as LOAD (連), GET (得), CMP (mapped to 比), and JZ (mapped to 零)—allows the abstract esoteric logic to be perfectly mapped to low-level CPU branching mechanisms.

## The "安" Mock Virtual Machine Architecture

Before generating the final bare-metal assembly binary, the py1 environment leverages a custom Native VM (`vm_win_mock.py1`) to execute the IR in a sandboxed state. Because arbitrary memory writing and simulated register states can easily lead to unhandled exceptions and catastrophic segmentation faults, the developers instituted a highly robust safety helper function within the VM, appropriately denoted by the character 安 (meaning 'safe' or 'peaceful'). The 安(鍵, レ) function operates as a strict bounds-checked access mediator for the virtual registers (レ) and memory space (メ). It mathematically guarantees that any uninitialized registry keys default to a zero state, preventing catastrophic failure within the interpreter framework. When executing simulated Windows API calls (such as `WriteFile`, dynamically aliased as ラ), the VM intercepts the generated output buffer (器) and flushes it safely to the standard console under the prefix Mock::.

The empirical outputs generated by this mock VM explicitly validate the algorithmic correctness of the underlying IR before it touches the silicon, ensuring that the semantic translation from esoteric python to low-level pseudo-assembly is entirely lossless.

## Complete Eradication of the C Runtime (UCRT Bypass)

A ubiquitous point of failure and architectural dependency in modern Windows systems programming is the reliance on the Universal C Runtime (UCRT). Modern compilation pipelines heavily bind executables to specific versions of `vcruntime140.dll` or `ucrtbase.dll`, injecting massive amounts of boilerplate code to handle setup and teardown procedures. From a philosophical standpoint, relying on a massive, bloated runtime library fundamentally contradicts the foundational ethos of lexical minimalism and esoteric independence.

To overcome this existential dependency, the py1 native compiler pipeline (`compiler_native.py1`)

undertakes a radical bypass strategy: it completely eliminates the C standard library from its compilation target.

## Direct Kernel32 Subsystem Calls

Rather than calling standard high-level functions like printf or malloc, which inherently entangle the compiled binary with the UCRT, the generated assembly interfaces exclusively and directly with the Windows API via kernel32.dll.

The application initialization sequence is meticulously defined as follows:

1. **Loading the Library:** The IR issues a LOAD kernel32.dll pseudo-instruction to map the core system library into the process address space.
2. **Acquiring File Handles:** The IR invokes GET GetStdHandle. The compiler manually issues a MOV RCX, -11 (where -11 is the mathematically constant identifier for STD\_OUTPUT\_HANDLE) and executes a call to the function, securely caching the resulting execution handle in the R13 (肆) register for later retrieval.
3. **Buffer Preparation:** Pre-computed ASCII byte arrays representing the output strings (e.g., 70 for 'F', 105 for 'i', 122 for 'z') are manually loaded into a statically allocated .bss memory segment (mem\_base).
4. **Writing to stdout:** The IR executes GET WriteFile. The previously cached handle is loaded into RCX, the exact memory buffer pointer is loaded into RDX, the output length is specified in R8, and a dummy variable is passed into R9 to perfectly satisfy the strict Microsoft x64 calling convention.

## Modifying the Entry Point and Stack Alignment Rules

To completely sever ties with the GNU Compiler Collection (GCC) and the MSVC mainCRTStartup wrapper—which usually handles the hidden complexity of launching a C program—the py1 compiler explicitly renames the assembly entry point from main to start. The generated assembly header enforces absolute control over the execution environment:

```
default rel
section.text
    global start
    extern GetStdHandle
    extern WriteFile
    extern ExitProcess

start:
    sub rsp, 40
```

The sub rsp, 40 instruction is a critical, highly pedantic inclusion. It is explicitly designed to allocate 32 bytes of shadow space required by the Windows x64 ABI for function parameters, plus an additional 8 bytes to maintain the mandatory 16-byte stack alignment prior to invoking external system functions. By manually managing these complex ABI requirements, the py1 native compiler allows the MSVC linker (link.exe) to link the object file directly using only the flags /subsystem:console, /entry:start, and /defaultlib:kernel32.lib. The result is a radically lightweight, pure native executable (fizzbuzz\_native.exe) completely devoid of any underlying C runtime bloat.

## Register Mapping and the RAX/AL Conundrum

During the translation phase from IR to NASM assembly, a specific optimization challenge arises concerning the accumulator RAX register (蓄). When performing a single-byte memory write (such as moving an individual ASCII character into the .bss section for output), x64 assembly strictly requires the use of the 8-bit lower half of the register (al) rather than the full 64-bit RAX. Attempting to move a 64-bit register into an 8-bit memory address results in a catastrophic operand size mismatch during assembly.

The compiler\_native.py1 logic natively incorporates an algorithmic switch to detect this operation without requiring human intervention. Upon encountering the WRITE instruction targeting the RAX alias, the transpiler dynamically downgrades the register reference to al (小), preventing assembly failure and maintaining the flawless automated pipeline.

## Empirical Evaluation and Benchmark Analysis

To ascertain the absolute correctness and execution viability of the py1 language across its various compilation stages (Python Transpilation, Mock VM Execution, and Pure Native ASM), the classical FizzBuzz algorithm was utilized as the primary benchmarking heuristic. This algorithm, while simple, rigorously tests control flow, modulo arithmetic, string manipulation, and system I/O.

### Algorithmic Tracing and Mathematical Subroutines

The FizzBuzz logic implemented in fizzbuzz\_while.py1 initializes an iterative sequence beginning at 1 and terminating strictly before 101 (佰). At each iteration, a series of modulo arithmetic evaluations (剰) are systematically performed to determine the correct output stream:

1. If the iterator  $i \bmod 15 == 0$ , the program concatenates and prints "Fizz" + "Buzz".
2. Else if the iterator  $i \bmod 3 == 0$ , it branches to print "Fizz".
3. Else if the iterator  $i \bmod 5 == 0$ , it branches to print "Buzz".
4. Otherwise, it outputs the direct string representation of  $i$ .

The logic proceeds fluidly using the previously established Kanji mappings. The modulo constraint logic necessitates highly accurate type conversion and arithmetic evaluation at the lowest level. When transpiled to Native IR, the high-level modulo operation is mapped directly to the hardware DIV (分) instruction. Because the DIV instruction in the x64 architecture implicitly utilizes the RDX:RAX register pair for division, failing to zero out the RDX register will cause a floating-point exception or erroneous mathematical result. Consequently, the native compiler explicitly injects the instruction xor rdx, rdx (清) to clear the upper dividend before execution, ensuring mathematical purity.

### Execution Trace Integrity

The comprehensive execution logs demonstrate 100% deterministic accuracy across all target platforms, proving that semantic destruction does not occur during transpilation.

In the Mock VM output environment, the intercept function correctly yields: Mock:1 Mock:2 Mock:Fizz Mock:4 Mock:Buzz ... culminating perfectly at the expected loop termination.

Simultaneously, the execution trace of the Pure Native Output (fizzbuzz\_native.exe), running directly on the Windows kernel without standard libraries, perfectly mirrors this behavior: 1 2

## Fizz 4 Buzz

The flawless synchronization between the Mock VM's virtual, sandboxed execution and the actual physical CPU execution verifies the structural validity of the py1 Intermediate Representation. Despite the immense layers of architectural abstraction—translating from standard Python, into a single-character esoteric AST, through a self-hosted generated compiler, down to Win64 IR, and finally resolving as unmanaged, raw assembly—the semantic payload remains totally uncorrupted and mathematically verifiable.

## Advanced Mathematical Capabilities

While FizzBuzz serves as an adequate baseline evaluation metric, the native compiler possesses advanced math processing features required for complex algorithmic synthesis. These are implemented through native opcodes specifically mapped for Subtraction (SUB / 引), Multiplication (MUL / 掛), Division (DIV / 割), and Bitwise XOR (XOR / 排). Furthermore, by fully integrating real, hardware-level stack operations (PUSH / 真 and POP / 抜), the system establishes a robust foundation for complex procedural calculus and deep recursive execution, intentionally avoiding the severe memory constraints typically found in parody or toy languages.

## Philosophical Implications and Future Work

The realization of the py1 compiler proves conclusively that an esoteric programming dialect can transcend the trivial boundaries of a simple conceptual joke or mere aesthetic experimentation and evolve into a mathematically rigid, natively executing, self-hosting powerhouse.

The total abolition of the Universal C Runtime establishes a new operational baseline for minimal executable footprints. By completely controlling the application entry point and ABI stack alignments, py1 demonstrates that modern software engineering has become overly reliant on bloated, opaque standard libraries. Furthermore, the Kanji-based syntax introduces a highly novel method of structural obfuscation. Because the entire execution environment relies on an extreme form of lexical parsimony, traditional heuristic analysis and reverse engineering tools are rendered largely ineffective.

Future research within this specific domain of computational heresy will explore the implementation of zero-character variable names (leveraging invisible zero-width Unicode joiners and non-printing spaces) to further push lexical density beyond the visible spectrum. Additionally, applying this specific compiler architecture to quantum disadvantage simulations on vintage hardware architectures remains a highly promising avenue for subsequent SIGBOVIK submissions.

## Conclusion

The present analysis has detailed the profound theoretical underpinnings and unprecedented empirical success of py1, a uniquely constrained, self-hosting programming environment. By seamlessly integrating the principles of Shannon entropy, formal fixed-point idempotency verification, and extreme lexical parsimony, py1 successfully bridges the vast gap between humorous obfuscation and robust, low-level systems engineering. The successful generation of a pure Native x64 executable completely devoid of C Runtime dependencies demonstrates that semantic depth is not inextricably linked to syntactical verbosity. Ultimately, this comprehensive

body of work stands as an irrefutable testament to the fact that in the realm of computational heresy, reducing the complex identifier to a single character does not diminish its computational power—it magnifies its absolute, foundational meaning.

1. SIGBOVIK 2019, <https://sigbovik.org/2019/proceedings.pdf>
2. Meaningful Identifier Names: The Case of Single-Letter Variables - The Rachel and Selim Benin School of Computer Science and Engineering, <https://www.cs.huji.ac.il/~feit/papers/SingleLetter17ICPC.pdf>
3. Minimalism in Programming Language Design - Pointers Gone Wild, <https://pointersgonewild.com/2022/05/23/minimalism-in-programming-language-design/>
4. SIGBOVIK: The Ig Nobel for Academics and Computer Science Researchers, <https://www.cesarsotovalero.net/blog/sigbovik-the-ig-nobel-for-academics-and-computer-science-researchers.html>
5. SIGBOVIK 0x2023, <https://sigbovik.org/2023/proceedings.pdf>
6. Esoteric programming language - Wikipedia, [https://en.wikipedia.org/wiki/Esoteric\\_programming\\_language](https://en.wikipedia.org/wiki/Esoteric_programming_language)
7. Let's Take Esoteric Programming Languages Seriously - arXiv, <https://arxiv.org/html/2505.15327v2>
8. The Hacker Folk Art of Esoteric Coding - The MIT Press Reader, <https://thereader.mitpress.mit.edu/the-hacker-folk-art-of-esoteric-coding/>
9. In search of the Ballmer Peak, and other results from SIGBOVIK 2024 - The Old New Thing, <https://devblogs.microsoft.com/oldnewthing/20240416-01/?p=109672>
10. SIGBOVIK 2024, <https://sigbovik.org/2024/proceedings.pdf>
11. Infinite golfing - Esolang, [https://esolangs.org/wiki/Infinite\\_golfing](https://esolangs.org/wiki/Infinite_golfing)
12. (PDF) Protecting Quantum Circuits Through Compiler-Resistant Obfuscation - ResearchGate, [https://www.researchgate.net/publication/398979586\\_Protecting\\_Quantum\\_Circuits\\_Through\\_Compiler-Resistant\\_Obfuscation](https://www.researchgate.net/publication/398979586_Protecting_Quantum_Circuits_Through_Compiler-Resistant_Obfuscation)
13. Co6GC: Program Obfuscation - COSIC - KU Leuven, <https://www.esat.kuleuven.be/cosic/blog/program-obfuscation/>
14. An SMT Theory of Fixed-Point Arithmetic - PMC - NIH, <https://PMC.ncbi.nlm.nih.gov/articles/PMC7324132/>
15. CS 6110 Lecture 22 The Fixed-Point Theorem 17 March 2025, <https://www.cs.cornell.edu/courses/cs6110/2025sp/lectures/fixed-pt-thm.pdf>
16. Upgrade your code to the Universal CRT | Microsoft Learn, <https://learn.microsoft.com/en-us/cpp/porting/upgrade-your-code-to-the-universal-crt?view=msvc-170>
17. How to circumvent Windows Universal CRT headers dependency on vcruntime.h, <https://stackoverflow.com/questions/45340527/how-to-circumvent-windows-universal-crt-headers-dependency-on-vcruntime-h>
18. How to get rid of ucrt under Windows platform completely? - Stack Overflow, <https://stackoverflow.com/questions/79017326/how-to-get-rid-of-ucrt-under-windows-platform-completely>
19. Static Analysis and Compiler Design for Idempotent Processing - University of Wisconsin–Madison, <https://research.cs.wisc.edu/vertical/papers/2012/pldi12-idem.pdf>
20. Why are self-hosting compilers considered a rite of passage for new languages?, <https://softwareengineering.stackexchange.com/questions/263651/why-are-self-hosting-compilers-considered-a-rite-of-passage-for-new-languages>
21. Entropy (information theory) - Wikipedia, [https://en.wikipedia.org/wiki/Entropy\\_\(information\\_theory\)](https://en.wikipedia.org/wiki/Entropy_(information_theory))
22. Information Theory: Entropy, Markov Chains, and Huffman Coding - Department of Mathematics, [https://math.nd.edu/assets/275279/leblanc\\_thesis.pdf](https://math.nd.edu/assets/275279/leblanc_thesis.pdf)
23. An Estimate of an Upper Bound for the Entropy of English - ACL Anthology, <https://aclanthology.org/J92-1002.pdf>
24. Prediction and Entropy of Printed English - Princeton University, [https://www.princeton.edu/~wbialek/rome/refs/shannon\\_51.pdf](https://www.princeton.edu/~wbialek/rome/refs/shannon_51.pdf)
25. Compiler Design: Theory,

Tools, and Examples - Rowan Digital Works,  
[https://rdw.rowan.edu/context/oer/article/1001/viewcontent/CompilerDesignMay17\\_24.pdf](https://rdw.rowan.edu/context/oer/article/1001/viewcontent/CompilerDesignMay17_24.pdf) 26.

Naming convention (programming) - Wikipedia,  
[https://en.wikipedia.org/wiki/Naming\\_convention\\_\(programming\)](https://en.wikipedia.org/wiki/Naming_convention_(programming)) 27. Using single characters for variable names in loops/exceptions - Stack Overflow,  
<https://stackoverflow.com/questions/5802403/using-single-characters-for-variable-names-in-loops-exceptions> 28. [2504.14024] Simplicity by Obfuscation: Evaluating LLM-Driven Code Transformation with Semantic Elasticity - arXiv, <https://arxiv.org/abs/2504.14024> 29. Entropy (information theory) - Wikipedia, the free encyclopedia,  
[http://home.zcu.cz/~potmesil/ADM%202015/4%20Regrese/Coefficients%20-%20Gamma%20Tau%20etc./Z-Entropy%20\(information%20theory\)%20-%20Wikipedia.htm](http://home.zcu.cz/~potmesil/ADM%202015/4%20Regrese/Coefficients%20-%20Gamma%20Tau%20etc./Z-Entropy%20(information%20theory)%20-%20Wikipedia.htm) 30. A syntax-lexicon trade-off in language production - PMC, <https://PMC.ncbi.nlm.nih.gov/articles/PMC9231468/> 31.

Lexical Density, and Other Bedtime Stories - the Appsmith Community,  
<https://community.appsmith.com/content/blog/lexical-density-and-other-bedtime-stories> 32.

Lexical density - Wikipedia, [https://en.wikipedia.org/wiki/Lexical\\_density](https://en.wikipedia.org/wiki/Lexical_density) 33. Parsimony: An IDE for Example-Guided Synthesis of Lexers and Parsers - Cornell: Computer Science,  
<https://www.cs.cornell.edu/~lerner/papers/parsimony-ase2017.pdf> 34. Learning Lexical Features of Programming Languages from Imagery Using Convolutional Neural Networks - Chapman University, <http://www1.chapman.edu/~linstead/ottICPC2018.pdf> 35. Design of efficient Programming Language with Lexer using '\$'-prefixed identifier - EAI Endorsed Transactions, <https://publications.eai.eu/index.php/sis/article/download/3920/2522/25253> 36. Joke language list - Esolang, [https://esolangs.org/wiki/Joke\\_language\\_list](https://esolangs.org/wiki/Joke_language_list) 37. What do you call it when someone programs in an overly obfuscated way such that only they will ever be able to understand it? - Reddit,  
[https://www.reddit.com/r/softwaredevelopment/comments/1gdnxyo/what\\_do\\_you\\_call\\_it\\_when\\_someone\\_programs\\_in\\_an/](https://www.reddit.com/r/softwaredevelopment/comments/1gdnxyo/what_do_you_call_it_when_someone_programs_in_an/) 38. Obfuscation (software) - Wikipedia,  
[https://en.wikipedia.org/wiki/Obfuscation\\_\(software\)](https://en.wikipedia.org/wiki/Obfuscation_(software)) 39. Code obfuscation - Security Software Glossary - Promon, <https://promon.io/resources/security-software-glossary/code-obfuscation> 40. A key concern I've consistently had regarding formal verification systems is: ho... | Hacker News, <https://news.ycombinator.com/item?id=43550402> 41. List of ideas - Esolang, [https://esolangs.org/wiki/List\\_of\\_ideas](https://esolangs.org/wiki/List_of_ideas) 42. The Evolvability of Words: On the Nature of Lexical Items in Minimalism - Frontiers, <https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2019.03071/full> 43. The Evolvability of Words: On the Nature of Lexical Items in Minimalism - PMC - NIH, <https://PMC.ncbi.nlm.nih.gov/articles/PMC6992612/> 44. SIGBOVIK 2025, <https://sigbovik.org/2025/proceedings.pdf> 45. Compiler-compiler - Wikipedia, <https://en.wikipedia.org/wiki/Compiler-compiler> 46. Bootstrapping (compilers) - Wikipedia, [https://en.wikipedia.org/wiki/Bootstrapping\\_\(compilers\)](https://en.wikipedia.org/wiki/Bootstrapping_(compilers)) 47. Self-hosting (compilers) - Wikipedia, [https://en.wikipedia.org/wiki/Self-hosting\\_\(compilers\)](https://en.wikipedia.org/wiki/Self-hosting_(compilers)) 48. Bootstrapping and self-hosting - Tom Mewett, <https://tmewett.com/bootstrapping-metacompiling/> 49. Running the "Reflections on Trusting Trust" Compiler - ACM Queue, <https://queue.acm.org/detail.cfm?id=3786614> 50. Special Issue: Fixed-Point Theory and Its Applications, Dedicated to the Memory of Professor William Arthur Kirk - MDPI, <https://www.mdpi.com/2073-8994/16/11/1408> 51. (PDF) A New Computer Science Academic Word List - ResearchGate, [https://www.researchgate.net/publication/369877950\\_A\\_New\\_Computer\\_Science\\_Academic\\_Word\\_List](https://www.researchgate.net/publication/369877950_A_New_Computer_Science_Academic_Word_List) 52. Compiler writers who have undergone self-hosting: any tips for keeping it all straight? : r/ProgrammingLanguages - Reddit, [https://www.reddit.com/r/ProgrammingLanguages/comments/5kv2t0/compiler\\_writers\\_who\\_hav](https://www.reddit.com/r/ProgrammingLanguages/comments/5kv2t0/compiler_writers_who_hav)

e\_undergone\_selfhosting/ 53. [1201.4719] On Synergy of Metal, Slicing, and Symbolic Execution, <https://arxiv.labs.arxiv.org/html/1201.4719> 54. Fucking Shell Scripts - Hacker News, <https://news.ycombinator.com/item?id=7401803> 55. Formal Verification of a Realistic Compiler - Communications of the ACM, <https://cacm.acm.org/research/formal-verification-of-a-realistic-compiler/> 56. CS 6120: Formal Verification of a Realistic Compiler, <https://www.cs.cornell.edu/courses/cs6120/2022sp/blog/compcert/> 57. Formal verification of program obfuscations, <https://mw.hh.se/wg211/images/1/1d/M15Blazy-Slides.pdf> 58. Formal Specification and Verification of Secure Information Flow for Hardware Platforms - UC Berkeley EECS, <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-224.pdf> 59. SIGBOVIK 2022, <https://sigbovik.org/2022/proceedings.pdf> 60. Formal Verification of Transcompiled Mobile Applications Using First-Order Logic - MDPI, <https://www.mdpi.com/2227-7080/13/12/580> 61. Formally speaking, "Transpiler" is a useless word | Rachit Nigam - People, <https://people.csail.mit.edu/rachit/post/transpiler-formal/> 62. Computational perspectives on minimalism - Linguistics - UCLA, <https://linguistics.ucla.edu/people/stabler/Stabler10-Min.pdf> 63. Minimalism in Programming Language Design - Reddit, [https://www.reddit.com/r/programming/comments/uwntz1/minimalism\\_in\\_programming\\_language\\_design/](https://www.reddit.com/r/programming/comments/uwntz1/minimalism_in_programming_language_design/) 64. Understanding Windows x64 Assembly - GitHub Pages, [https://sonictk.github.io/asm\\_tutorial/](https://sonictk.github.io/asm_tutorial/) 65. - Esolang, the esoteric programming languages wiki, <https://esolangs.org/wiki/%E2%84%B2> 66. Compiler with support for formal verification and mathematical proofs of code - Reddit, [https://www.reddit.com/r/ProgrammingLanguages/comments/jysev8/compiler\\_with\\_support\\_for\\_formal\\_verification\\_and/](https://www.reddit.com/r/ProgrammingLanguages/comments/jysev8/compiler_with_support_for_formal_verification_and/)

## YML execution log excerpt in github actions environment

```
2026-01-11T07:35:56.0325382Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}
2026-01-11T07:35:56.0326442Z env:
2026-01-11T07:35:56.0326614Z PYTHONIOENCODING: utf-8
2026-01-11T07:35:56.0326805Z PYTHONUTF8: 1
2026-01-11T07:35:56.0326970Z PYTHONUNBUFFERED: 1
2026-01-11T07:35:56.0327241Z pythonLocation: C:
|hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:56.0327691Z PKG_CONFIG_PATH: C:
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig
2026-01-11T07:35:56.0328116Z Python_ROOT_DIR: C:
|hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:56.0328500Z Python2_ROOT_DIR: C:
|hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:56.0328862Z Python3_ROOT_DIR: C:
|hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:56.0329149Z ##[endgroup]
2026-01-11T07:35:56.1615344Z SUCCESS: Byte-level reproducibility achieved.
2026-01-11T07:35:59.8396362Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}
2026-01-11T07:35:59.8396729Z env:
2026-01-11T07:35:59.8396892Z PYTHONIOENCODING: utf-8
2026-01-11T07:35:59.8397095Z PYTHONUTF8: 1
2026-01-11T07:35:59.8397260Z PYTHONUNBUFFERED: 1
2026-01-11T07:35:59.8397529Z pythonLocation: C:
|hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:59.8397944Z PKG_CONFIG_PATH: C:
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig
2026-01-11T07:35:59.8398359Z Python_ROOT_DIR: C:
|hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:59.8398757Z Python2_ROOT_DIR: C:
|hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:59.8399146Z Python3_ROOT_DIR: C:
|hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:59.8399437Z ##[endgroup]
2026-01-11T07:36:00.0736951Z 1
2026-01-11T07:36:00.0737201Z 2
2026-01-11T07:36:00.0737707Z Fizz
2026-01-11T07:36:00.0737957Z 4
```

2026-01-11T07:36:00.0738309Z Buzz  
2026-01-11T07:36:00.0739103Z Fizz  
2026-01-11T07:36:00.0739736Z 7  
2026-01-11T07:36:00.0740410Z 8  
2026-01-11T07:36:00.0741001Z Fizz  
2026-01-11T07:36:00.0742030Z Buzz  
2026-01-11T07:36:00.0742328Z 11  
2026-01-11T07:36:00.0742558Z Fizz  
2026-01-11T07:36:00.0743331Z 13  
2026-01-11T07:36:00.0743940Z 14  
2026-01-11T07:36:00.0744458Z FizzBuzz  
2026-01-11T07:36:00.0745105Z 16  
2026-01-11T07:36:00.0745742Z 17  
2026-01-11T07:36:00.0746332Z Fizz  
2026-01-11T07:36:00.0746975Z 19  
2026-01-11T07:36:00.0747610Z Buzz  
2026-01-11T07:36:00.0748200Z Fizz  
2026-01-11T07:36:00.0748847Z 22  
2026-01-11T07:36:00.0749499Z 23  
2026-01-11T07:36:00.0750072Z Fizz  
2026-01-11T07:36:00.0750704Z Buzz  
2026-01-11T07:36:00.0751359Z 26  
2026-01-11T07:36:00.0751940Z Fizz  
2026-01-11T07:36:00.0752590Z 28  
2026-01-11T07:36:00.0753955Z 29  
2026-01-11T07:36:00.0754277Z FizzBuzz  
2026-01-11T07:36:00.0754515Z 31  
2026-01-11T07:36:00.0754780Z 32  
2026-01-11T07:36:00.0755697Z Fizz  
2026-01-11T07:36:00.0755886Z 34  
2026-01-11T07:36:00.0756662Z Buzz  
2026-01-11T07:36:00.0757230Z Fizz  
2026-01-11T07:36:00.0757738Z 37  
2026-01-11T07:36:00.0758547Z 38  
2026-01-11T07:36:00.0758881Z Fizz  
2026-01-11T07:36:00.0760614Z Buzz  
2026-01-11T07:36:00.0761535Z 41  
2026-01-11T07:36:00.0762148Z Fizz  
2026-01-11T07:36:00.0763025Z 43  
2026-01-11T07:36:00.0763260Z 44  
2026-01-11T07:36:00.0763480Z FizzBuzz

2026-01-11T07:36:00.0763743Z 46  
2026-01-11T07:36:00.0764825Z 47  
2026-01-11T07:36:00.0765069Z Fizz  
2026-01-11T07:36:00.0765530Z 49  
2026-01-11T07:36:00.0766444Z Buzz  
2026-01-11T07:36:00.0766748Z Fizz  
2026-01-11T07:36:00.0767739Z 52  
2026-01-11T07:36:00.0768024Z 53  
2026-01-11T07:36:00.0768879Z Fizz  
2026-01-11T07:36:00.0769273Z Buzz  
2026-01-11T07:36:00.0770208Z 56  
2026-01-11T07:36:00.0770483Z Fizz  
2026-01-11T07:36:00.0771226Z 58  
2026-01-11T07:36:00.0772148Z 59  
2026-01-11T07:36:00.0773603Z FizzBuzz  
2026-01-11T07:36:00.0773908Z 61  
2026-01-11T07:36:00.0774136Z 62  
2026-01-11T07:36:00.0774348Z Fizz  
2026-01-11T07:36:00.0775084Z 64  
2026-01-11T07:36:00.0775566Z Buzz  
2026-01-11T07:36:00.0776085Z Fizz  
2026-01-11T07:36:00.0776580Z 67  
2026-01-11T07:36:00.0776913Z 68  
2026-01-11T07:36:00.0777232Z Fizz  
2026-01-11T07:36:00.0777947Z Buzz  
2026-01-11T07:36:00.0778680Z 71  
2026-01-11T07:36:00.0780520Z Fizz  
2026-01-11T07:36:00.0780916Z 73  
2026-01-11T07:36:00.0781263Z 74  
2026-01-11T07:36:00.0781936Z FizzBuzz  
2026-01-11T07:36:00.0782282Z 76  
2026-01-11T07:36:00.0782619Z 77  
2026-01-11T07:36:00.0783301Z Fizz  
2026-01-11T07:36:00.0784590Z 79  
2026-01-11T07:36:00.0785510Z Buzz  
2026-01-11T07:36:00.0786105Z Fizz  
2026-01-11T07:36:00.0786443Z 82  
2026-01-11T07:36:00.0790259Z 83  
2026-01-11T07:36:00.0790933Z Fizz  
2026-01-11T07:36:00.0791310Z Buzz  
2026-01-11T07:36:00.0791450Z 86

2026-01-11T07:36:00.0791578Z Fizz  
2026-01-11T07:36:00.0791810Z 88  
2026-01-11T07:36:00.0792020Z 89  
2026-01-11T07:36:00.0792154ZFizzBuzz  
2026-01-11T07:36:00.0792290Z 91  
2026-01-11T07:36:00.0792494Z 92  
2026-01-11T07:36:00.0792738Z Fizz  
2026-01-11T07:36:00.0793635Z 94  
2026-01-11T07:36:00.0794362Z Buzz  
2026-01-11T07:36:00.0795048Z Fizz  
2026-01-11T07:36:00.0796043Z 97  
2026-01-11T07:36:00.0796438Z 98  
2026-01-11T07:36:00.0797278Z Fizz  
2026-01-11T07:36:00.0797941Z Buzz  
2026-01-11T07:36:08.0104814Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}  
2026-01-11T07:36:08.0105184Z env:  
2026-01-11T07:36:08.0105345Z PYTHONIOENCODING: utf-8  
2026-01-11T07:36:08.0105543Z PYTHONUTF8: 1  
2026-01-11T07:36:08.0105714Z PYTHONUNBUFFERED: 1  
2026-01-11T07:36:08.0105976Z pythonLocation: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:08.0106388Z PKG\_CONFIG\_PATH: C:  
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig  
2026-01-11T07:36:08.0107180Z Python\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:08.0107605Z Python2\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:08.0107975Z Python3\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:08.0108258Z ##[endgroup]  
2026-01-11T07:36:08.0457557Z === Ouroboros Test: 3-Stage Bootstrap Verification ===  
2026-01-11T07:36:08.0458230Z Reason: The new compiler preserves comments, while the bootstrap one didn't.  
2026-01-11T07:36:08.0458696Z We must verify stability between Stage 2 and Stage 3.  
2026-01-11T07:36:08.0458999Z Generating Stage 1...  
2026-01-11T07:36:08.1059702Z Generating Stage 2...  
2026-01-11T07:36:08.1489523Z Generating Stage 3...  
2026-01-11T07:36:08.2123881Z dos2unix: converting file stage2\_compiler.py to Unix format...  
2026-01-11T07:36:08.2336228Z dos2unix: converting file stage3\_compiler.py to Unix format...  
2026-01-11T07:36:08.3041514Z Stage 2:

fab66e6b2fc7ebfc2c744ccd753b5afbdde2ad316cb5923004cafbdbe27d2bb  
2026-01-11T07:36:08.3058240Z Stage 3:  
fab66e6b2fc7ebfc2c744ccd753b5afbdde2ad316cb5923004cafbdbe27d2bb  
2026-01-11T07:36:08.3060566Z SUCCESS: The compiler has reached a fixed point.  
2026-01-11T07:36:08.3061672Z Stage 2 and Stage 3 are bit-perfectly identical.  
2026-01-11T07:36:17.3607890Z env:  
2026-01-11T07:36:17.3608064Z PYTHONIOENCODING: utf-8  
2026-01-11T07:36:17.3608274Z PYTHONUTF8: 1  
2026-01-11T07:36:17.3608444Z PYTHONUNBUFFERED: 1  
2026-01-11T07:36:17.3608717Z pythonLocation: C:  
\hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:17.3609155Z PKG\_CONFIG\_PATH: C:  
\hostedtoolcache\windows\Python\3.10.11\x64/lib/pkgconfig  
2026-01-11T07:36:17.3609575Z Python\_ROOT\_DIR: C:  
\hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:17.3610124Z Python2\_ROOT\_DIR: C:  
\hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:17.3610506Z Python3\_ROOT\_DIR: C:  
\hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:17.3610817Z CommandPromptType: Native  
2026-01-11T07:36:17.3611144Z DevEnvDir: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\Common7\IDE\  
2026-01-11T07:36:17.3611658Z ExtensionSdkDir: C:\Program Files (x86)\Microsoft SDKs\Windows Kits\10\ExtensionSDKs  
2026-01-11T07:36:17.3614015Z EXTERNAL\_INCLUDE: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VC\Tools\MSVC\14.44.35207\include;C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VC\Tools\MSVC\14.44.35207\ATLMFC\include;C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VC\Auxiliary\VS\include;C:\Program Files (x86)\Windows Kits\10\include\10.0.26100.0\ucrt;C:\Program Files (x86)\Windows Kits\10\include\10.0.26100.0\um;C:\Program Files (x86)\Windows Kits\10\include\10.0.26100.0\shared;C:\Program Files (x86)\Windows Kits\10\include\10.0.26100.0\cppwinrt;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.8\include\um  
2026-01-11T07:36:17.3616216Z Framework40Version: v4.0  
2026-01-11T07:36:17.3616472Z FrameworkDir: C:\Windows\Microsoft.NET\Framework64\  
2026-01-11T07:36:17.3616801Z FrameworkDir64: C:\Windows\Microsoft.NET\Framework64\  
2026-01-11T07:36:17.3617075Z FrameworkVersion: v4.0.30319  
2026-01-11T07:36:17.3617293Z FrameworkVersion64: v4.0.30319  
2026-01-11T07:36:17.3617800Z FSHARPISTALLDIR: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\Common7\IDE\CommonExtensions\Microsoft\FSharp\Tools  
2026-01-11T07:36:17.3618371Z HTMLHelpDir: C:\Program Files (x86)\HTML Help Workshop  
2026-01-11T07:36:17.3619312Z IFCPATH: C:\Program Files\Microsoft Visual



\Windows\Microsoft.NET\Framework64\v4.0.30319;C:\Program Files\Microsoft Visual Studio\2022\Enterprise\Common7\IDE\;C:\Program Files\Microsoft Visual Studio\2022\Enterprise\Common7\Tools\;C:\Program Files\NASM;C:  
|Users\runneradmin\AppData\Roaming\Python\Python310\Scripts;C:  
|hostedtoolcache\windows\Python\3.10.11\x64\Scripts;C:  
|hostedtoolcache\windows\Python\3.10.11\x64;C:\Program Files\MongoDB\Server\7.0\bin;C:  
\vcpkg;C:\tools\zstd;C:\hostedtoolcache\windows\stack\3.9.1\x64;C:\cabal\bin;C:\ghcup\bin;C:  
\mingw64\bin;C:\Program Files\dotnet;C:\Program Files\MySQL\MySQL Server 8.0\bin;C:  
\Program Files\R\R-4.5.2\bin\x64;C:\SeleniumWebDrivers\GeckoDriver;C:  
\SeleniumWebDrivers\EdgeDriver\;C:\SeleniumWebDrivers\ChromeDriver;C:\Program Files  
(x86)\sbt\bin;C:\Program Files (x86)\GitHub CLI;C:\Program Files\Git\bin;C:\Program Files  
(x86)\pipx\_bin;C:\npm\prefix;C:\hostedtoolcache\windows\go\1.24.11\x64\bin;C:  
|hostedtoolcache\windows\Python\3.9.13\x64\Scripts;C:  
|hostedtoolcache\windows\Python\3.9.13\x64;C:  
|hostedtoolcache\windows\Ruby\3.3.10\x64\bin;C:\Program Files\OpenSSL\bin;C:  
\tools\kotlinc\bin;C:\hostedtoolcache\windows\Java\_Temurin-Hotspot\_jdk\17.0.17-10\x64\bin;C:  
\Program Files\ImageMagick-7.1.2-Q16-HDRI;C:\Program Files\Microsoft  
SDKs\Azure\CLI2\wbin;C:\ProgramData\kind;C:\ProgramData\Chocolatey\bin;C:  
\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:  
\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program  
Files\PowerShell\7\;C:\Program Files\Microsoft\Web Platform Installer\;C:\Program Files\Microsoft  
SQL Server\Client SDK\ODBC\170\Tools\Binn\;C:\Program Files\Microsoft SQL  
Server\150\Tools\Binn\;C:\Program Files\dotnet\;C:\Program Files (x86)\Windows Kits\10\Windows  
Performance Toolkit\;C:\Program Files (x86)\WiX Toolset v3.14\bin;C:\Program Files\Microsoft  
SQL Server\130\DT\Binn\;C:\Program Files\Microsoft SQL Server\140\DT\Binn\;C:\Program  
Files\Microsoft SQL Server\150\DT\Binn\;C:\Program Files\Microsoft SQL  
Server\160\DT\Binn\;C:\Program Files\Microsoft SQL Server\170\DT\Binn\;C:  
\ProgramData\chocolatey\lib\pulumi\tools\Pulumi\bin;C:\Program Files\CMake\bin;C:  
\Strawberry\c\bin;C:\Strawberry\perl\site\bin;C:\Strawberry\perl\bin;C:  
\ProgramData\chocolatey\lib\maven\apache-maven-3.9.12\bin;C:\Program Files\Microsoft Service  
Fabric\bin\Fabric\Fabric.Code;C:\Program Files\Microsoft SDKs\Service  
Fabric\Tools\ServiceFabricLocalClusterManager;C:\Program Files\nodejs\;C:\Program  
Files\Git\cmd;C:\Program Files\Git\mingw64\bin;C:\Program Files\Git\usr\bin;C:\Program  
Files\GitHub CLI\;c:\tools\php;C:\Program Files\Amazon\AWSCLIV2\;C:\Program  
Files\Amazon\SessionManagerPlugin\bin\;C:\Program Files\Amazon\AWSSAMCLI\bin\;C:\Program  
Files\Microsoft SQL Server\130\Tools\Binn\;C:\Program Files\mongosh\;C:\Program  
Files\LLVM\bin;C:\Program Files (x86)\LLVM\bin;C:\Users\runneradmin\.dotnet\tools;C:  
\Users\runneradmin\.cargo\bin;C:\Users\runneradmin\AppData\Local\Microsoft\WindowsApps;C:  
\Program Files (x86)\Microsoft Visual Studio\Installer;C:\Program Files\Microsoft Visual  
Studio\2022\Enterprise\Common7\IDE\CommonExtensions\Microsoft\CMake\CMake\bin;C:  
\Program Files\Microsoft Visual

Studio\2022\Enterprise\Common7\IDE\CommonExtensions\Microsoft\CMake\Ninja;C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VC\Linux\bin\ConnectionManagerExe;C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VC\vcpkg  
2026-01-11T07:36:17.3664081Z Platform: x64  
2026-01-11T07:36:17.3664264Z UCRTVersion: 10.0.26100.0  
2026-01-11T07:36:17.3664547Z UniversalCRTSdkDir: C:\Program Files (x86)\Windows Kits\10\  
2026-01-11T07:36:17.3665007Z VCIDEInstallDir: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\Common7\IDE\VC\  
2026-01-11T07:36:17.3665516Z VCINSTALLDIR: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VC\  
2026-01-11T07:36:17.3665970Z VCPKG\_ROOT: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VC\vcpkg  
2026-01-11T07:36:17.3666506Z VCToolsInstallDir: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VC\Tools\MSVC\14.44.35207\  
2026-01-11T07:36:17.3667134Z VCToolsRedistDir: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VC\Redist\MSVC\14.44.35112\  
2026-01-11T07:36:17.3667577Z VCToolsVersion: 14.44.35207  
2026-01-11T07:36:17.3667792Z VisualStudioVersion: 17.0  
2026-01-11T07:36:17.3668158Z VS170COMNTOOLS: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\Common7\Tools\  
2026-01-11T07:36:17.3668550Z VSCMD\_ARG\_app\_plat: Desktop  
2026-01-11T07:36:17.3668752Z VSCMD\_ARG\_HOST\_ARCH: x64  
2026-01-11T07:36:17.3668941Z VSCMD\_ARG\_TGT\_ARCH: x64  
2026-01-11T07:36:17.3669123Z VSCMD\_VER: 17.14.23  
2026-01-11T07:36:17.3669404Z VSINSTALLDIR: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\  
2026-01-11T07:36:17.3669854Z VSSDK150INSTALL: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VSSDK  
2026-01-11T07:36:17.3670323Z VSSDKINSTALL: C:\Program Files\Microsoft Visual Studio\2022\Enterprise\VSSDK  
2026-01-11T07:36:17.3670995Z WindowsLibPath: C:\Program Files (x86)\Windows Kits\10\UnionMetadata\10.0.26100.0;C:\Program Files (x86)\Windows Kits\10\References\10.0.26100.0  
2026-01-11T07:36:17.3671626Z WindowsSdkBinPath: C:\Program Files (x86)\Windows Kits\10\bin\  
2026-01-11T07:36:17.3671984Z WindowsSdkDir: C:\Program Files (x86)\Windows Kits\10\  
2026-01-11T07:36:17.3672269Z WindowsSDKLibVersion: 10.0.26100.0\  
2026-01-11T07:36:17.3672612Z WindowsSdkVerBinPath: C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\  
2026-01-11T07:36:17.3672980Z WindowsSDKVersion: 10.0.26100.0\

2026-01-11T07:36:17.3673546Z WindowsSDK\_ExecutablePath\_x64: C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8 Tools\x64\  
2026-01-11T07:36:17.3674196Z WindowsSDK\_ExecutablePath\_x86: C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8 Tools\  
2026-01-11T07:36:17.3674628Z \_\_DOTNET\_ADD\_64BIT: 1  
2026-01-11T07:36:17.3674813Z \_\_DOTNET\_PREFERRED\_BITNESS: 64  
2026-01-11T07:36:17.3684430Z \_\_VSCMD\_PREINIT\_PATH: C:\Program Files\NASM;C:\Users\runneradmin\AppData\Roaming\Python\Python310\Scripts;C:\hostedtoolcache\windows\Python\3.10.11\x64\Scripts;C:\hostedtoolcache\windows\Python\3.10.11\x64;C:\Program Files\MongoDB\Server\7.0\bin;C:\vcpkg;C:\tools\zstd;C:\hostedtoolcache\windows\stack\3.9.1\x64;C:\cabal\bin;C:\ghcup\bin;C:\mingw64\bin;C:\Program Files\dotnet;C:\Program Files\MySQL\MySQL Server 8.0\bin;C:\Program Files\R\R-4.5.2\bin\x64;C:\SeleniumWebDrivers\GeckoDriver;C:\SeleniumWebDrivers\EdgeDriver\;C:\SeleniumWebDrivers\ChromeDriver;C:\Program Files (x86)\sbt\bin;C:\Program Files (x86)\GitHub CLI;C:\Program Files\Git\bin;C:\Program Files (x86)\pipx\_bin;C:\npm\prefix;C:\hostedtoolcache\windows\go\1.24.11\x64\bin;C:\hostedtoolcache\windows\Python\3.9.13\x64\Scripts;C:\hostedtoolcache\windows\Python\3.9.13\x64;C:\hostedtoolcache\windows\Ruby\3.3.10\x64\bin;C:\Program Files\OpenSSL\bin;C:\tools\kotlinc\bin;C:\hostedtoolcache\windows\Java\_Temurin-Hotspot\_jdk\17.0.17-10\x64\bin;C:\Program Files\ImageMagick-7.1.2-Q16-HDRI;C:\Program Files\Microsoft SDKs\Azure\CLI2\wbin;C:\ProgramData\kind;C:\ProgramData\Chocolatey\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files\PowerShell\7\;C:\Program Files\Microsoft\Web Platform Installer\;C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\Tools\Binn\;C:\Program Files\Microsoft SQL Server\150\Tools\Binn\;C:\Program Files\dotnet\;C:\Program Files (x86)\Windows Kits\10\Windows Performance Toolkit\;C:\Program Files (x86)\WiX Toolset v3.14\bin;C:\Program Files\Microsoft SQL Server\130\DT\Binn\;C:\Program Files\Microsoft SQL Server\140\DT\Binn\;C:\Program Files\Microsoft SQL Server\150\DT\Binn\;C:\Program Files\Microsoft SQL Server\160\DT\Binn\;C:\Program Files\Microsoft SQL Server\170\DT\Binn\;C:\ProgramData\chocolatey\lib\pulumi\tools\Pulumi\bin;C:\Program Files\CMake\bin;C:\Strawberry\c\bin;C:\Strawberry\perl\site\bin;C:\Strawberry\perl\bin;C:\ProgramData\chocolatey\lib\maven\apache-maven-3.9.12\bin;C:\Program Files\Microsoft Service Fabric\bin\Fabric\Fabric.Code;C:\Program Files\Microsoft SDKs\Service Fabric\Tools\ServiceFabricLocalClusterManager;C:\Program Files\nodejs\;C:\Program Files\Git\cmd;C:\Program Files\Git\mingw64\bin;C:\Program Files\Git\usr\bin;C:\Program Files\GitHub CLI\;C:\tools\php;C:\Program Files (x86)\sbt\bin;C:\Program Files\Amazon\AWSCLIV2\;C:\Program Files\Amazon\SessionManagerPlugin\bin\;C:\Program Files\Amazon\AWSSAMCLI\bin\;C:\Program Files\Microsoft SQL Server\130\Tools\Binn\;C:\Program Files\mongosh\;C:\Program Files\LLVM\bin;C:\Program Files (x86)\LLVM\bin;C:

```
|Users\runneradmin\.dotnet\tools;C:\Users\runneradmin\.cargo\bin;C:  
|Users\runneradmin\AppData\Local\Microsoft\WindowsApps;C:\Program Files (x86)\Microsoft  
Visual Studio\Installer  
2026-01-11T07:36:17.3694008Z ##[endgroup]  
2026-01-11T07:36:17.3850925Z --- Compiling Native Compiler ---  
2026-01-11T07:36:17.4182680Z --- Generating Native ASM ---  
2026-01-11T07:36:17.4481689Z --- Assembling with NASM ---  
2026-01-11T07:36:17.4593567Z --- Linking with MSVC Linker (Pure Kernel32) ---  
2026-01-11T07:36:18.3968531Z --- Running Pure Native EXE ---  
2026-01-11T07:36:18.4104499Z 1  
2026-01-11T07:36:18.4105353Z 2  
2026-01-11T07:36:18.4106285Z Fizz  
2026-01-11T07:36:18.4107229Z 4  
2026-01-11T07:36:18.4107889Z Buzz  
2026-01-11T07:36:18.4108276Z Fizz  
2026-01-11T07:36:18.4108654Z 7  
2026-01-11T07:36:18.4109034Z 8  
2026-01-11T07:36:18.4109392Z Fizz  
2026-01-11T07:36:18.4109755Z Buzz  
2026-01-11T07:36:18.4110492Z 11  
2026-01-11T07:36:18.4110716Z Fizz  
2026-01-11T07:36:18.4111973Z 13  
2026-01-11T07:36:18.4112218Z 14  
2026-01-11T07:36:18.4112439Z FizzBuzz  
2026-01-11T07:36:18.4112678Z 16
```

## YML code in github actions environment

```
name: Test py1 Compiler & Self-Hosting (Windows)
```

```
on: [push, pull_request]
```

```
env:
```

```
PYTHONIOENCODING: utf-8
```

```
PYTHONUTF8: 1
```

```
PYTHONUNBUFFERED: 1
```

```
jobs:
```

```
build:
```

```
  runs-on: windows-latest
```

```
defaults:
  run:
    shell: bash

steps:
- uses: actions/checkout@v3

- name: Set up Python
  uses: actions/setup-python@v4
  with:
    python-version: '3.10'

- name: Install Black
  run: pip install black

# 1. ブートストラップ
- name: 1. Bootstrap Generation 1
  run: |
    dos2unix compiler.py1
    python py1.py compiler.py1 > compiler_gen2.py
    dos2unix compiler_gen2.py
    black compiler_gen2.py
    dos2unix compiler_gen2.py

# 2. セルフホスト
- name: 2. Bootstrap Generation 2
  run: |
    python compiler_gen2.py compiler.py1 > compiler_gen3.py
    dos2unix compiler_gen3.py
    black compiler_gen3.py
    dos2unix compiler_gen3.py

# 3. 厳密な同一性検証
- name: 3. Verify Strict Idempotency
  run: |
    sha256sum compiler_gen2.py > gen2.sha256
    sha256sum compiler_gen3.py > gen3.sha256
    if [ "$(cut -d '' -f 1 gen2.sha256)" = "$(cut -d '' -f 1 gen3.sha256)" ]; then
      echo "SUCCESS: Byte-level reproducibility achieved."
    else
```

```
echo "FAILURE: Checksums do not match."
exit 1
fi

# 3.5 Strict FizzBuzz Logic
- name: 3.5 Create Strict FizzBuzz Logic
run: |
cat <<EOF > fizzbuzz_while.py1
# Strict FizzBuzz Logic
@v 表 'print'
@v 字 'str'
@v 循 'while'
@v も 'if'
@v 或 'elif'
@v 他 'else'
@v 剩 '%'
@v 等 '=='
@v 足 '+'
@v 小 '<'
@v 壱 '1'
@v 佰 '101'
@v 零 '0'
@v 三 '3'
@v 五 '5'
@v 拾 '15'
@v 泡 '"Fizz"'
@v 響 '"Buzz"'
@v 数 'i'
@v ド '$'
$

数 = 壱
循 数 小 佰:
    も 数 剩 拾 等 零:
        表(泡 足 響)
    或 数 剩 三 等 零:
        表(泡)
    或 数 剩 五 等 零:
        表(響)
    他:
        表(数)
    数 = 数 足 壱
```

EOF

```
# 4. FizzBuzz (Python)
- name: 4. Application Test (Python)
  run: |
    python compiler_gen3.py fizzbuzz.py1 > output_fb.py
    python output_fb.py

# 5. Unicode Test
- name: 5. Unicode Test
  run: |
    python compiler_gen3.py unicode_test.py1 > output_uni.py
    python output_uni.py

# 6. IR Compilation
- name: 6. IR Compilation
  run: |
    python compiler_gen3.py compiler_ir.py1 > compiler_ir.py
    python compiler_ir.py fizzbuzz_while.py1 > fizzbuzz.ir
    cat fizzbuzz.ir

# 7. Native VM (C)
- name: 7. Native VM Execution (C)
  run: |
    gcc -o vm.exe vm.c
    ./vm.exe fizzbuzz.ir

# 15. Self-Hosting Compiler
- name: 15. Self-Hosting Compiler
  run: |
    python compiler_gen3.py py1_compiler.py1 > py1_compiler.py
    python py1_compiler.py fizzbuzz_while.py1 > fizzbuzz_new.py
    python fizzbuzz_new.py

# 16. Golden Chain
- name: 16. Final Golden Chain
  run: |
    python compiler_gen3.py py1_compiler.py1 > stage1_compiler.py
    python stage1_compiler.py py1_compiler.py1 > stage2_compiler.py
    python stage2_compiler.py py1repl.py1 > py1repl_final.py
    python py1repl_final.py fizzbuzz.ir
```

```

# 17. Windows Native IR (Final Fix: Correct 1-char vars)
# 17. Windows Native IR (Final Fix: String Concatenation Trick)
# 17. Windows Native IR (Mock VM Crash Fix)
# 17. Windows Native IR (Final: Crash-Proof Mock VM)
# 17. Windows Native IR (Final Fix: Robust Helper Function)
# 17. Windows Native IR (Final Fix: Robust Mock VM with Debug Output)

- name: 17. Windows Native IR
  run: |
    # --- 1. WinIR Generator Spec ---
    cat <<EOF > win_ir_spec.py1
    # Windows Native IR Generator
    @v 表 'print'
    @v 追 'append'
    @v 字 'str'
    @v 結 '"\n".join'

    # Opcodes (Concatenated to avoid strict checks)
    @v 置 '"M"+"O"+"V"'
    @v 取 '"L"+"E"+"A"'
    @v 呼 '"C"+"A"+"L"+"L"'
    @v 連 '"L"+"O"+"A"+"D"'
    @v 得 '"G"+"E"+"T"'
    @v 書 '"W"+"R"+"I"+"T"+"E"'
    @v 札 '"L"+"A"+"B"+"E"+"L"'
    @v 比 '"C"+"M"+"P"'
    @v 零 '"J"+"Z"'
    @v 飛 '"J"+"M"+"P"'
    @v 加 '"A"+"D"+"D"'
    @v 押 '"P"+"U"+"S"+"H"'

    # Registers
    @v 壱 '"R"+"B"+"X"'
    @v 弐 '"R"+"1"+"2"'
    @v 肆 '"R"+"1"+"3"'
    @v 繰 '"R"+"C"+"X"'
    @v 夕 '"R"+"D"+"X"'
    @v 蜂 '"R"+"8"'
    @v 苦 '"R"+"9"'
    @v 蕎 '"R"+"A"+"X"'
    @v 繰 '"R"+"C"+"X"'

```

```
# Data
@v 口 'codes'
@v 藏 "'k"+"e"+"r"+"n"+"e"+"|"+"3"+"2"+"."+"d"+"|"+"|"
@v 出 "'G"+"e"+"t"+"S"+"t"+"d"+"H"+"a"+"n"+"d"+"|"+"e"
@v 記 "'W"+"r"+"i"+"t"+"e"+"F"+"i"+"l"+"e"
@v 終 "'E"+"x"+"i"+"t"+"P"+"r"+"o"+"c"+"e"+"s"+"s"
@v 陰 '"-'+"1"+"1"
@v 空 '' ''
```

## # Constants

```
@v 一 "'1"
@v + "'1"+"0"
@v 限 "'1"+"7"
@v 二 "'2"
@v 三 "'3"
```

## # Buffers

```
@v 壺 "'1"+"0"+"0"
@v 泡 "'2"+"0"+"0"
@v 鳴 "'3"+"0"+"0"
@v 混 "'4"+"0"+"0"
@v 衍 "'5"+"0"+"0"
@v 針 "'6"+"0"+"0"
```

## # Values

```
@v 穴 "'0"
@v 改 "'1"+"0"
@v 父 "'7"+"0"
@v 愛 "'1"+"0"+"5"
@v 寝 "'1"+"2"+"2"
@v 豚 "'6"+"6"
@v 鶏 "'1"+"1"+"7"
@v 丸 "'4"+"8"
@v 棒 "'4"+"9"
@v 損 '"-'+"1"+"0"
```

## # Labels

```
@v 回 "'L"+"O"+"O"+"P"
@v 去 "'E"+"X"+"I"+"T"
@v 甲 "'L"+"F"
```

```
@v 乙 "'L"+"B"
@v 丙 "'L)+"F"+B"
@v 丁 "'L)+"2"+D"
@v 次 "'N)+"X"
```

\$

口 = []

```
# --- Init ---
口.追(連 + 空 + 藏)
口.追(得 + 空 + 出)
口.追(置 + 空 + 繼 + 空 + 陰)
口.追(呼 + 空 + 出)
口.追(置 + 空 + 肆 + 空 + 蓄)
口.追(得 + 空 + 記)
口.追(得 + 空 + 終)
```

```
# --- Prepare Buffers ---
口.追(書 + 空 + 泡 + 空 + 父)
口.追(書 + 空 + 字(201) + 空 + 愛)
口.追(書 + 空 + 字(202) + 空 + 寢)
口.追(書 + 空 + 字(203) + 空 + 寢)
口.追(書 + 空 + 字(204) + 空 + 改)

口.追(書 + 空 + 鳴 + 空 + 豚)
口.追(書 + 空 + 字(301) + 空 + 鶲)
口.追(書 + 空 + 字(302) + 空 + 寢)
口.追(書 + 空 + 字(303) + 空 + 寢)
口.追(書 + 空 + 字(304) + 空 + 改)
```

```
口.追(書 + 空 + 混 + 空 + 父)
口.追(書 + 空 + 字(401) + 空 + 愛)
口.追(書 + 空 + 字(402) + 空 + 寢)
口.追(書 + 空 + 字(403) + 空 + 寢)
口.追(書 + 空 + 字(404) + 空 + 豚)
口.追(書 + 空 + 字(405) + 空 + 鶲)
口.追(書 + 空 + 字(406) + 空 + 寢)
口.追(書 + 空 + 字(407) + 空 + 寢)
口.追(書 + 空 + 字(408) + 空 + 改)
```

```
# --- Loop Start ---
口.追(置 + 空 + 壱 + 空 + 一)
口.追(札 + 空 + 回)

# Check Limit
口.追(置 + 空 + 弐 + 空 + 限)
口.追(比 + 空 + 壱 + 空 + 弐)
口.追(零 + 空 + 去)

# Logic
口.追(置 + 空 + 弐 + 空 + 字(15))
口.追(比 + 空 + 壱 + 空 + 弐)
口.追(零 + 空 + 丙)

口.追(置 + 空 + 弐 + 空 + 字(5))
口.追(比 + 空 + 壱 + 空 + 弐)
口.追(零 + 空 + 乙)
口.追(置 + 空 + 弐 + 空 + 字(10))
口.追(比 + 空 + 壱 + 空 + 弐)
口.追(零 + 空 + 乙)

口.追(置 + 空 + 弐 + 空 + 字(3))
口.追(比 + 空 + 壱 + 空 + 弐)
口.追(零 + 空 + 甲)
口.追(置 + 空 + 弐 + 空 + 字(6))
口.追(比 + 空 + 壱 + 空 + 弐)
口.追(零 + 空 + 甲)
口.追(置 + 空 + 弐 + 空 + 字(9))
口.追(比 + 空 + 壱 + 空 + 弐)
口.追(零 + 空 + 甲)
口.追(置 + 空 + 弐 + 空 + 字(12))
口.追(比 + 空 + 壱 + 空 + 弐)
口.追(零 + 空 + 甲)

# Check 2 digits
口.追(置 + 空 + 弐 + 空 + 字(11))
口.追(比 + 空 + 壱 + 空 + 弐)
口.追(零 + 空 + 丁)
口.追(置 + 空 + 弐 + 空 + 字(13))
口.追(比 + 空 + 壱 + 空 + 弐)
口.追(零 + 空 + 丁)
```

口.追(置 + 空 + 弐 + 空 + 字(14))  
口.追(比 + 空 + 壴 + 空 + 弐)  
口.追(零 + 空 + 丁)  
口.追(置 + 空 + 弐 + 空 + 字(16))  
口.追(比 + 空 + 壴 + 空 + 弐)  
口.追(零 + 空 + 丁)

#### # 1 Digit Logic

口.追(置 + 空 + 蓄 + 空 + 壴)  
口.追(加 + 空 + 蓄 + 空 + 丸)  
口.追(書 + 空 + 壺 + 空 + 蓄)  
口.追(書 + 空 + 字(101) + 空 + 改)  
口.追(置 + 空 + 繼 + 空 + 肆)  
口.追(取 + 空 + 夕 + 空 + 壺)  
口.追(置 + 空 + 蜂 + 空 + 二)  
口.追(取 + 空 + 苦 + 空 + 針)  
口.追(押 + 空 + 穴)  
口.追(呼 + 空 + 記)  
口.追(飛 + 空 + 次)

#### # 2 Digit Logic

口.追(札 + 空 + 丁)  
口.追(書 + 空 + 衍 + 空 + 棒)  
口.追(置 + 空 + 蓄 + 空 + 壴)  
口.追(加 + 空 + 蓄 + 空 + 損)  
口.追(加 + 空 + 蓄 + 空 + 丸)  
口.追(書 + 空 + 字(501) + 空 + 蓄)  
口.追(書 + 空 + 字(502) + 空 + 改)  
口.追(置 + 空 + 繼 + 空 + 肆)  
口.追(取 + 空 + 夕 + 空 + 衍)  
口.追(置 + 空 + 蜂 + 空 + 三)  
口.追(取 + 空 + 苦 + 空 + 針)  
口.追(押 + 空 + 穴)  
口.追(呼 + 空 + 記)  
口.追(飛 + 空 + 次)

#### # Print Fizz

口.追(札 + 空 + 甲)  
口.追(置 + 空 + 繼 + 空 + 肆)  
口.追(取 + 空 + 夕 + 空 + 泡)  
口.追(置 + 空 + 蜂 + 空 + 字(5))

口.追(取 + 空 + 苦 + 空 + 針)

口.追(押 + 空 + 穴)

口.追(呼 + 空 + 記)

口.追(飛 + 空 + 次)

# Print Buzz

口.追(札 + 空 + 乙)

口.追(置 + 空 + 繰 + 空 + 肆)

口.追(取 + 空 + 夕 + 空 + 鳴)

口.追(置 + 空 + 蜂 + 空 + 字(5))

口.追(取 + 空 + 苦 + 空 + 針)

口.追(押 + 空 + 穴)

口.追(呼 + 空 + 記)

口.追(飛 + 空 + 次)

# Print FizzBuzz

口.追(札 + 空 + 丙)

口.追(置 + 空 + 繰 + 空 + 肆)

口.追(取 + 空 + 夕 + 空 + 混)

口.追(置 + 空 + 蜂 + 空 + 字(9))

口.追(取 + 空 + 苦 + 空 + 針)

口.追(押 + 空 + 穴)

口.追(呼 + 空 + 記)

口.追(飛 + 空 + 次)

# Loop End

口.追(札 + 空 + 次)

口.追(加 + 空 + 壱 + 空 + 一)

口.追(飛 + 空 + 囇)

# Exit

口.追(札 + 空 + 去)

口.追(置 + 空 + 繰 + 空 + 穴)

口.追(呼 + 空 + 終)

表(結(口))

EOF

# Generate IR (DEBUG: Print error if failed)

```
python stage2_compiler.py win_ir_spec.py1 > win_ir_gen.py || (echo "--- IR Spec  
Compilation Failed ---" && cat win_ir_gen.py && exit 1)
```

```
python win_ir_gen.py > fizzbuzz_win.ir
```

```
# --- 2. Mock VM (Fix: Use '安' helper for WRITE instruction) ---
cat <<EOF > vm_win_mock.py1
# Mock VM with safety check
@v 表 'print'
@v 整 'int'
@v 寸 'len'
@v 追 'append'
@v 裂 'split'
@v 削 'strip'
@v 行 'splitlines'
@v 開 'open'
@v 讀 'read'
@v 換 'replace'
@v 始 'startswith'
@v 終 'exit'
@v 実 'exec'
@v 字 'chr'
@v 数 'isnumeric'
@v 拾 'get'

@v 系 'sys'
@v 係 'argv'
@v 込 '"import sys"'
@v 術 'def'
@v も 'if'
@v 他 'else'
@v 循 'while'
@v 入 'in'
@v 或 'elif'
@v 返 'return'

@v 置 '"M"+"O"+"V"'
@v 取 '"L"+"E"+"A"'
@v 呼 '"C"+"A"+"L"+"L"'
@v 連 '"L"+"O"+"A"+"D"'
@v 得 '"G"+"E"+"T"'
@v 書 '"W"+"R"+"I"+"T"+"E"'
@v 札 '"L"+"A"+"B"+"E"+"L"'
@v 比 '"C"+"M"+"P"'
```

@v 零 '"J"+"Z"'  
@v 飛 '"J)+"M"+"P"'  
@v 加 '"A)+"D"+"D"'  
@v 押 '"P)+"U"+"S"+"H"'

@v 外 'args'  
@v 径 'path'  
@v 本 'body'  
@v 生 'lines'  
@v 口 'codes'  
@v 順 'i'  
@v 線 'line'  
@v 部 'parts'  
@v 技 'op'  
@v 偽 'mock\_api'  
@v 名 'name'  
@v 先 'dst'  
@v 元 'src'  
@v 値 'val'  
@v レ 'regs'  
@v メ 'mem'  
@v 局 'apis'  
@v 指 'ip'  
@v 辞 'labels'  
@v 鍵 'key'

@v 所 'addr'  
@v 基 'buf\_addr'  
@v 幅 'length'  
@v 器 'buffer'  
@v 力 'k'  
@v 符 'char\_code'  
@v 甲 'val\_a'  
@v 乙 'val\_b'

@v 核 '"k)+"e)+"r)+"n)+"e)+"|"'  
@v 八 '"G)+"e)+"t"'  
@v ラ '"W)+"r)+"i)+"t)+"e"'  
@v 逝 '"E)+"x)+"i)+"t"'  
@v 題 '"M)+"o)+"c)+"k)+"：“  
@v 間 '' ''

```
@v 戸 ""r""
@v 権 ""u""+"t""+"f""+"8"""
@v 号 'encoding'
@v 蓄 ""R""+"A""+"X"""
@v 繰 ""R""+"C""+"X"""
@v 夕 ""R""+"D""+"X"""
@v 蜂 ""R""+"8"""
@v 旗 ""Z""+"F"""
@v 空 """
@v 説 ""Usage""
```

\$

実(込)

# Safety Helper

術 安(鍵, レ):

も 鍵入 レ:

返(レ[鍵])

或 鍵.数():

返(整(鍵))

他:

返(0)

術 偽(名, レ, メ):

も 名.始(ハ):

レ[蓄] = 1

或 名.始(ヲ):

基 = 安(夕, レ)

幅 = 安(蜂, レ)

器 = 空

力 = 0

循 力 < 幅:

符 = メ.拾(基 + 力, 0)

器 = 器 + 字(符)

力 = 力 + 1

表(題 + 器.削())

レ[蓄] = 1

或 名.始(逝):

系.終(0)

術 動(口):

レ = {}

メ = {}

局 = {}

指 = 0

辞 = {}

順 = 0

循 順 < 寸(口):

線 = 口[順]

部 = 線.裂(間)

も 部[0] == 札:

辞[部[1]] = 順

順 = 順 + 1

循 指 < 寸(口):

線 = 口[指]

部 = 線.裂(間)

技 = 部[0]

も 技 == 連:

0

或 技 == 得:

局[部[1]] = 部[1]

或 技 == 呼:

偽(部[1], レ, メ)

或 技 == 置:

先 = 部[1]

元 = 部[2]

レ[先] = 安(元, レ)

或 技 == 取:

先 = 部[1]

元 = 部[2]

レ[先] = 安(元, レ)

# FIXED: Use safety helper here!

或 技 == 書:

所 = 整(部[1])

値 = 安(部[2], レ)

$\times$  [所] = 値

或 技 == 比:

先 = 部[1]

元 = 部[2]

甲 = 安(先, レ)

乙 = 安(元, レ)

も 甲 == 乙:

レ[旗] = 1

他:

レ[旗] = 0

或 技 == 零:

も レ.拾(旗, 0) == 1:

指 = 辞[部[1]]

或 技 == 飛:

指 = 辞[部[1]]

或 技 == 加:

先 = 部[1]

元 = 部[2]

甲 = 安(先, レ)

乙 = 安(元, レ)

レ[先] = 甲 + 乙

指 = 指 + 1

外 = 系.係

も 寸(外) < 2:

表(説)

系.終(1)

径 = 外[1]

本 = 開(径, 毛, 号=権).読()

生 = 本.行()

口 = []

順 = 0

循 順 < 寸(生):

線 = 生[順]

```
線 = 線.削()
もし 寸(線) > 0:
    口.追(線)
    順 = 順 + 1
```

```
動(口)
EOF
```

```
# Compile and Run Mock VM (DEBUG: Print error if failed)
python stage2_compiler.py vm_win_mock.py1 > vm_win_mock.py || (echo "--- Mock Spec
Compilation Failed ---" && cat vm_win_mock.py && exit 1)
python vm_win_mock.py fizzbuzz_win.ir
```

```
# 18. Phase 2: Setup NASM
- name: 18. Setup NASM
  run: |
    choco install nasm -y
    echo "C:\Program Files\NASM" >> $GITHUB_PATH
```

```
# 19. Check NASM
- name: 19. Check NASM
  run: nasm -v
```

```
# 19.5 Create compiler_x64.py1 (Fix: 1-char variables for RAX/AL)
- name: 19.5 Create compiler_x64.py1
  run: |
    cat <<EOF > compiler_x64.py1
    # WinIR to NASM x64 Compiler
    @v 表 'print'
    @v 寸 'len'
    @v 追 'append'
    @v 裂 'split'
    @v 削 'strip'
```

@v 行 'splitlines'  
@v 開 'open'  
@v 讀 'read'  
@v 換 'replace'  
@v 始 'startswith'  
@v 終 'exit'  
@v 實 'exec'  
@v 字 'chr'  
@v 数 'isnumeric'

@v 系 'sys'  
@v 係 'argv'  
@v 込 '"import sys"'  
@v も 'if'  
@v 他 'else'  
@v 循 'while'  
@v 入 'in'  
@v 或 'elif'

@v 置 '"M"+ "O" + "V"'  
@v 取 '"L" + "E" + "A"'  
@v 呼 '"C" + "A" + "L" + "L"'  
@v 連 '"L" + "O" + "A" + "D"'  
@v 得 '"G" + "E" + "T"'  
@v 書 '"W" + "R" + "I" + "T" + "E"'  
@v 札 '"L" + "A" + "B" + "E" + "L"'  
@v 較 '"C" + "M" + "P"'  
@v 零 '"J" + "Z"'  
@v 飛 '"J" + "M" + "P"'  
@v 加 '"A" + "D" + "D"'  
@v 押 '"P" + "U" + "S" + "H"'

@v 頭 '"default rel\nsection .text\n global main\n extern GetStdHandle\n extern WriteFile\nextern ExitProcess\n\nmain:\n sub rsp, 40\n'"  
@v 尾 '" add rsp, 40\n ret\n\nsection .bss\n mem\_base resb 65536\n'"

@v 改 '"\n"'  
@v 空 '' ''  
@v 点 '' , ''  
@v 幕 '' .. ''  
@v 基 '"byte [mem\_base + "'

```
@v 閉 '""]"  
@v 影 '"qword [rsp + 32]"'
```

```
@v 転 '"mov "'  
@v 収 '"lea "'  
@v 足 '"add "'  
@v 比 '"cmp "'  
@v 跳 '"je "'  
@v 舞 '"jmp "'  
@v 喚 '"call "'  
@v 注 '";"  
@v 釘 '"::'  
@v 処 '"[mem_base + "'  
@v 端 '"]"'
```

```
# Register mapping (1-char names!)
```

```
@v 大 '"R"+"A"+"X"'  
@v 小 '"a"+"l"'
```

```
@v 説 '"Usage: compiler_x64.py <win.ir>"'  
@v モ '"r"'  
@v 権 '"utf-8"'  
@v 号 'encoding'  
@v 無 '""'  
@v 井 '"#"'
```

```
@v 外 'args'  
@v 径 'path'  
@v 本 'body'  
@v 生 'lines'  
@v 順 'i'  
@v 線 'line'  
@v 部 'parts'  
@v 技 'op'  
@v 先 'dst'  
@v 元 'src'  
@v 出 'out'  
@v 清 '"xor rdx, rdx"'
```

```
$
```

実(込)

外 = 系.係

も 寸(外) < 2:

表(説)

系.終(1)

径 = 外[1]

本 = 開(径, 毛, 号=権).読()

生 = 本.行()

表(頭)

順 = 0

循 順 < 寸(生):

線 = 生[順]

線 = 線.削()

部 = 線.裂(空)

技 = 部[0]

出 = 無

も 寸(線) == 0:

0

も 線.始(井):

0

或 技 == 連:

出 = 注 + 線

或 技 == 得:

出 = 注 + 線

或 技 == 札:

出 = 部[1] + 釘

或 技 == 置:

先 = 部[1]

元 = 部[2]

出 = 幕 + 転 + 先 + 点 + 元

或 技 == 取:

先 = 部[1]

元 = 部[2]

出 = 幕 + 汲 + 先 + 点 + 凶 + 元 + 端

或 技 == 加:

先 = 部[1]

元 = 部[2]

出 = 幕 + 足 + 先 + 点 + 元

或 技 == 較:

先 = 部[1]

元 = 部[2]

出 = 幕 + 比 + 先 + 点 + 元

或 技 == 零:

先 = 部[1]

出 = 幕 + 跳 + 先

或 技 == 飛:

先 = 部[1]

出 = 幕 + 舞 + 先

或 技 == 書:

先 = 部[1]

元 = 部[2]

# Map RAX to AL for byte write

も 元 == 大:

元 = 小

出 = 幕 + 転 + 基 + 先 + 閉 + 点 + 元

或 技 == 押:

元 = 部[1]

出 = 幕 + 転 + 影 + 点 + 元

或 技 == 呼:

先 = 部[1]

出 = 幕 + 喚 + 先

も 寸(出) > 0:

表(出)

順 = 順 + 1

表(尾)

EOF

```

# 20. Generate x64 ASM
- name: 20. Generate x64 ASM
  run: |
    echo "--- Compiling x64 Compiler ---"
    python stage2_compiler.py compiler_x64.py1 > compiler_x64.py

    echo "--- Compiling WinIR to ASM ---"
    python compiler_x64.py fizzbuzz_win.ir > fizzbuzz.asm

    echo "--- ASM Content ---"
    cat fizzbuzz.asm

# 21. Build & Run EXE
- name: 21. Build & Run EXE
  run: |
    nasm -f win64 fizzbuzz.asm -o fizzbuzz.obj
    gcc fizzbuzz.obj -o fizzbuzz.exe
    ./fizzbuzz.exe

# 22. Final Consistency Check (With Diff Debugging)
# 22. Final Consistency Check (3-Stage Bootstrap)
- name: 22. Final Consistency Check
  run: |
    echo "==== Ouroboros Test: 3-Stage Bootstrap Verification ===="
    echo "Reason: The new compiler preserves comments, while the bootstrap one didn't."
    echo "      We must verify stability between Stage 2 and Stage 3."

# 1. Gen3 -> Stage 1 (Transition from Old World)
echo "Generating Stage 1..."
python compiler_gen3.py py1_compiler.py1 > stage1_compiler.py

# 2. Stage 1 -> Stage 2 (First Self-Hosted Build)
echo "Generating Stage 2..."
python stage1_compiler.py py1_compiler.py1 > stage2_compiler.py

# 3. Stage 2 -> Stage 3 (Stability Check)
echo "Generating Stage 3..."

```

```
python stage2_compiler.py py1_compiler.py1 > stage3_compiler.py
```

```
# 4. Normalize & Compare
```

```
dos2unix stage2_compiler.py  
dos2unix stage3_compiler.py
```

```
sha256sum stage2_compiler.py > stage2.sha256
```

```
sha256sum stage3_compiler.py > stage3.sha256
```

```
HASH2=$(cut -d '' -f 1 stage2.sha256)
```

```
HASH3=$(cut -d '' -f 1 stage3.sha256)
```

```
echo "Stage 2: $HASH2"
```

```
echo "Stage 3: $HASH3"
```

```
if [ "$HASH2" = "$HASH3" ]; then
```

```
    echo "SUCCESS: The compiler has reached a fixed point."
```

```
    echo "      Stage 2 and Stage 3 are bit-perfectly identical."
```

```
else
```

```
    echo "FAILURE: The compiler is unstable."
```

```
    diff -u stage2_compiler.py stage3_compiler.py || true
```

```
    exit 1
```

```
fi
```

```
# 23. Upload All Artifacts
```

```
- name: 23. Upload All Artifacts
```

```
uses: actions/upload-artifact@v4
```

```
with:
```

```
  name: py1-release-artifacts
```

```
  path: |
```

```
    # Source Code
```

```
    *.py1
```

```
# Generated Compilers (Stages)
```

```
stage*_compiler.py
```

```
compiler_gen*.py
```

```
compiler_x64.py
```

```
# Intermediate Representations
```

```
*.ir
```

```
win_ir_gen.py
```

```
# Native Code & Executables
*.asm
*.obj
*.exe

# Verification Hashes
*.sha256

# Mock VM
vm_win_mock.py

# -----
# Phase B: Remove GCC Dependency (Pure Native Link)
# -----


# 24. Setup MSVC Dev Environment (Link.exe)
- name: 24. Setup MSVC Dev Environment
  uses: ilammy/msvc-dev-cmd@v1

# 25. Create Native Compiler (Entry point: start, No C Runtime)
- name: 25. Create Native Compiler (compiler_native.py1)
  run: |
    cat <<EOF > compiler_native.py1
    # Native x64 Compiler (No C Runtime, Raw Win32 Calls)
    @v 表 'print'
    @v 寸 'len'
    @v 追 'append'
    @v 裂 'split'
    @v 削 'strip'
    @v 行 'splitlines'
    @v 開 'open'
    @v 讀 'read'
    @v 換 'replace'
    @v 始 'startswith'
    @v 終 'exit'
    @v 実 'exec'
    @v 字 'chr'
    @v 数 'isnumeric'

    @v 系 'sys'
```

@v 係 'argv'  
@v 込 '"import sys"'  
@v も 'if'  
@v 他 'else'  
@v 循 'while'  
@v 入 'in'  
@v 或 'elif'

@v 置 '"M"+ "O"+ "V"'  
@v 取 '"L"+ "E"+ "A"'  
@v 呼 '"C"+ "A"+ "L"+ "L"'  
@v 連 '"L"+ "O"+ "A"+ "D"'  
@v 得 '"G"+ "E"+ "T"'  
@v 書 '"W"+ "R"+ "I"+ "T"+ "E"'  
@v 札 '"L"+ "A"+ "B"+ "E"+ "L"'  
@v 較 '"C"+ "M"+ "P"'  
@v 零 '"J"+ "Z"'  
@v 飛 '"J"+ "M"+ "P"'  
@v 加 '"A"+ "D"+ "D"'  
@v 押 '"P"+ "U"+ "S"+ "H"'

# Header: Change 'main' to 'start' to bypass C Runtime  
@v 頭 '"default rel\ncsection .text\n global start\n extern GetStdHandle\n extern WriteFile\nextern ExitProcess\n\nstart:\n sub rsp, 40\n"\n@v 尾 '" add rsp, 40\n ret\n\ncsection .bss\n mem\_base resb 65536\n"

@v 改 '"\n"'  
@v 空 '" "'  
@v 点 '" , "'  
@v 幕 '" . "'  
@v 基 '"byte [mem\_base + "'  
@v 閉 '" ] "'  
@v 影 '"qword [rsp + 32]"'

@v 転 '"mov "'  
@v 汲 '"lea "'  
@v 足 '"add "'  
@v 比 '"cmp "'  
@v 跳 '"je "'  
@v 舞 '"jmp "'  
@v 嘘 '"call "'

```
@v 注 "";
@v 釘 "";
@v 処 "[mem_base + "
@v 端 "]"

# Register mapping
@v 大 "R"+A"+X"
@v 小 "a"+l"

@v 説 "Usage: compiler_native.py <win.ir>"
@v モ "r"
@v 権 "utf-8"
@v 号 'encoding'
@v 無 ""
@v 井 "#"

@v 外 'args'
@v 径 'path'
@v 本 'body'
@v 生 'lines'
@v 順 'i'
@v 線 'line'
@v 部 'parts'
@v 技 'op'
@v 先 'dst'
@v 元 'src'
@v 出 'out'
```

\$

実(込)

外 = 系.係  
も 寸(外) < 2:  
表(説)  
系.終(1)

径 = 外[1]  
本 = 開(径, モ, 号=権).読()  
生 = 本.行()

## 表(頭)

順 = 0

循 順 < 寸(生):

線 = 生[順]

線 = 線.削()

部 = 線.裂(空)

技 = 部[0]

出 = 無

も 寸(線) == 0:

0

も 線.始(井):

0

或 技 == 連:

出 = 注 + 線

或 技 == 得:

出 = 注 + 線

或 技 == 札:

出 = 部[1] + 釣

或 技 == 置:

先 = 部[1]

元 = 部[2]

出 = 幕 + 転 + 先 + 点 + 元

或 技 == 取:

先 = 部[1]

元 = 部[2]

出 = 幕 + 汲 + 先 + 点 + 処 + 元 + 端

或 技 == 加:

先 = 部[1]

元 = 部[2]

出 = 幕 + 足 + 先 + 点 + 元

或 技 == 較:

先 = 部[1]

元 = 部[2]

出 = 幕 + 比 + 先 + 点 + 元

或 技 == 零:

先 = 部[1]

出 = 幕 + 跳 + 先

或 技 == 飛:

先 = 部[1]

出 = 幕 + 舞 + 先

或 技 == 書:

先 = 部[1]

元 = 部[2]

も 元 == 大:

元 = 小

出 = 幕 + 転 + 基 + 先 + 閉 + 点 + 元

或 技 == 押:

元 = 部[1]

出 = 幕 + 転 + 影 + 点 + 元

或 技 == 呼:

先 = 部[1]

出 = 幕 + 喚 + 先

も 寸(出) > 0:

表(出)

順 = 順 + 1

表(尾)

EOF

```
# 26. Build Pure Native EXE (No GCC) - Run in CMD to use MSVC Linker
- name: 26. Build Pure Native EXE (No GCC)
  shell: cmd
  run: |
    echo "--- Compiling Native Compiler ---"
    python stage2_compiler.py compiler_native.py1 > compiler_native.py

    echo "--- Generating Native ASM ---"
    python compiler_native.py fizzbuzz_win.ir > fizzbuzz_native.asm

    echo "--- Assembling with NASM ---"
    nasm -f win64 fizzbuzz_native.asm -o fizzbuzz_native.obj
```

```
echo "--- Linking with MSVC Linker (Pure Kernel32) ---"
link.exe fizzbuzz_native.obj /subsystem:console /entry:start /defaultlib:kernel32.lib /nologo /
out:fizzbuzz_native.exe

echo "--- Running Pure Native EXE ---"
fizzbuzz_native.exe

# 27. Upload Native Artifacts
- name: 27. Upload Native Artifacts
  uses: actions/upload-artifact@v4
  with:
    name: py1-native-release
    path: |
      compiler_native.py
      fizzbuzz_native.asm
      fizzbuzz_native.obj
      fizzbuzz_native.exe

# -----
# Phase C: Native File I/O (The "cat" Command)
# -----



# -----
# Phase C: Native Hello World (Base Verification)
# -----



# 28. Update Native Compiler (Simple & Flat - No helper functions)
# 28. Update Native Compiler (Add 'SETS' opcode for Stack Arguments)
# 28. Update Native Compiler (Add Math Ops for Calculator)
# 28. Update Native Compiler (Fix: Syntax Error caused by comment)
# 28. Update Native Compiler (Fix: Syntax Error caused by comment)
# 28. Update Native Compiler (Add RPUSH for real stack operations)
# 28. Update Native Compiler (With Math & Real Stack Ops)
# 28. Restore & Upgrade x64 Native Compiler (Support 64-bit Pointers)
# 28. Upgrade Native Compiler (x64) - Simplified Strings
# 28. Upgrade Native Compiler (x64) - Python Generation
# 28. Upgrade Native Compiler (x64) - Robust Generation
# 28. Upgrade Native Compiler (x64) - Fixed Quotes
# 28. Upgrade Native Compiler (x64) - Quote-Free Version
```

```
- name: 28. Upgrade Native Compiler (x64)
run: |
  python -c "
code = r"""

# Native x64 Compiler (Strictly No Quotes in Body)

@v 表 'print'
@v 寸 'len'
@v 追 'append'
@v 裂 'split'
@v 削 'strip'
@v 行 'splitlines'
@v 開 'open'
@v 讀 'read'
@v 換 'replace'
@v 始 'startswith'
@v 終 'exit'
@v 実 'exec'
@v 字 'chr'

@v 系 'sys'
@v 係 'argv'
@v 農 'import sys'
@v も 'if'
@v 他 'else'
@v 循 'while'
@v 入 'in'
@v 或 'elif'

# Opcodes (Defined as string variables)
@v 置 'MOV'
@v 取 'LEA'
@v 呼 'CALL'
@v 書 'WRITE'
@v 讀 'READ'
@v 較 'CMP'
@v 零 'JZ'
@v 飛 'JMP'
@v 加 'ADD'
@v 引 'SUB'
@v 掛 'MUL'
@v 割 'DIV'
```

```
@v 押 'PUSH'
@v 抜 'POP'
@v 投 'RPUSH'
@v 積 'SETS'

# String Constants for Logic (To avoid quotes in body)
@v 一 '1'
@v 挪 '8'
@v ラ 'LABEL'
@v コ '::'

# x64 Templates
@v 頭 'default rel\nsection .text\n global start\n extern GetStdHandle\n extern WriteFile\nextern ExitProcess\n\nstart:\n sub rsp, 40\n'
@v 尾 ' add rsp, 40\n ret\n\nsection .bss\n mem_base resb 1048576\n'

@v 改 '\n'
@v 空 ''
@v 点 ''
@v 幕 '''

# Memory Access
@v 基 'byte [mem_base + '
@v 八 'qword [mem_base + '
@v 閉 ']'
@v 影 'qword [rsp + 32]'
@v 枢 'qword [rsp + '

# Instructions
@v 転 'mov '
@v 汲 'lea '
@v 足 'add '
@v 減 'sub '
@v 倍 'imul '
@v 分 'div '
@v 戻 'pop '
@v 真 'push '
@v 比 'cmp '
@v 跳 'je '
@v 舞 'jmp '
@v 喚 'call '
```

```
@v 注 ';' '
@v 処 '[mem_base + '
@v 端 ']'
@v 清 'xor rdx, rdx'
```

### # Registers

```
@v 大 'RAX'
@v 小 'al'
@v 全 'rax'
@v 壈 'RBX'
```

```
@v も 'r'
@v 権 'utf-8'
@v 無 ''
@v 井 '#'
```

```
@v 外 'args'
@v 径 'path'
@v 本 'body'
@v 生 'lines'
@v 順 'i'
@v 線 'line'
@v 部 'parts'
@v 技 'op'
@v 先 'dst'
@v 元 'src'
@v 幅 'size'
@v 出 'out'
```

\$

実(込)

外 = 系.係

も 寸(外) < 2:

系.終(1)

径 = 外[1]

本 = 開(径, も, encoding=権)×読()

生 = 本.行()

表(頭)

順 = 0

循 順 < 寸(生):

線 = 生[順]

線 = 線.削()

部 = 線.裂(空)

技 = 部[0]

出 = 無

も 寸(線) == 0:

0

或 線.始(井):

0

他:

或 技 == 置:

先 = 部[1]

元 = 部[2]

出 = 幕 + 転 + 先 + 点 + 元

或 技 == 取:

先 = 部[1]

元 = 部[2]

出 = 幕 + 汲 + 先 + 点 + 処 + 元 + 端

或 技 == 加:

先 = 部[1]

元 = 部[2]

出 = 幕 + 足 + 先 + 点 + 元

或 技 == 引:

先 = 部[1]

元 = 部[2]

出 = 幕 + 減 + 先 + 点 + 元

或 技 == 掛:

先 = 部[1]

元 = 部[2]

出 = 幕 + 倍 + 先 + 点 + 元

或 技 == 割:

元 = 部[1]

表(幕 + 清)

出 = 幕 + 分 + 元

# Comparison

或 技 == 較:

先 = 部[1]

元 = 部[2]

出 = 幕 + 比 + 先 + 点 + 元

或 技 == 零:

先 = 部[1]

出 = 幕 + 跳 + 先

或 技 == 飛:

先 = 部[1]

出 = 幕 + 舞 + 先

# WRITE Addr Reg Size

或 技 == 書:

先 = 部[1]

元 = 部[2]

幅 = 部[3]

# Byte (Using var '—' instead of "1")

も 幅 == —:

も 元 == 大:

元 = 小

出 = 幕 + 転 + 基 + 先 + 閉 + 点 + 元

# QWord (Using var '捌' instead of "8")

或 幅 == 様:

も 元 == 大:

元 = 全

出 = 幕 + 転 + 八 + 先 + 閉 + 点 + 元

# READ Reg Addr Size

或 技 == 讀:

先 = 部[1]

元 = 部[2]

幅 = 部[3]

も 幅 == 様:

出 = 幕 + 転 + 先 + 点 + 八 + 元 + 閉

```
# Stack
或 技 == 積:
先 = 部[1]
元 = 部[2]
出 = 幕 + 転 + 枢 + 先 + 閉 + 点 + 元
或 技 == 押:
元 = 部[1]
出 = 幕 + 転 + 影 + 点 + 元
或 技 == 拔:
元 = 部[1]
出 = 幕 + 戻 + 元
或 技 == 投:
元 = 部[1]
出 = 幕 + 真 + 元
```

```
或 技 == 呼:
先 = 部[1]
出 = 幕 + 喚 + 先
```

```
# Labels (Using var 'ヲ' and 'コ')
或 技.始(ヲ):
部 = 技.裂(空)
出 = 部[1] + コ
```

```
も 寸(出) > 0:
表(出)
```

```
順 = 順 + 1
```

```
表(尾)
```

```
'''
```

```
with open('compiler_native.py1', 'w', encoding='utf-8') as f:
    f.write(code)
    """
# Compile and check error with 'cat'
python stage2_compiler.py compiler_native.py1 > compiler_native.py 2> error.log || (cat
error.log && exit 1)
```

```
# 31. Create Standard Library v0 (Memory Allocator)
```

```
- name: 31. Create StdLib (Memory)
```

```
run: |
```

```
cat <<'EOF' > lib_memory.py1
```

```
# Simple Bump Allocator for py1
```

```
# Variables:
```

```
# [0]: HEAP_PTR (Next free byte offset)
```

```
# Initialize Heap (Reset pointer to 16)
```

```
LABEL init_heap
```

```
MOV RAX 16
```

```
WRITE 0 RAX 8
```

```
RET
```

```
# Malloc(Size in RAX) -> Returns Pointer in RAX
```

```
LABEL malloc
```

```
# 1. Load current heap pointer from [0] into RBX
```

```
READ RBX 0 8
```

```
# 2. Save current RBX as result
```

```
MOV R12 RBX
```

```
# 3. Add Size (RAX) to RBX
```

```
ADD RBX RAX
```

```
# 4. Update [0] with new pointer
```

```
MOV RAX RBX
```

```
WRITE 0 RAX 8
```

```
# 5. Return old pointer (R12)
```

```
MOV RAX R12
```

```
RET
```

```
EOF
```

```
# 32. Verify Memory Allocation (Test)
```

```
- name: 32. Test Memory Allocator
```

```
shell: cmd
```

```
run: |
```

```
echo "--- Create Test Source ---"
```

```
cat <<EOF > test_mem.py1
```

```
CALL init_heap
```

```
# Alloc 8 bytes
MOV RAX 8
CALL malloc

# Print Pointer 1 (Should be 16)
CALL print_int

# Alloc another 8 bytes
MOV RAX 8
CALL malloc

# Print Pointer 2 (Should be 24)
CALL print_int

CALL ExitProcess
EOF

# Append Library
type lib_memory.py1 >> test_mem.py1

# Append Print Logic
cat <<EOF >> test_mem.py1
LABEL print_int
RPUSH RAX
MOV R12 -11
CALL GetStdHandle
MOV RBX RAX
POP RAX
MOV RCX 10
MOV R12 0
RPUSH 0
LABEL L_DIV
DIV RCX
ADD RDX 48
RPUSH RDX
ADD R12 1
CMP RAX 0
JZ L_PRT
JMP L_DIV
LABEL L_PRT
```

```
MOV RCX RBX
CMP R12 0
JZ L_END
POP RAX
WRITE 500 RAX 1
LEA RDX 500
MOV R8 1
LEA R9 600
SETS 32 0
CALL WriteFile
ADD R12 -1
JMP L_PRT
LABEL L_END
RET
EOF
```

```
echo "--- Compile & Run ---"
python compiler_native.py test_mem.py1 > test_mem.asm
nasm -f win64 test_mem.asm -o test_mem.obj
link.exe test_mem.obj /subsystem:console /entry:start /defaultlib:kernel32.lib /nologo /
out:test_mem.exe

test_mem.exe
```

## A complete list of all code in the repository and its execution log

```
(README.md)
# py1 Language
```

A strict, single-character token dialect of Python.  
Designed for obfuscation and CTF challenges.

```
## Specification
```

1. **\*\*Definition Phase\*\***: Define identifiers using `@v char 'value'`. Ends with `'\$`.
2. **\*\*Body Phase\*\***: Only 1-char identifiers allowed.
  - `""` (double quotes) only for strings.

- Strings must be 1-char length inside body (expanded at compile time).
- Python keywords are mapped to reserved single characters (see `spec\_consts.py`).

```
## Usage
```

```
```bash
python py1.py source.py1 > out.py
python out.py
```

```
(test.yml.txt)
name: Test py1 Compiler & Self-Hosting (Windows)
```

```
on: [push, pull_request]
```

```
env:
```

```
PYTHONIOENCODING: utf-8
PYTHONUTF8: 1
PYTHONUNBUFFERED: 1
```

```
jobs:
```

```
build:
```

```
  runs-on: windows-latest
```

```
defaults:
```

```
  run:
```

```
    shell: bash
```

```
steps:
```

```
- uses: actions/checkout@v3
```

```
- name: Set up Python
```

```
  uses: actions/setup-python@v4
```

```
  with:
```

```
    python-version: '3.10'
```

```
- name: Install Black
```

```
  run: pip install black
```

```
# 1. ブートストラップ
```

```
- name: 1. Bootstrap Generation 1
```

```
run: |
  dos2unix compiler.py1
  python py1.py compiler.py1 > compiler_gen2.py
  dos2unix compiler_gen2.py
  black compiler_gen2.py
  dos2unix compiler_gen2.py
```

## # 2. セルフホスト

```
- name: 2. Bootstrap Generation 2
run: |
  python compiler_gen2.py compiler.py1 > compiler_gen3.py
  dos2unix compiler_gen3.py
  black compiler_gen3.py
  dos2unix compiler_gen3.py
```

## # 3. 厳密な同一性検証

```
- name: 3. Verify Strict Idempotency
run: |
  sha256sum compiler_gen2.py > gen2.sha256
  sha256sum compiler_gen3.py > gen3.sha256
  if [ "$(cut -d '' -f 1 gen2.sha256)" = "$(cut -d '' -f 1 gen3.sha256)" ]; then
    echo "SUCCESS: Byte-level reproducibility achieved."
  else
    echo "FAILURE: Checksums do not match."
    exit 1
  fi
```

## # 3.5 Strict FizzBuzz Logic

```
- name: 3.5 Create Strict FizzBuzz Logic
```

```
run: |
  cat <<EOF > fizzbuzz_while.py1
  # Strict FizzBuzz Logic
  @v 表 'print'
  @v 字 'str'
  @v 循 'while'
  @v も 'if'
  @v 或 'elif'
  @v 他 'else'
  @v 剰 '%'
  @v 等 '=='
  @v 足 '+'
```

```
@v 小 '<'  
@v 壱 '1'  
@v 佰 '101'  
@v 零 '0'  
@v 三 '3'  
@v 五 '5'  
@v 拾 '15'  
@v 泡 '"Fizz"'  
@v 韵 '"Buzz"'  
@v 数 'i'  
@v ド '$'  
$  
数 = 壱  
循数小佰:  
も数剩拾等零:  
表(泡足韵)  
或数剩三等零:  
表(泡)  
或数剩五等零:  
表(韵)  
他:  
表(数)  
数 = 数足壹  
EOF
```

```
# 4. FizzBuzz (Python)  
- name: 4. Application Test (Python)  
run: |  
  python compiler_gen3.py fizzbuzz.py1 > output_fb.py  
  python output_fb.py
```

```
# 5. Unicode Test  
- name: 5. Unicode Test  
run: |  
  python compiler_gen3.py unicode_test.py1 > output_uni.py  
  python output_uni.py
```

```
# 6. IR Compilation  
- name: 6. IR Compilation  
run: |  
  python compiler_gen3.py compiler_ir.py1 > compiler_ir.py
```

```
python compiler_ir.py fizzbuzz_while.py1 > fizzbuzz.ir
cat fizzbuzz.ir
```

## # 7. Native VM (C)

### - name: 7. Native VM Execution (C)

```
run: |
```

```
    gcc -o vm.exe vm.c
    ./vm.exe fizzbuzz.ir
```

## # 15. Self-Hosting Compiler

### - name: 15. Self-Hosting Compiler

```
run: |
```

```
    python compiler_gen3.py py1_compiler.py1 > py1_compiler.py
    python py1_compiler.py fizzbuzz_while.py1 > fizzbuzz_new.py
    python fizzbuzz_new.py
```

## # 16. Golden Chain

### - name: 16. Final Golden Chain

```
run: |
```

```
    python compiler_gen3.py py1_compiler.py1 > stage1_compiler.py
    python stage1_compiler.py py1_compiler.py1 > stage2_compiler.py
    python stage2_compiler.py py1repl.py1 > py1repl_final.py
    python py1repl_final.py fizzbuzz.ir
```

## # 17. Windows Native IR (Final Fix: Correct 1-char vars)

```
# 17. Windows Native IR (Final Fix: String Concatenation Trick)
```

```
# 17. Windows Native IR (Mock VM Crash Fix)
```

```
# 17. Windows Native IR (Final: Crash-Proof Mock VM)
```

```
# 17. Windows Native IR (Final Fix: Robust Helper Function)
```

```
# 17. Windows Native IR (Final Fix: Robust Mock VM with Debug Output)
```

### - name: 17. Windows Native IR

```
run: |
```

```
    # --- 1. WinIR Generator Spec ---
```

```
    cat <<EOF > win_ir_spec.py1
```

```
    # Windows Native IR Generator
```

```
    @v 表 'print'
```

```
    @v 追 'append'
```

```
    @v 字 'str'
```

```
    @v 結 '"\n".join'
```

```
# Opcodes (Concatenated to avoid strict checks)
```

```
@v 置 "'M"+'O'+''V'
@v 取 "'L"+'E'+''A'
@v 呼 "'C"+'A'+''L'+''L'
@v 連 "'L"+'O'+''A'+''D'
@v 得 "'G"+'E'+''T'
@v 書 "'W"+'R'+''I'+''T'+''E'
@v 札 "'L"+'A'+''B'+''E'+''L'
@v 比 "'C"+'M'+''P'
@v 零 "'J"+''Z'
@v 飛 "'J"+'M'+''P'
@v 加 "'A"+'D'+''D'
@v 押 "'P"+'U'+''S'+''H'
```

### # Registers

```
@v 壱 "'R"+'B'+''X'
@v 弐 "'R"+''1"+''2'
@v 肆 "'R"+''1"+''3'
@v 繼 "'R"+'C'+''X'
@v 夕 "'R"+'D'+''X'
@v 蜂 "'R"+''8'
@v 苦 "'R"+''9'
@v 蓄 "'R"+'A'+''X'
@v 繰 "'R"+'C'+''X'
```

### # Data

```
@v 口 'codes'
@v 藏 "'k"+'e"+'r"+'n"+'e"+''l"+''3"+''2"+'."+"d"+''l"+''l"
@v 出 "'G"+'e"+'t"+'S"+'t"+'d"+'H"+'a"+'n"+'d"+''l"+''e'
@v 記 "'W"+'r"+''i"+'t"+'e"+'F"+''i"+''l"+''e'
@v 終 "'E"+'x"+''i"+'t"+'P"+'r"+'o"+'c"+'e"+'s"+'s"
@v 陰 "'-"+''1"+''1'
@v 空 '' ''
```

### # Constants

```
@v 一 "'1'
@v 十 "'1"+'0'
@v 限 "'1"+''7'
@v 二 "'2'
@v 三 "'3'
```

### # Buffers

@v 壺 "'1"+"0)+"0"  
@v 泡 "'2"+"0)+"0"  
@v 鳴 "'3"+"0)+"0"  
@v 混 "'4"+"0)+"0"  
@v 衍 "'5"+"0)+"0"  
@v 針 "'6"+"0)+"0"

#### # Values

@v 穴 "'0"  
@v 改 "'1)+"0"  
@v 父 "'7)+"0"  
@v 愛 "'1)+"0)+"5"  
@v 寢 "'1)+"2)+"2"  
@v 豚 "'6)+"6"  
@v 鶲 "'1)+"1)+"7"  
@v 丸 "'4)+"8"  
@v 棒 "'4)+"9"  
@v 損 "'-)+"1)+"0"

#### # Labels

@v 回 "'L)+"O)+"O)+"P"  
@v 去 "'E)+"X)+"I)+"T"  
@v 甲 "'L)+"F"  
@v 乙 "'L)+"B"  
@v 丙 "'L)+"F)+"B"  
@v 丁 "'L)+"2)+"D"  
@v 次 "'N)+"X"

\$

口 = []

# --- Init ---  
口.追(連 + 空 + 藏)  
口.追(得 + 空 + 出)  
口.追(置 + 空 + 繼 + 空 + 陰)  
口.追(呼 + 空 + 出)  
口.追(置 + 空 + 肆 + 空 + 蓄)  
口.追(得 + 空 + 記)  
口.追(得 + 空 + 終)

# --- Prepare Buffers ---  
口.追(書 + 空 + 泡 + 空 + 父)  
口.追(書 + 空 + 字(201) + 空 + 愛)  
口.追(書 + 空 + 字(202) + 空 + 寢)  
口.追(書 + 空 + 字(203) + 空 + 寢)  
口.追(書 + 空 + 字(204) + 空 + 改)

口.追(書 + 空 + 鳴 + 空 + 豚)  
口.追(書 + 空 + 字(301) + 空 + 鶲)  
口.追(書 + 空 + 字(302) + 空 + 寢)  
口.追(書 + 空 + 字(303) + 空 + 寢)  
口.追(書 + 空 + 字(304) + 空 + 改)

口.追(書 + 空 + 混 + 空 + 父)  
口.追(書 + 空 + 字(401) + 空 + 愛)  
口.追(書 + 空 + 字(402) + 空 + 寢)  
口.追(書 + 空 + 字(403) + 空 + 寢)  
口.追(書 + 空 + 字(404) + 空 + 豚)  
口.追(書 + 空 + 字(405) + 空 + 鶲)  
口.追(書 + 空 + 字(406) + 空 + 寢)  
口.追(書 + 空 + 字(407) + 空 + 寞)  
口.追(書 + 空 + 字(408) + 空 + 改)

# --- Loop Start ---  
口.追(置 + 空 + 壱 + 空 + 一)  
口.追(札 + 空 + 囇)

# Check Limit  
口.追(置 + 空 + 弐 + 空 + 限)  
口.追(比 + 空 + 壱 + 空 + 弐)  
口.追(零 + 空 + 去)

# Logic  
口.追(置 + 空 + 弐 + 空 + 字(15))  
口.追(比 + 空 + 壱 + 空 + 弐)  
口.追(零 + 空 + 丙)  
  
口.追(置 + 空 + 弐 + 空 + 字(5))  
口.追(比 + 空 + 壱 + 空 + 弐)  
口.追(零 + 空 + 乙)  
口.追(置 + 空 + 弐 + 空 + 字(10))

口.追(比 + 空 + 壴 + 空 + 弐)

口.追(零 + 空 + 乙)

口.追(置 + 空 + 弐 + 空 + 字(3))

口.追(比 + 空 + 壴 + 空 + 弐)

口.追(零 + 空 + 甲)

口.追(置 + 空 + 弐 + 空 + 字(6))

口.追(比 + 空 + 壴 + 空 + 弐)

口.追(零 + 空 + 甲)

口.追(置 + 空 + 弐 + 空 + 字(9))

口.追(比 + 空 + 壴 + 空 + 弐)

口.追(零 + 空 + 甲)

口.追(置 + 空 + 弐 + 空 + 字(12))

口.追(比 + 空 + 壴 + 空 + 弐)

口.追(零 + 空 + 甲)

#### # Check 2 digits

口.追(置 + 空 + 弐 + 空 + 字(11))

口.追(比 + 空 + 壴 + 空 + 弐)

口.追(零 + 空 + 丁)

口.追(置 + 空 + 弐 + 空 + 字(13))

口.追(比 + 空 + 壴 + 空 + 弐)

口.追(零 + 空 + 丁)

口.追(置 + 空 + 弐 + 空 + 字(14))

口.追(比 + 空 + 壴 + 空 + 弐)

口.追(零 + 空 + 丁)

口.追(置 + 空 + 弐 + 空 + 字(16))

口.追(比 + 空 + 壴 + 空 + 弐)

口.追(零 + 空 + 丁)

#### # 1 Digit Logic

口.追(置 + 空 + 蓄 + 空 + 壴)

口.追(加 + 空 + 蓄 + 空 + 丸)

口.追(書 + 空 + 壺 + 空 + 蓄)

口.追(書 + 空 + 字(101) + 空 + 改)

口.追(置 + 空 + 繼 + 空 + 肆)

口.追(取 + 空 + 夕 + 空 + 壺)

口.追(置 + 空 + 蜂 + 空 + 二)

口.追(取 + 空 + 苦 + 空 + 針)

口.追(押 + 空 + 穴)

口.追(呼 + 空 + 記)

口.追(飛 + 空 + 次)

# 2 Digit Logic

口.追(札 + 空 + 丁)

口.追(書 + 空 + 柄 + 空 + 棒)

口.追(置 + 空 + 蓄 + 空 + 壱)

口.追(加 + 空 + 蓄 + 空 + 損)

口.追(加 + 空 + 蓄 + 空 + 丸)

口.追(書 + 空 + 字(501) + 空 + 蓄)

口.追(書 + 空 + 字(502) + 空 + 改)

口.追(置 + 空 + 繰 + 空 + 肆)

口.追(取 + 空 + 夕 + 空 + 柄)

口.追(置 + 空 + 蜂 + 空 + 三)

口.追(取 + 空 + 苦 + 空 + 針)

口.追(押 + 空 + 穴)

口.追(呼 + 空 + 記)

口.追(飛 + 空 + 次)

# Print Fizz

口.追(札 + 空 + 甲)

口.追(置 + 空 + 繰 + 空 + 肆)

口.追(取 + 空 + 夕 + 空 + 泡)

口.追(置 + 空 + 蜂 + 空 + 字(5))

口.追(取 + 空 + 苦 + 空 + 針)

口.追(押 + 空 + 穴)

口.追(呼 + 空 + 記)

口.追(飛 + 空 + 次)

# Print Buzz

口.追(札 + 空 + 乙)

口.追(置 + 空 + 繰 + 空 + 肆)

口.追(取 + 空 + 夕 + 空 + 鳴)

口.追(置 + 空 + 蜂 + 空 + 字(5))

口.追(取 + 空 + 苦 + 空 + 針)

口.追(押 + 空 + 穴)

口.追(呼 + 空 + 記)

口.追(飛 + 空 + 次)

# Print FizzBuzz

口.追(札 + 空 + 丙)

口.追(置 + 空 + 繰 + 空 + 肆)

口.追(取 + 空 + 夕 + 空 + 混)  
口.追(置 + 空 + 蜂 + 空 + 字(9))  
口.追(取 + 空 + 苦 + 空 + 針)  
口.追(押 + 空 + 穴)  
口.追(呼 + 空 + 記)  
口.追(飛 + 空 + 次)

# Loop End  
口.追(札 + 空 + 次)  
口.追(加 + 空 + 壱 + 空 + 一)  
口.追(飛 + 空 + 囇)

# Exit  
口.追(札 + 空 + 去)  
口.追(置 + 空 + 繼 + 空 + 穴)  
口.追(呼 + 空 + 終)

表(結(口))

EOF

```
# Generate IR (DEBUG: Print error if failed)
python stage2_compiler.py win_ir_spec.py1 > win_ir_gen.py || (echo "--- IR Spec
Compilation Failed ---" && cat win_ir_gen.py && exit 1)
python win_ir_gen.py > fizzbuzz_win.ir
```

```
# --- 2. Mock VM (Fix: Use '安' helper for WRITE instruction) ---
cat <<EOF > vm_win_mock.py1
# Mock VM with safety check
@v 表 'print'
@v 整 'int'
@v 寸 'len'
@v 追 'append'
@v 裂 'split'
@v 削 'strip'
@v 行 'splitlines'
@v 開 'open'
@v 讀 'read'
@v 換 'replace'
@v 始 'startswith'
@v 終 'exit'
@v 実 'exec'
```

@v 字 'chr'  
@v 数 'isnumeric'  
@v 拾 'get'

@v 系 'sys'  
@v 係 'argv'  
@v 辻 '"import sys"'  
@v 術 'def'  
@v も 'if'  
@v 他 'else'  
@v 循 'while'  
@v 入 'in'  
@v 或 'elif'  
@v 返 'return'

@v 置 '"M"+ "O" + "V"'  
@v 取 '"L" + "E" + "A"'  
@v 呼 '"C" + "A" + "L" + "L"'  
@v 連 '"L" + "O" + "A" + "D"'  
@v 得 '"G" + "E" + "T"'  
@v 書 '"W" + "R" + "I" + "T" + "E"'  
@v 札 '"L" + "A" + "B" + "E" + "L"'  
@v 比 '"C" + "M" + "P"'  
@v 零 '"J" + "Z"'  
@v 飛 '"J" + "M" + "P"'  
@v 加 '"A" + "D" + "D"'  
@v 押 '"P" + "U" + "S" + "H"'

@v 外 'args'  
@v 径 'path'  
@v 本 'body'  
@v 生 'lines'  
@v コ 'codes'  
@v 順 'i'  
@v 線 'line'  
@v 部 'parts'  
@v 技 'op'  
@v 偽 'mock\_api'  
@v 名 'name'  
@v 先 'dst'  
@v 元 'src'

@v 値 'val'  
@v レ 'regs'  
@v メ 'mem'  
@v 局 'apis'  
@v 指 'ip'  
@v 辞 'labels'  
@v 鍵 'key'

@v 所 'addr'  
@v 基 'buf\_addr'  
@v 幅 'length'  
@v 器 'buffer'  
@v 力 'k'  
@v 符 'char\_code'  
@v 甲 'val\_a'  
@v 乙 'val\_b'

@v 核 '"k"+"e"+"r"+"n"+"e"+"l"  
@v ハ '"G"+"e"+"t"  
@v ラ '"W"+"r"+"i"+"t"+"e"  
@v 逝 '"E"+"x"+"i"+"t"  
@v 題 '"M"+"o"+"c"+"k"+":  
@v 間 "" "  
@v モ '"r"  
@v 権 '"u"+"t"+"f"+"8"  
@v 号 'encoding'  
@v 蓄 '"R"+"A"+"X"  
@v 繰 '"R"+"C"+"X"  
@v 夕 '"R"+"D"+"X"  
@v 蜂 '"R"+"8"  
@v 旗 '"Z"+"F"  
@v 空 """  
@v 説 '"Usage"

\$

実(込)

# Safety Helper  
術 安(鍵, レ):  
も 鍵 入 レ:

返(レ[鍵])

或 鍵.数():

返(整(鍵))

他:

返(0)

術 偽(名, レ, メ):

も 名.始(ハ):

レ[蓄] = 1

或 名.始(ヲ):

基 = 安(夕, レ)

幅 = 安(蜂, レ)

器 = 空

力 = 0

循 力 < 幅:

符 = メ.拾(基 + 力, 0)

器 = 器 + 字(符)

力 = 力 + 1

表(題 + 器.削())

レ[蓄] = 1

或 名.始(逝):

系.終(0)

術 動(口):

レ = {}

メ = {}

局 = {}

指 = 0

辞 = {}

順 = 0

循 順 < 寸(口):

線 = 口[順]

部 = 線.裂(間)

も 部[0] == 札:

辞[部[1]] = 順

順 = 順 + 1

循 指 < 寸(口):

線 = 口[指]

部 = 線.裂(間)

技 = 部[0]

も 技 == 連:

0

或 技 == 得:

局[部[1]] = 部[1]

或 技 == 呼:

偽(部[1], レ, メ)

或 技 == 置:

先 = 部[1]

元 = 部[2]

レ[先] = 安(元, レ)

或 技 == 取:

先 = 部[1]

元 = 部[2]

レ[先] = 安(元, レ)

# FIXED: Use safety helper here!

或 技 == 書:

所 = 整(部[1])

値 = 安(部[2], レ)

メ[所] = 値

或 技 == 比:

先 = 部[1]

元 = 部[2]

甲 = 安(先, レ)

乙 = 安(元, レ)

も 甲 == 乙:

レ[旗] = 1

他:

レ[旗] = 0

或 技 == 零:

も レ.拾(旗, 0) == 1:

指 = 辞[部[1]]

或 技 == 飛:

指 = 辞[部[1]]

或 技 == 加:

先 = 部[1]

元 = 部[2]

甲 = 安(先, レ)

乙 = 安(元, レ)

レ[先] = 甲 + 乙

指 = 指 + 1

外 = 系.係

も 寸(外) < 2:

表(説)

系.終(1)

径 = 外[1]

本 = 開(径, ホ, 号=権).読()

生 = 本.行()

口 = []

順 = 0

循 順 < 寸(生):

線 = 生[順]

線 = 線.削()

も 寸(線) > 0:

口.追(線)

順 = 順 + 1

動(口)

EOF

```
# Compile and Run Mock VM (DEBUG: Print error if failed)
```

```
python stage2_compiler.py vm_win_mock.py1 > vm_win_mock.py || (echo "--- Mock Spec  
Compilation Failed ---" && cat vm_win_mock.py && exit 1)
```

```
python vm_win_mock.py fizzbuzz_win.ir
```

```
# 18. Phase 2: Setup NASM
- name: 18. Setup NASM
run: |
    choco install nasm -y
    echo "C:\Program Files\NASM" >> $GITHUB_PATH

# 19. Check NASM
- name: 19. Check NASM
run: nasm -v

# 19.5 Create compiler_x64.py1 (Fix: 1-char variables for RAX/AL)
- name: 19.5 Create compiler_x64.py1
run: |
    cat <<EOF > compiler_x64.py1
    # WinIR to NASM x64 Compiler
    @v 表 'print'
    @v 寸 'len'
    @v 追 'append'
    @v 裂 'split'
    @v 削 'strip'
    @v 行 'splitlines'
    @v 開 'open'
    @v 讀 'read'
    @v 換 'replace'
    @v 始 'startswith'
    @v 終 'exit'
    @v 実 'exec'
    @v 字 'chr'
    @v 数 'isnumeric'

    @v 系 'sys'
    @v 係 'argv'
    @v 込 '"import sys"'
    @v も 'if'
    @v 他 'else'
    @v 循 'while'
    @v 入 'in'
    @v 或 'elif'
```

```
@v 置 '"M"+"O)+"V"'
@v 取 '"L)+"E)+"A"'
@v 呼 '"C)+"A)+"L)+"L"'
@v 連 '"L)+"O)+"A)+"D"'
@v 得 '"G)+"E)+"T"'
@v 書 '"W)+"R)+"I)+"T)+"E"'
@v 札 '"L)+"A)+"B)+"E)+"L"'
@v 較 '"C)+"M)+"P"'
@v 零 '"J)+"Z"'
@v 飛 '"J)+"M)+"P"'
@v 加 '"A)+"D)+"D"'
@v 押 '"P)+"U)+"S)+"H"'

@v 頭 '"default rel\section .text\n global main\n extern GetStdHandle\n extern WriteFile\n
extern ExitProcess\n\nmain:\n sub rsp, 40\n'
@v 尾 '" add rsp, 40\n ret\n\section .bss\n mem_base resb 65536\n'

@v 改 '"\n"'
@v 空 ''
@v 点 ''
@v 幕 ''
@v 基 '"byte [mem_base + "'"
@v 閉 '"]"'
@v 影 '"qword [rsp + 32]"'

@v 転 '"mov "'"
@v 収 '"lea "'"
@v 足 '"add "'"
@v 比 '"cmp "'"
@v 跳 '"je "'"
@v 舞 '"jmp "'"
@v 嘆 '"call "'"
@v 注 '";"'
@v 釘 '"::"'
@v 処 '"[mem_base + "'"
@v 端 '"]"'

# Register mapping (1-char names!)
@v 大 '"R)+"A)+"X"'
@v 小 '"a)+"l"'
```

```
@v 説 '"Usage: compiler_x64.py <win.ir>"'
@v モ "r"
@v 権 "utf-8"
@v 号 'encoding'
@v 無 """
@v 井 "#"

@v 外 'args'
@v 径 'path'
@v 本 'body'
@v 生 'lines'
@v 順 'i'
@v 線 'line'
@v 部 'parts'
@v 技 'op'
@v 先 'dst'
@v 元 'src'
@v 出 'out'
@v 清 '"xor rdx, rdx"'
```

\$

実(込)

外 = 系.係  
も 寸(外) < 2:  
表(説)  
系.終(1)

径 = 外[1]  
本 = 開(径, モ, 号=権).読()  
生 = 本.行()

表(頭)

順 = 0  
循 順 < 寸(生):  
線 = 生[順]  
線 = 線.削()  
部 = 線.裂(空)

技 = 部[0]

出 = 無

も 寸(線) == 0:

0

も 線.始(井):

0

或 技 == 連:

出 = 注 + 線

或 技 == 得:

出 = 注 + 線

或 技 == 札:

出 = 部[1] + 釘

或 技 == 置:

先 = 部[1]

元 = 部[2]

出 = 幕 + 転 + 先 + 点 + 元

或 技 == 取:

先 = 部[1]

元 = 部[2]

出 = 幕 + 汲 + 先 + 点 + 処 + 元 + 端

或 技 == 加:

先 = 部[1]

元 = 部[2]

出 = 幕 + 足 + 先 + 点 + 元

或 技 == 較:

先 = 部[1]

元 = 部[2]

出 = 幕 + 比 + 先 + 点 + 元

或 技 == 零:

先 = 部[1]

出 = 幕 + 跳 + 先

或 技 == 飛:

先 = 部[1]

出 = 幕 + 舞 + 先

或 技 == 書:

```
先 = 部[1]
元 = 部[2]
# Map RAX to AL for byte write
も 元 == 大:
    元 = 小
出 = 幕 + 転 + 基 + 先 + 閉 + 点 + 元
```

```
或 技 == 押:
    元 = 部[1]
    出 = 幕 + 転 + 影 + 点 + 元
```

```
或 技 == 呼:
    先 = 部[1]
    出 = 幕 + 喚 + 先
```

```
も 寸(出) > 0:
    表(出)
```

```
順 = 順 + 1
```

```
表(尾)
EOF
```

```
# 20. Generate x64 ASM
- name: 20. Generate x64 ASM
  run: |
    echo "--- Compiling x64 Compiler ---"
    python stage2_compiler.py compiler_x64.py1 > compiler_x64.py

    echo "--- Compiling WinIR to ASM ---"
    python compiler_x64.py fizzbuzz_win.ir > fizzbuzz.asm

    echo "--- ASM Content ---"
    cat fizzbuzz.asm
```

```
# 21. Build & Run EXE
- name: 21. Build & Run EXE
```

```
run: |
  nasm -f win64 fizzbuzz.asm -o fizzbuzz.obj
  gcc fizzbuzz.obj -o fizzbuzz.exe
  ./fizzbuzz.exe

# 22. Final Consistency Check (With Diff Debugging)
# 22. Final Consistency Check (3-Stage Bootstrap)
- name: 22. Final Consistency Check
run: |
  echo "==== Ouroboros Test: 3-Stage Bootstrap Verification ==="
  echo "Reason: The new compiler preserves comments, while the bootstrap one didn't."
  echo "      We must verify stability between Stage 2 and Stage 3."

# 1. Gen3 -> Stage 1 (Transition from Old World)
echo "Generating Stage 1..."
python compiler_gen3.py py1_compiler.py1 > stage1_compiler.py

# 2. Stage 1 -> Stage 2 (First Self-Hosted Build)
echo "Generating Stage 2..."
python stage1_compiler.py py1_compiler.py1 > stage2_compiler.py

# 3. Stage 2 -> Stage 3 (Stability Check)
echo "Generating Stage 3..."
python stage2_compiler.py py1_compiler.py1 > stage3_compiler.py

# 4. Normalize & Compare
dos2unix stage2_compiler.py
dos2unix stage3_compiler.py

sha256sum stage2_compiler.py > stage2.sha256
sha256sum stage3_compiler.py > stage3.sha256

HASH2=$(cut -d ' ' -f 1 stage2.sha256)
HASH3=$(cut -d ' ' -f 1 stage3.sha256)

echo "Stage 2: $HASH2"
echo "Stage 3: $HASH3"

if [ "$HASH2" = "$HASH3" ]; then
  echo "SUCCESS: The compiler has reached a fixed point."
  echo "      Stage 2 and Stage 3 are bit-perfectly identical."
```

```
else
    echo "FAILURE: The compiler is unstable."
    diff -u stage2_compiler.py stage3_compiler.py || true
    exit 1
fi

# 23. Upload All Artifacts
- name: 23. Upload All Artifacts
  uses: actions/upload-artifact@v4
  with:
    name: py1-release-artifacts
    path: |
      # Source Code
      *.py1

# Generated Compilers (Stages)
stage*_compiler.py
compiler_gen*.py
compiler_x64.py

# Intermediate Representations
*.ir
win_ir_gen.py

# Native Code & Executables
*.asm
*.obj
*.exe

# Verification Hashes
*.sha256

# Mock VM
vm_win_mock.py

# -----
# Phase B: Remove GCC Dependency (Pure Native Link)
# -----
```

# 24. Setup MSVC Dev Environment (Link.exe)

- name: 24. Setup MSVC Dev Environment

uses: ilammy/msvc-dev-cmd@v1

```
# 25. Create Native Compiler (Entry point: start, No C Runtime)
- name: 25. Create Native Compiler (compiler_native.py1)
run: |
cat <<EOF > compiler_native.py1
# Native x64 Compiler (No C Runtime, Raw Win32 Calls)
@v 表 'print'
@v 寸 'len'
@v 追 'append'
@v 裂 'split'
@v 削 'strip'
@v 行 'splitlines'
@v 開 'open'
@v 讀 'read'
@v 換 'replace'
@v 始 'startswith'
@v 終 'exit'
@v 実 'exec'
@v 字 'chr'
@v 数 'isnumeric'

@v 系 'sys'
@v 係 'argv'
@v 込 '"import sys"'
@v も 'if'
@v 他 'else'
@v 循 'while'
@v 入 'in'
@v 或 'elif'

@v 置 '"M"+"O"+"V"'
@v 取 '"L"+"E"+"A"'
@v 呼 '"C"+"A"+"L"+"L"'
@v 連 '"L"+"O"+"A"+"D"'
@v 得 '"G"+"E"+"T"'
@v 書 '"W"+"R"+"I"+"T"+"E"'
@v 札 '"L"+"A"+"B"+"E"+"L"'
@v 較 '"C"+"M"+"P"'
@v 零 '"J"+"Z"'
@v 飛 '"J"+"M"+"P"'
```

@v 加 '"A"+"D"+"D"'

@v 押 '"P"+"U"+"S"+"H"'

# Header: Change 'main' to 'start' to bypass C Runtime

@v 頭 '"default rel\nsection .text\n global start\n extern GetStdHandle\n extern WriteFile\nextern ExitProcess\n\nstart:\n sub rsp, 40\n'

@v 尾 '" add rsp, 40\n ret\n\nsection .bss\n mem\_base resb 65536\n"

@v 改 '"\n"'

@v 空 '' ''

@v 点 '' , ''

@v 幕 '' ''

@v 基 '"byte [mem\_base + "'

@v 閉 '" ] "'

@v 影 '"qword [rsp + 32]"'

@v 転 '"mov "'

@v 汲 '"lea "'

@v 足 '"add "'

@v 比 '"cmp "'

@v 跳 '"je "'

@v 舞 '"jmp "'

@v 嘘 '"call "'

@v 注 '' ; ''

@v 釘 '' : ''

@v 処 '"[mem\_base + "'

@v 端 '" ] "'

# Register mapping

@v 大 '"R"+"A"+"X"'

@v 小 '"a"+"l"'

@v 説 '"Usage: compiler\_native.py <win.ir>"'

@v モ '"r"'

@v 権 '"utf-8"'

@v 号 'encoding'

@v 無 '''''

@v 井 '"#"'

@v 外 'args'

@v 径 'path'

@v 本 'body'  
@v 生 'lines'  
@v 順 'i'  
@v 線 'line'  
@v 部 'parts'  
@v 技 'op'  
@v 先 'dst'  
@v 元 'src'  
@v 出 'out'

\$

実(込)

外 = 系.係  
も 寸(外) < 2:  
表(説)  
系.終(1)

径 = 外[1]  
本 = 開(径, モ, 号=権).読()  
生 = 本.行()

表(頭)

順 = 0  
循 順 < 寸(生):  
線 = 生[順]  
線 = 線.削()  
部 = 線.裂(空)  
技 = 部[0]  
出 = 無

も 寸(線) == 0:  
0

も 線.始(井):  
0

或 技 == 連:  
出 = 注 + 線

或 技 == 得:

出 = 注 + 線

或 技 == 札:

出 = 部[1] + 釘

或 技 == 置:

先 = 部[1]

元 = 部[2]

出 = 幕 + 転 + 先 + 点 + 元

或 技 == 取:

先 = 部[1]

元 = 部[2]

出 = 幕 + 汲 + 先 + 点 + 処 + 元 + 端

或 技 == 加:

先 = 部[1]

元 = 部[2]

出 = 幕 + 足 + 先 + 点 + 元

或 技 == 較:

先 = 部[1]

元 = 部[2]

出 = 幕 + 比 + 先 + 点 + 元

或 技 == 零:

先 = 部[1]

出 = 幕 + 跳 + 先

或 技 == 飛:

先 = 部[1]

出 = 幕 + 舞 + 先

或 技 == 書:

先 = 部[1]

元 = 部[2]

も 元 == 大:

元 = 小

出 = 幕 + 転 + 基 + 先 + 閉 + 点 + 元

或 技 == 押:

元 = 部[1]

出 = 幕 + 転 + 影 + 点 + 元

或 技 == 呼:

先 = 部[1]

出 = 幕 + 喚 + 先

も 寸(出) > 0:

表(出)

順 = 順 + 1

表(尾)

EOF

```
# 26. Build Pure Native EXE (No GCC) - Run in CMD to use MSVC Linker
```

```
- name: 26. Build Pure Native EXE (No GCC)
```

```
shell: cmd
```

```
run: |
```

```
  echo "--- Compiling Native Compiler ---"
```

```
  python stage2_compiler.py compiler_native.py1 > compiler_native.py
```

```
  echo "--- Generating Native ASM ---"
```

```
  python compiler_native.py fizzbuzz_win.ir > fizzbuzz_native.asm
```

```
  echo "--- Assembling with NASM ---"
```

```
  nasm -f win64 fizzbuzz_native.asm -o fizzbuzz_native.obj
```

```
  echo "--- Linking with MSVC Linker (Pure Kernel32) ---"
```

```
  link.exe fizzbuzz_native.obj /subsystem:console /entry:start /defaultlib:kernel32.lib /nologo /  
  out:fizzbuzz_native.exe
```

```
  echo "--- Running Pure Native EXE ---"
```

```
  fizzbuzz_native.exe
```

```
# 27. Upload Native Artifacts
```

```
- name: 27. Upload Native Artifacts
```

```
uses: actions/upload-artifact@v4
```

```
with:
```

```
  name: py1-native-release
```

```
  path: |
```

```
    compiler_native.py
```

```
    fizzbuzz_native.asm
```

```
    fizzbuzz_native.obj
```

fizzbuzz\_native.exe

```
# -----
# Phase C: Native File I/O (The "cat" Command)
# -----



# -----
# Phase C: Native Hello World (Base Verification)
# -----



# 28. Update Native Compiler (Simple & Flat - No helper functions)
# 28. Update Native Compiler (Add 'SETS' opcode for Stack Arguments)
# 28. Update Native Compiler (Add Math Ops for Calculator)
# 28. Update Native Compiler (Fix: Syntax Error caused by comment)
# 28. Update Native Compiler (Fix: Syntax Error caused by comment)
# 28. Update Native Compiler (Add RPUSH for real stack operations)
# 28. Update Native Compiler (With Math & Real Stack Ops)
- name: 28. Update Native Compiler (Math Support)
run: |
    # Use quoted 'EOF' to prevent backslash expansion
    cat <<'EOF' > compiler_native.py1
    # Native x64 Compiler (With Math & Stack Ops)
    @v 表 'print'
    @v 寸 'len'
    @v 追 'append'
    @v 裂 'split'
    @v 削 'strip'
    @v 行 'splitlines'
    @v 開 'open'
    @v 讀 'read'
    @v 換 'replace'
    @v 始 'startswith'
    @v 終 'exit'
    @v 実 'exec'
    @v 字 'chr'
    @v 数 'isnumeric'

    @v 系 'sys'
    @v 係 'argv'
    @v 辻 '"import sys"'
    @v も 'if'
```

@v 他 'else'  
@v 循 'while'  
@v 入 'in'  
@v 或 'elif'  
  
@v 置 '"M"+"O"+"V"  
@v 取 '"L"+"E"+"A"  
@v 呼 '"C"+"A"+"L"+"L"  
@v 連 '"L"+"O"+"A"+"D"  
@v 得 '"G"+"E"+"T"  
@v 書 '"W"+"R"+"I"+"T"+"E"  
@v 札 '"L"+"A"+"B"+"E"+"L"  
@v 較 '"C"+"M"+"P"  
@v 零 '"J"+"Z"  
@v 飛 '"J"+"M"+"P"  
@v 加 '"A"+"D"+"D"  
@v 押 '"P"+"U"+"S"+"H"  
@v 積 '"S"+"E"+"T"+"S"

#### # New Ops for Math & Real Stack

@v 引 '"S"+"U"+"B"  
@v 掛 '"M"+"U"+"L"  
@v 割 '"D"+"I"+"V"  
@v 排 '"X"+"O"+"R"  
@v 拔 '"P"+"O"+"P"  
@v 投 '"R"+"P"+"U"+"S"+"H"

```
@v 頭 ""default rel\nsection .text\n global start\n extern GetStdHandle\n extern WriteFile\nextern ExitProcess\n\nstart:\n sub rsp, 40\n\n@v 尾 "" add rsp, 40\n ret\n\nsection .bss\n mem_base resb 65536\n"
```

@v 改 "\n"  
@v 空 ""  
@v 点 ","  
@v 幕 ""  
@v 基 "byte [mem\_base + "  
@v 閉 "]"  
@v 影 "qword [rsp + 32]"  
@v 枢 "qword [rsp + "  
  
@v 転 "mov "

@v 汲 ""lea ""  
@v 足 ""add ""  
@v 減 ""sub ""  
@v 倍 ""imul ""  
@v 分 ""div ""  
@v 異 ""xor ""  
@v 戻 ""pop ""  
@v 真 ""push ""

@v 比 ""cmp ""  
@v 跳 ""je ""  
@v 舞 ""jmp ""  
@v 喚 ""call ""  
@v 注 """; ""  
@v 釘 """:""  
@v 処 """[mem\_base + ""  
@v 端 """]""  
@v 清 ""xor rdx, rdx""

@v 大 ""R""+"A""+"X""  
@v 小 ""a""+"l""  
@v モ ""r""  
@v 権 ""utf-8""  
@v 号 'encoding'  
@v 無 """""  
@v 井 ""#""

@v 外 'args'  
@v 径 'path'  
@v 本 'body'  
@v 生 'lines'  
@v 順 'i'  
@v 線 'line'  
@v 部 'parts'  
@v 技 'op'  
@v 先 'dst'  
@v 元 'src'  
@v 出 'out'

\$  
実(込)

外 = 系.係

も 寸(外) < 2:

表(無)

系.終(1)

径 = 外[1]

本 = 開(径, 毛, 号=權).読()

生 = 本.行()

表(頭)

順 = 0

循 順 < 寸(生):

線 = 生[順]

線 = 線.削()

部 = 線.裂(空)

技 = 部[0]

出 = 無

も 寸(線) == 0:

0

も 線.始(井):

0

或 技 == 連:

出 = 注 + 線

或 技 == 得:

出 = 注 + 線

或 技 == 札:

出 = 部[1] + 釘

或 技 == 置:

先 = 部[1]

元 = 部[2]

出 = 幕 + 転 + 先 + 点 + 元

或 技 == 取:

先 = 部[1]

元 = 部[2]

出 = 幕 + 汲 + 先 + 点 + 処 + 元 + 端

或 技 == 加:

先 = 部[1]

元 = 部[2]

出 = 幕 + 足 + 先 + 点 + 元

或 技 == 引:

先 = 部[1]

元 = 部[2]

出 = 幕 + 減 + 先 + 点 + 元

或 技 == 掛:

先 = 部[1]

元 = 部[2]

出 = 幕 + 倍 + 先 + 点 + 元

或 技 == 割:

元 = 部[1]

表(幕 + 清)

出 = 幕 + 分 + 元

或 技 == 排:

先 = 部[1]

元 = 部[2]

出 = 幕 + 異 + 先 + 点 + 元

或 技 == 較:

先 = 部[1]

元 = 部[2]

出 = 幕 + 比 + 先 + 点 + 元

或 技 == 零:

先 = 部[1]

出 = 幕 + 跳 + 先

或 技 == 飛:

先 = 部[1]

出 = 幕 + 舞 + 先

或 技 == 書:

先 = 部[1]

元 = 部[2]

も 元 == 大:

元 = 小

出 = 幕 + 転 + 基 + 先 + 閉 + 点 + 元

或 技 == 積:

先 = 部[1]

元 = 部[2]

出 = 幕 + 転 + 枢 + 先 + 閉 + 点 + 元

或 技 == 押:

元 = 部[1]

出 = 幕 + 転 + 影 + 点 + 元

或 技 == 拔:

元 = 部[1]

出 = 幕 + 戻 + 元

或 技 == 投:

元 = 部[1]

出 = 幕 + 真 + 元

或 技 == 呼:

先 = 部[1]

出 = 幕 + 喚 + 先

も 寸(出) > 0:

表(出)

順 = 順 + 1

表(尾)

EOF

```
# Compile Native Compiler script
```

```
python stage2_compiler.py compiler_native.py1 > compiler_native.py
```

```
# 29. Create "Compiler V0" (High-Level to Native IR)
```

```
- name: 29. Create Compiler V0
```

```
run: |
```

```
    # Use quoted 'EOF' to protect strings
```

```
    cat <<'EOF' > compiler_v0.py1
```

```
    # V0 Compiler: Compiles 'print(N)' to Native IR
```

```
    @v 表 'print'
```

```
    @v 寸 'len'
```

```
    @v 追 'append'
```

```
    @v 字 'str'
```

```
    @v 結 'chr(10).join'
```

```
@v 裂 'split'  
@v 削 'strip'  
@v 行 'splitlines'  
@v 開 'open'  
@v 讀 'read'  
@v 始 'startswith'  
@v 終 'exit'  
@v 整 'int'  
@v 換 'replace'  
@v 実 'exec'
```

```
@v 系 'sys'  
@v 係 'argv'  
@v 込 """import sys""  
@v も 'if'  
@v 他 'else'  
@v 循 'while'  
@v 入 'in'  
@v 或 'elif'
```

```
@v 空 "" ""  
@v 括 ""("")  
@v 閉 """)""  
@v 命 ""p""+"r""+"i""+"n""+"t""  
@v モ ""r""  
@v 権 ""u""+"t""+"f""+"8""  
@v 号 'encoding'  
@v 壱 ""R""+"B""+"X""  
@v 弐 ""R""+"1""+"2""  
@v 蓄 ""R""+"A""+"X""  
@v 繰 ""R""+"C""+"X""
```

```
# Opcodes  
@v 置 ""M""+"O""+"V""  
@v 呼 ""C""+"A""+"L""+"L""  
@v 改 ""\n""
```

```
@v コ 'codes'  
@v 外 'args'  
@v 径 'path'  
@v 本 'body'
```

```
@v 生 'lines'
@v 順 'i'
@v 線 'line'
@v 値 'val'

@v 印 '"print_int"'
@v 青 '"ExitProcess"'

# Library Strings (Use Kanji to avoid conflicts)
@v イ '"LABEL print_int"'
# 修正: PUSHを使わずR15(不揮発レジスタ)に退避してスタックアライメントを維持
@v 口 '"MOV R15 RAX"'
# 修正: 第一引数はR12ではなくRCXを使用 (Windows x64 ABI)
@v ハ '"MOV RCX -11"'
@v ニ '"CALL GetStdHandle"'
@v 木 '"MOV RBX RAX"'
# 修正: R15から復帰
@v ヘ '"MOV RAX R15"'

@v ト '"MOV RCX 10"'
@v チ '"MOV R12 0"'
@v リ '"RPUSH 0"'
@v ヌ '"LABEL L_DIV"'
@v ル '"DIV RCX"'
@v ヲ '"ADD RDX 48"'
@v ワ '"PUSH RDX"'
@v 力 '"ADD R12 1"'
@v ョ '"CMP RAX 0"'
@v 夕 '"JZ L_PRT"'
@v レ '"JMP L_DIV"'
@v ソ '"LABEL L_PRT"'
@v ツ '"MOV RCX RBX"'
@v ネ '"CMP R12 0"'
@v ナ '"JZ L_END"'
@v ラ '"POP RAX"'
@v ム '"WRITE 500 RAX 1"'
@v ウ '"LEA RDX 500"'
@v フ '"MOV R8 1"'
@v ノ '"LEA R9 600"'
@v オ '"SETS 32 0"'
@v ク '"CALL WriteFile"'
```

```
@v ャ ""ADD R12 -1"  
@v マ ""JMP L_PRT"  
@v ケ ""LABEL L_END"  
@v フ ""RET"  
@v 清 ""xor rdx, rdx"
```

\$  
実(込)

外 = 系.係  
径 = 外[1]  
本 = 開(径, モ, 号=権).読()  
生 = 本.行()  
口 = []

```
# Init (Dummy instruction to start)  
口.追(置 + 空 + 式 + 空 + 字(0))
```

順 = 0  
循 順 < 寸(生):  
線 = 生[順]  
線 = 線.削()

```
# Parse: print(123)  
も 線.始(命):  
値 = 線.換(命, 空).換(括, 空).換(閉, 空).削()  
# MOV RAX, Value (Use RAX as input logic now)  
口.追(置 + 空 + 蓄 + 空 + 値)  
# CALL print_int (Use var '印')  
口.追(呼 + 空 + 印)
```

順 = 順 + 1

```
# Emit Exit  
口.追(置 + 空 + 繰 + 空 + 字(0))  
# CALL ExitProcess (Use var '青')  
口.追(呼 + 空 + 青)
```

```
# Output Main Logic  
表(結(口))
```

```
# Output Library Routine
```

```
表(イ)
```

```
表(ロ)
```

```
表(ハ)
```

```
表(ニ)
```

```
表(木)
```

```
表(ヘ)
```

```
表(ト)
```

```
表(チ)
```

```
表(リ)
```

```
表(ヌ)
```

```
表(清)
```

```
表(ル)
```

```
表(ヲ)
```

```
表(ワ)
```

```
表(力)
```

```
表(ヨ)
```

```
表(タ)
```

```
表(レ)
```

```
表(ソ)
```

```
表(ツ)
```

```
表(ネ)
```

```
表(ナ)
```

```
表(ラ)
```

```
表(ム)
```

```
表(ウ)
```

```
表(ヰ)
```

```
表(ノ)
```

```
表(オ)
```

```
表(ク)
```

```
表(ヤ)
```

```
表(マ)
```

```
表(ケ)
```

```
表(フ)
```

```
EOF
```

```
# Compile Compiler V0
```

```
python stage2_compiler.py compiler_v0.py1 > compiler_v0.py
```

(gen2.sha256)

5f2812d21c257574ab5a8cf38085c841c78e42955f7c10b473932ba8b05b9a43

\*compiler\_gen2.py

(0\_build.txt)

2026-01-11T07:35:38.6756904Z Current runner version: '2.330.0'  
2026-01-11T07:35:38.6790138Z ##[group]Runner Image Provisioner  
2026-01-11T07:35:38.6791298Z Hosted Compute Agent  
2026-01-11T07:35:38.6792075Z Version: 20251211.462  
2026-01-11T07:35:38.6792871Z Commit: 6cbad8c2bb55d58165063d031ccabf57e2d2db61  
2026-01-11T07:35:38.6793995Z Build Date: 2025-12-11T16:28:49Z  
2026-01-11T07:35:38.6794930Z Worker ID: {7f35ca63-b172-4182-b9d1-6ab51a6dc7ae}  
2026-01-11T07:35:38.6796039Z ##[endgroup]  
2026-01-11T07:35:38.6796810Z ##[group]Operating System  
2026-01-11T07:35:38.6797714Z Microsoft Windows Server 2025  
2026-01-11T07:35:38.6798570Z 10.0.26100  
2026-01-11T07:35:38.6799240Z Datacenter  
2026-01-11T07:35:38.6799955Z ##[endgroup]  
2026-01-11T07:35:38.6800680Z ##[group]Runner Image  
2026-01-11T07:35:38.6801561Z Image: windows-2025  
2026-01-11T07:35:38.6802334Z Version: 20260105.172.1  
2026-01-11T07:35:38.6805046Z Included Software: <https://github.com/actions/runner-images/blob/win25/20260105.172/images/windows/Windows2025-Readme.md>  
2026-01-11T07:35:38.6807683Z Image Release: <https://github.com/actions/runner-images/releases/tag/win25%2F20260105.172>  
2026-01-11T07:35:38.6809209Z ##[endgroup]  
2026-01-11T07:35:38.6810973Z ##[group]GITHUB\_TOKEN Permissions  
2026-01-11T07:35:38.6813407Z Contents: read  
2026-01-11T07:35:38.6814167Z Metadata: read  
2026-01-11T07:35:38.6815303Z Packages: read  
2026-01-11T07:35:38.6816020Z ##[endgroup]  
2026-01-11T07:35:38.6818999Z Secret source: Actions  
2026-01-11T07:35:38.6820031Z Prepare workflow directory  
2026-01-11T07:35:38.7378212Z Prepare all required actions  
2026-01-11T07:35:38.7428364Z Getting action download info  
2026-01-11T07:35:39.2054065Z Download action repository 'actions/checkout@v3'  
(SHA:f43a0e5ff2bd294095638e18286ca9a3d1956744)  
2026-01-11T07:35:39.3470123Z Download action repository 'actions/setup-python@v4'  
(SHA:7f4fc3e22c37d6ff65e88745f38bd3157c663f7c)  
2026-01-11T07:35:39.5108796Z Download action repository 'actions/upload-artifact@v4'  
(SHA:ea165f8d65b6e75b540449e92b4886f43607fa02)  
2026-01-11T07:35:39.7272974Z Download action repository 'ilammy/msvc-dev-cmd@v1'

(SHA:0b201ec74fa43914dc39ae48a89fd1d8cb592756)  
2026-01-11T07:35:40.4601779Z Complete job name: build  
2026-01-11T07:35:40.6821346Z ##[group]Run actions/checkout@v3  
2026-01-11T07:35:40.6822580Z with:  
2026-01-11T07:35:40.6823224Z repository: ryo11aori-ship-it/py1-1-5-14-40  
2026-01-11T07:35:40.6824309Z token: \*\*\*  
2026-01-11T07:35:40.6824878Z ssh-strict: true  
2026-01-11T07:35:40.6825502Z persist-credentials: true  
2026-01-11T07:35:40.6826187Z clean: true  
2026-01-11T07:35:40.6826793Z sparse-checkout-cone-mode: true  
2026-01-11T07:35:40.6827539Z fetch-depth: 1  
2026-01-11T07:35:40.6828123Z fetch-tags: false  
2026-01-11T07:35:40.6828722Z lfs: false  
2026-01-11T07:35:40.6829283Z submodules: false  
2026-01-11T07:35:40.6829978Z set-safe-directory: true  
2026-01-11T07:35:40.6831057Z env:  
2026-01-11T07:35:40.6831630Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:40.6832520Z PYTHONUTF8: 1  
2026-01-11T07:35:40.6833130Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:40.6833774Z ##[endgroup]  
2026-01-11T07:35:40.8371225Z Syncing repository: ryo11aori-ship-it/py1-1-5-14-40  
2026-01-11T07:35:40.8373531Z ##[group]Getting Git version info  
2026-01-11T07:35:40.8374490Z Working directory is 'D:\a\py1-1-5-14-40\py1-1-5-14-40'  
2026-01-11T07:35:40.9680305Z [command]"C:\Program Files\Git\bin\git.exe" version  
2026-01-11T07:35:41.7005208Z git version 2.52.0.windows.1  
2026-01-11T07:35:41.7059167Z ##[endgroup]  
2026-01-11T07:35:41.7082553Z Temporarily overriding HOME='D:  
\a\\_temp\8b9d34d2-7130-4d8a-868b-ceadf5387bfa' before making global git config changes  
2026-01-11T07:35:41.7084802Z Adding repository directory to the temporary git global config  
as a safe directory  
2026-01-11T07:35:41.7091158Z [command]"C:\Program Files\Git\bin\git.exe" config --global --  
add safe.directory D:\a\py1-1-5-14-40\py1-1-5-14-40  
2026-01-11T07:35:41.8005810Z Deleting the contents of 'D:\a\py1-1-5-14-40\py1-1-5-14-40'  
2026-01-11T07:35:41.8013215Z ##[group]Initializing the repository  
2026-01-11T07:35:41.8021241Z [command]"C:\Program Files\Git\bin\git.exe" init D:  
\a\py1-1-5-14-40\py1-1-5-14-40  
2026-01-11T07:35:41.9924886Z Initialized empty Git repository in D:/a/py1-1-5-14-40/  
py1-1-5-14-40/.git/  
2026-01-11T07:35:41.9968092Z [command]"C:\Program Files\Git\bin\git.exe" remote add origin  
<https://github.com/ryo11aori-ship-it/py1-1-5-14-40>  
2026-01-11T07:35:42.0732084Z ##[endgroup]

2026-01-11T07:35:42.0733270Z ##[group]Disabling automatic garbage collection  
2026-01-11T07:35:42.0738844Z [command]"C:\Program Files\Git\bin\git.exe" config --local gc.auto 0  
2026-01-11T07:35:42.1052843Z ##[endgroup]  
2026-01-11T07:35:42.1054097Z ##[group]Setting up auth  
2026-01-11T07:35:42.1062262Z [command]"C:\Program Files\Git\bin\git.exe" config --local --name-only --get-regexp core\.sshCommand  
2026-01-11T07:35:42.1359525Z [command]"C:\Program Files\Git\bin\git.exe" submodule foreach --recursive "sh -c \"git config --local --name-only --get-regexp 'core\.sshCommand' && git config --local --unset-all 'core.sshCommand' || :\""  
2026-01-11T07:35:44.5495027Z [command]"C:\Program Files\Git\bin\git.exe" config --local --name-only --get-regexp http\.https\:\/\/github\.com\/\.\extraheader  
2026-01-11T07:35:44.5808699Z [command]"C:\Program Files\Git\bin\git.exe" submodule foreach --recursive "sh -c \"git config --local --name-only --get-regexp 'http\.https\:\/\/github\.com\/\.\extraheader' && git config --local --unset-all 'http.https://github.com/.extraheader' || :\""  
2026-01-11T07:35:45.1198409Z [command]"C:\Program Files\Git\bin\git.exe" config --local http.https://github.com/.extraheader "AUTHORIZATION: basic \*\*\*"  
2026-01-11T07:35:45.1512723Z ##[endgroup]  
2026-01-11T07:35:45.1513607Z ##[group]Fetching the repository  
2026-01-11T07:35:45.1527355Z [command]"C:\Program Files\Git\bin\git.exe" -c protocol.version=2 fetch --no-tags --prune --progress --no-recurse-submodules --depth=1 origin +bd833a575b40756bdb4ce8802ad452d7490b980f:refs/remotes/origin/main  
2026-01-11T07:35:46.4356278Z remote: Enumerating objects: 29, done.  
2026-01-11T07:35:46.4356892Z remote: Counting objects: 3% (1/29)  
2026-01-11T07:35:46.4359867Z remote: Counting objects: 6% (2/29)  
2026-01-11T07:35:46.4360311Z remote: Counting objects: 10% (3/29)  
2026-01-11T07:35:46.4360616Z remote: Counting objects: 13% (4/29)  
2026-01-11T07:35:46.4360882Z remote: Counting objects: 17% (5/29)  
2026-01-11T07:35:46.4361135Z remote: Counting objects: 20% (6/29)  
2026-01-11T07:35:46.4361462Z remote: Counting objects: 24% (7/29)  
2026-01-11T07:35:46.4361740Z remote: Counting objects: 27% (8/29)  
2026-01-11T07:35:46.4362123Z remote: Counting objects: 31% (9/29)  
2026-01-11T07:35:46.4362389Z remote: Counting objects: 34% (10/29)  
2026-01-11T07:35:46.4362668Z remote: Counting objects: 37% (11/29)  
2026-01-11T07:35:46.4362940Z remote: Counting objects: 41% (12/29)  
2026-01-11T07:35:46.4363212Z remote: Counting objects: 44% (13/29)  
2026-01-11T07:35:46.4363544Z remote: Counting objects: 48% (14/29)  
2026-01-11T07:35:46.4432260Z remote: Counting objects: 51% (15/29)  
2026-01-11T07:35:46.4432941Z remote: Counting objects: 55% (16/29)  
2026-01-11T07:35:46.4433446Z remote: Counting objects: 58% (17/29)

2026-01-11T07:35:46.4434173Z remote: Counting objects: 62% (18/29)  
2026-01-11T07:35:46.4434579Z remote: Counting objects: 65% (19/29)  
2026-01-11T07:35:46.4434871Z remote: Counting objects: 68% (20/29)  
2026-01-11T07:35:46.4435137Z remote: Counting objects: 72% (21/29)  
2026-01-11T07:35:46.4435401Z remote: Counting objects: 75% (22/29)  
2026-01-11T07:35:46.4435788Z remote: Counting objects: 79% (23/29)  
2026-01-11T07:35:46.4436336Z remote: Counting objects: 82% (24/29)  
2026-01-11T07:35:46.4436757Z remote: Counting objects: 86% (25/29)  
2026-01-11T07:35:46.4437204Z remote: Counting objects: 89% (26/29)  
2026-01-11T07:35:46.4438732Z remote: Counting objects: 93% (27/29)  
2026-01-11T07:35:46.4439193Z remote: Counting objects: 96% (28/29)  
2026-01-11T07:35:46.4439638Z remote: Counting objects: 100% (29/29)  
2026-01-11T07:35:46.4440118Z remote: Counting objects: 100% (29/29), done.  
2026-01-11T07:35:46.4440614Z remote: Compressing objects: 3% (1/26)  
2026-01-11T07:35:46.4441413Z remote: Compressing objects: 7% (2/26)  
2026-01-11T07:35:46.4441968Z remote: Compressing objects: 11% (3/26)  
2026-01-11T07:35:46.4442446Z remote: Compressing objects: 15% (4/26)  
2026-01-11T07:35:46.4442727Z remote: Compressing objects: 19% (5/26)  
2026-01-11T07:35:46.4443012Z remote: Compressing objects: 23% (6/26)  
2026-01-11T07:35:46.4443282Z remote: Compressing objects: 26% (7/26)  
2026-01-11T07:35:46.4443549Z remote: Compressing objects: 30% (8/26)  
2026-01-11T07:35:46.4443815Z remote: Compressing objects: 34% (9/26)  
2026-01-11T07:35:46.4444300Z remote: Compressing objects: 38% (10/26)  
2026-01-11T07:35:46.4444792Z remote: Compressing objects: 42% (11/26)  
2026-01-11T07:35:46.4445309Z remote: Compressing objects: 46% (12/26)  
2026-01-11T07:35:46.4445690Z remote: Compressing objects: 50% (13/26)  
2026-01-11T07:35:46.4445987Z remote: Compressing objects: 53% (14/26)  
2026-01-11T07:35:46.4446297Z remote: Compressing objects: 57% (15/26)  
2026-01-11T07:35:46.4446638Z remote: Compressing objects: 61% (16/26)  
2026-01-11T07:35:46.4446964Z remote: Compressing objects: 65% (17/26)  
2026-01-11T07:35:46.4447274Z remote: Compressing objects: 69% (18/26)  
2026-01-11T07:35:46.4447546Z remote: Compressing objects: 73% (19/26)  
2026-01-11T07:35:46.4447825Z remote: Compressing objects: 76% (20/26)  
2026-01-11T07:35:46.4448110Z remote: Compressing objects: 80% (21/26)  
2026-01-11T07:35:46.4448511Z remote: Compressing objects: 84% (22/26)  
2026-01-11T07:35:46.4449015Z remote: Compressing objects: 88% (23/26)  
2026-01-11T07:35:46.4449483Z remote: Compressing objects: 92% (24/26)  
2026-01-11T07:35:46.4450604Z remote: Compressing objects: 96% (25/26)  
2026-01-11T07:35:46.4450980Z remote: Compressing objects: 100% (26/26)  
2026-01-11T07:35:46.4451300Z remote: Compressing objects: 100% (26/26), done.  
2026-01-11T07:35:46.5321713Z remote: Total 29 (delta 3), reused 16 (delta 1), pack-reused 0

(from 0)

2026-01-11T07:35:46.7007830Z From <https://github.com/ryo11aori-ship-it/py1-1-5-14-40>  
2026-01-11T07:35:46.7008655Z \* [new ref]  
bd833a575b40756bdb4ce8802ad452d7490b980f -> origin/main  
2026-01-11T07:35:46.7527050Z ##[endgroup]  
2026-01-11T07:35:46.7527735Z ##[group]Determining the checkout info  
2026-01-11T07:35:46.7529534Z ##[endgroup]  
2026-01-11T07:35:46.7530208Z ##[group]Checking out the ref  
2026-01-11T07:35:46.7540523Z [command]"C:\Program Files\Git\bin\git.exe" checkout --progress --force -B main refs/remotes/origin/main  
2026-01-11T07:35:46.8623322Z branch 'main' set up to track 'origin/main'.  
2026-01-11T07:35:46.8630640Z Switched to a new branch 'main'  
2026-01-11T07:35:46.8668848Z ##[endgroup]  
2026-01-11T07:35:46.9145688Z [command]"C:\Program Files\Git\bin\git.exe" log -1 --format='%H'  
2026-01-11T07:35:46.9423619Z 'bd833a575b40756bdb4ce8802ad452d7490b980f'  
2026-01-11T07:35:46.9855926Z ##[group]Run actions/setup-python@v4  
2026-01-11T07:35:46.9856240Z with:  
2026-01-11T07:35:46.9856412Z python-version: 3.10  
2026-01-11T07:35:46.9856606Z check-latest: false  
2026-01-11T07:35:46.9856933Z token: \*\*\*  
2026-01-11T07:35:46.9857108Z update-environment: true  
2026-01-11T07:35:46.9857313Z allow-prereleases: false  
2026-01-11T07:35:46.9857513Z env:  
2026-01-11T07:35:46.9857661Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:46.9857859Z PYTHONUTF8: 1  
2026-01-11T07:35:46.9858017Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:46.9858189Z ##[endgroup]  
2026-01-11T07:35:47.2136917Z ##[group]Installed versions  
2026-01-11T07:35:47.2487599Z Successfully set up CPython (3.10.11)  
2026-01-11T07:35:47.2489226Z ##[endgroup]  
2026-01-11T07:35:47.2799109Z ##[group]Run pip install black  
2026-01-11T07:35:47.2799476Z [36;1mpip install black[0m  
2026-01-11T07:35:47.2857866Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}  
2026-01-11T07:35:47.2858258Z env:  
2026-01-11T07:35:47.2858429Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:47.2858631Z PYTHONUTF8: 1  
2026-01-11T07:35:47.2858802Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:47.2859064Z pythonLocation: C:  
|hostedtoolcache\windows\Python\3.10.11\x64

2026-01-11T07:35:47.2859489Z PKG\_CONFIG\_PATH: C:  
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig  
2026-01-11T07:35:47.2859905Z Python\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:47.2860273Z Python2\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:47.2860655Z Python3\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:47.2860955Z ##[endgroup]  
2026-01-11T07:35:52.8629380Z Collecting black  
2026-01-11T07:35:52.8973289Z Downloading black-25.12.0-cp310-cp310-win\_amd64.whl.metadata (86 kB)  
2026-01-11T07:35:52.9709374Z Collecting click>=8.0.0 (from black)  
2026-01-11T07:35:52.9754258Z Downloading click-8.3.1-py3-none-any.whl.metadata (2.6 kB)  
2026-01-11T07:35:53.0113886Z Collecting mypy\_extensions>=0.4.3 (from black)  
2026-01-11T07:35:53.0157337Z Downloading mypy\_extensions-1.1.0-py3-none-any.whl.metadata (1.1 kB)  
2026-01-11T07:35:53.0567785Z Collecting packaging>=22.0 (from black)  
2026-01-11T07:35:53.0619665Z Downloading packaging-25.0-py3-none-any.whl.metadata (3.3 kB)  
2026-01-11T07:35:53.1016738Z Collecting pathspec>=0.9.0 (from black)  
2026-01-11T07:35:53.1080850Z Downloading pathspec-1.0.3-py3-none-any.whl.metadata (13 kB)  
2026-01-11T07:35:53.1572998Z Collecting platformdirs>=2 (from black)  
2026-01-11T07:35:53.1619189Z Downloading platformdirs-4.5.1-py3-none-any.whl.metadata (12 kB)  
2026-01-11T07:35:53.2037449Z Collecting pytokens>=0.3.0 (from black)  
2026-01-11T07:35:53.2078917Z Downloading pytokens-0.3.0-py3-none-any.whl.metadata (2.0 kB)  
2026-01-11T07:35:53.2509495Z Collecting tomli>=1.1.0 (from black)  
2026-01-11T07:35:53.2552605Z Downloading tomli-2.3.0-py3-none-any.whl.metadata (10 kB)  
2026-01-11T07:35:53.3000463Z Collecting typing\_extensions>=4.0.1 (from black)  
2026-01-11T07:35:53.3052691Z Downloading typing\_extensions-4.15.0-py3-none-any.whl.metadata (3.3 kB)  
2026-01-11T07:35:53.3507646Z Collecting colorama (from click>=8.0.0->black)  
2026-01-11T07:35:53.3549639Z Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)  
2026-01-11T07:35:53.3825184Z Downloading black-25.12.0-cp310-cp310-win\_amd64.whl (1.4 MB)  
2026-01-11T07:35:53.4238712Z ----- 1.4/1.4 MB 24.6 MB/s  
0:00:00

2026-01-11T07:35:53.4279436Z Downloading click-8.3.1-py3-none-any.whl (108 kB)  
2026-01-11T07:35:53.4471429Z Downloading mypy\_extensions-1.1.0-py3-none-any.whl (5.0 kB)  
2026-01-11T07:35:53.4651430Z Downloading packaging-25.0-py3-none-any.whl (66 kB)  
2026-01-11T07:35:53.4878002Z Downloading pathspec-1.0.3-py3-none-any.whl (55 kB)  
2026-01-11T07:35:53.5102787Z Downloading platformdirs-4.5.1-py3-none-any.whl (18 kB)  
2026-01-11T07:35:53.5282987Z Downloading pytokens-0.3.0-py3-none-any.whl (12 kB)  
2026-01-11T07:35:53.5501910Z Downloading tomli-2.3.0-py3-none-any.whl (14 kB)  
2026-01-11T07:35:53.5691828Z Downloading typing\_extensions-4.15.0-py3-none-any.whl (44 kB)  
2026-01-11T07:35:53.5872311Z Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)  
2026-01-11T07:35:53.6878492Z Installing collected packages: typing-extensions, tomli, pytokens, platformdirs, pathspec, packaging, mypy-extensions, colorama, click, black  
2026-01-11T07:35:54.6227055Z  
2026-01-11T07:35:54.6291200Z Successfully installed black-25.12.0 click-8.3.1 colorama-0.4.6 mypy-extensions-1.1.0 packaging-25.0 pathspec-1.0.3 platformdirs-4.5.1 pytokens-0.3.0 tomli-2.3.0 typing-extensions-4.15.0  
2026-01-11T07:35:54.7833068Z ##[group]Run dos2unix compiler.py1  
2026-01-11T07:35:54.7833421Z [36;1mdos2unix compiler.py1[0m  
2026-01-11T07:35:54.7833727Z [36;1mpython py1.py compiler.py1 > compiler\_gen2.py[0m  
2026-01-11T07:35:54.7834029Z [36;1mdos2unix compiler\_gen2.py[0m  
2026-01-11T07:35:54.7834251Z [36;1mblack compiler\_gen2.py[0m  
2026-01-11T07:35:54.7834476Z [36;1mdos2unix compiler\_gen2.py[0m  
2026-01-11T07:35:54.7851398Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}  
2026-01-11T07:35:54.7851767Z env:  
2026-01-11T07:35:54.7851938Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:54.7852139Z PYTHONUTF8: 1  
2026-01-11T07:35:54.7852304Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:54.7852585Z pythonLocation: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:54.7853014Z PKG\_CONFIG\_PATH: C:  
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig  
2026-01-11T07:35:54.7853438Z Python\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:54.7853855Z Python2\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:54.7854223Z Python3\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:54.7854510Z ##[endgroup]  
2026-01-11T07:35:54.8506933Z dos2unix: converting file compiler.py1 to Unix format...  
2026-01-11T07:35:54.9450808Z dos2unix: converting file compiler\_gen2.py to Unix format...

2026-01-11T07:35:55.4378466Z reformatted compiler\_gen2.py  
2026-01-11T07:35:55.4379059Z  
2026-01-11T07:35:55.4379714Z All done! ✨ 🎂 ✨  
2026-01-11T07:35:55.4380235Z 1 file reformatted.  
2026-01-11T07:35:55.4792780Z dos2unix: converting file compiler\_gen2.py to Unix format...  
2026-01-11T07:35:55.4998847Z ##[group]Run python compiler\_gen2.py compiler.py1 > compiler\_gen3.py  
2026-01-11T07:35:55.4999405Z [36;1mpython compiler\_gen2.py compiler.py1 > compiler\_gen3.py[0m  
2026-01-11T07:35:55.4999749Z [36;1mdos2unix compiler\_gen3.py[0m  
2026-01-11T07:35:55.4999983Z [36;1mblack compiler\_gen3.py[0m  
2026-01-11T07:35:55.5000211Z [36;1mdos2unix compiler\_gen3.py[0m  
2026-01-11T07:35:55.5017386Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}  
2026-01-11T07:35:55.5017734Z env:  
2026-01-11T07:35:55.5017902Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:55.5018092Z PYTHONUTF8: 1  
2026-01-11T07:35:55.5018261Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:55.5018523Z pythonLocation: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:55.5018934Z PKG\_CONFIG\_PATH: C:  
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig  
2026-01-11T07:35:55.5019343Z Python\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:55.5019743Z Python2\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:55.5020106Z Python3\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:55.5020396Z ##[endgroup]  
2026-01-11T07:35:55.6205395Z dos2unix: converting file compiler\_gen3.py to Unix format...  
2026-01-11T07:35:55.9719132Z reformatted compiler\_gen3.py  
2026-01-11T07:35:55.9719461Z  
2026-01-11T07:35:55.9721015Z All done! ✨ 🎂 ✨  
2026-01-11T07:35:55.9721307Z 1 file reformatted.  
2026-01-11T07:35:56.0132928Z dos2unix: converting file compiler\_gen3.py to Unix format...  
2026-01-11T07:35:56.0306522Z ##[group]Run sha256sum compiler\_gen2.py > gen2.sha256  
2026-01-11T07:35:56.0306960Z [36;1msha256sum compiler\_gen2.py > gen2.sha256[0m  
2026-01-11T07:35:56.0307270Z [36;1msha256sum compiler\_gen3.py > gen3.sha256[0m  
2026-01-11T07:35:56.0307642Z [36;1mif [ "\$(cut -d '' -f 1 gen2.sha256)" = "\$(cut -d '' -f 1 gen3.sha256)" ]; then[0m  
2026-01-11T07:35:56.0308071Z [36;1m echo "SUCCESS: Byte-level reproducibility

achieved."[0m  
2026-01-11T07:35:56.0308358Z [36;1melse[0m  
2026-01-11T07:35:56.0308551Z [36;1m echo "FAILURE: Checksums do not match."[0m  
2026-01-11T07:35:56.0308796Z [36;1m exit 1[0m  
2026-01-11T07:35:56.0308939Z [36;1mfif[0m  
2026-01-11T07:35:56.0325390Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e  
-o pipefail {0}  
2026-01-11T07:35:56.0326447Z env:  
2026-01-11T07:35:56.0326617Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:56.0326807Z PYTHONUTF8: 1  
2026-01-11T07:35:56.0326972Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:56.0327271Z pythonLocation: C:  
\hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.0327694Z PKG\_CONFIG\_PATH: C:  
\hostedtoolcache\windows\Python\3.10.11\x64/lib/pkgconfig  
2026-01-11T07:35:56.0328118Z Python\_ROOT\_DIR: C:  
\hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.0328502Z Python2\_ROOT\_DIR: C:  
\hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.0328864Z Python3\_ROOT\_DIR: C:  
\hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.0329151Z ##[endgroup]  
2026-01-11T07:35:56.1615843Z SUCCESS: Byte-level reproducibility achieved.  
2026-01-11T07:35:56.1812656Z ##[group]Run cat <<EOF > fizzbuzz\_while.py1  
2026-01-11T07:35:56.1813048Z [36;1mcat <<EOF > fizzbuzz\_while.py1[0m  
2026-01-11T07:35:56.1813300Z [36;1m# Strict FizzBuzz Logic[0m  
2026-01-11T07:35:56.1813532Z [36;1m@v 表 'print'[0m  
2026-01-11T07:35:56.1813772Z [36;1m@v 字 'str'[0m  
2026-01-11T07:35:56.1830441Z [36;1m@v 循 'while'[0m  
2026-01-11T07:35:56.1830760Z [36;1m@v 也 'if'[0m  
2026-01-11T07:35:56.1831059Z [36;1m@v 或 'elif'[0m  
2026-01-11T07:35:56.1831344Z [36;1m@v 他 'else'[0m  
2026-01-11T07:35:56.1831631Z [36;1m@v 剩 '%'[0m  
2026-01-11T07:35:56.1831894Z [36;1m@v 等 '=='[0m  
2026-01-11T07:35:56.1832173Z [36;1m@v 足 '+'[0m  
2026-01-11T07:35:56.1832433Z [36;1m@v 小 '<'[0m  
2026-01-11T07:35:56.1832696Z [36;1m@v 壱 '1'[0m  
2026-01-11T07:35:56.1832961Z [36;1m@v 佰 '101'[0m  
2026-01-11T07:35:56.1833249Z [36;1m@v 零 '0'[0m  
2026-01-11T07:35:56.1833507Z [36;1m@v 三 '3'[0m  
2026-01-11T07:35:56.1833761Z [36;1m@v 五 '5'[0m

2026-01-11T07:35:56.1834069Z [36;1m@v 拾 '15'[0m  
2026-01-11T07:35:56.1834342Z [36;1m@v 泡 ""Fizz""[0m  
2026-01-11T07:35:56.1834646Z [36;1m@v 響 ""Buzz""[0m  
2026-01-11T07:35:56.1834925Z [36;1m@v 数 'i'[0m  
2026-01-11T07:35:56.1835192Z [36;1m@v ド '\$'[0m  
2026-01-11T07:35:56.1835478Z [36;1m\$[0m  
2026-01-11T07:35:56.1835763Z [36;1m数 = 壱[0m  
2026-01-11T07:35:56.1836025Z [36;1m循 数 小 佰:[0m  
2026-01-11T07:35:56.1836287Z [36;1m も 数 剩 拾 等 零:[0m  
2026-01-11T07:35:56.1836584Z [36;1m 表(泡 足 響)[0m  
2026-01-11T07:35:56.1836859Z [36;1m 或 数 剩 三 等 零:[0m  
2026-01-11T07:35:56.1837152Z [36;1m 表(泡)[0m  
2026-01-11T07:35:56.1837406Z [36;1m 或 数 剩 五 等 零:[0m  
2026-01-11T07:35:56.1837699Z [36;1m 表(響)[0m  
2026-01-11T07:35:56.1837951Z [36;1m 他:[0m  
2026-01-11T07:35:56.1838207Z [36;1m 表(数)[0m  
2026-01-11T07:35:56.1838461Z [36;1m 数 = 数 足 壱[0m  
2026-01-11T07:35:56.1838743Z [36;1mEOF[0m  
2026-01-11T07:35:56.1866847Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e  
-o pipefail {0}  
2026-01-11T07:35:56.1867445Z env:  
2026-01-11T07:35:56.1867742Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:56.1868067Z PYTHONUTF8: 1  
2026-01-11T07:35:56.1868349Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:56.1868793Z pythonLocation: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.1869531Z PKG\_CONFIG\_PATH: C:  
|hostedtoolcache\windows\Python\3.10.11\x64/lib/pkgconfig  
2026-01-11T07:35:56.1870285Z Python\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.1871308Z Python2\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.1871983Z Python3\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.1872488Z ##[endgroup]  
2026-01-11T07:35:56.2705616Z ##[group]Run python compiler\_gen3.py fizzbuzz.py1 >  
output\_fb.py  
2026-01-11T07:35:56.2706147Z [36;1mpython compiler\_gen3.py fizzbuzz.py1 >  
output\_fb.py[0m  
2026-01-11T07:35:56.2706450Z [36;1mpython output\_fb.py[0m  
2026-01-11T07:35:56.2727664Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e

```
-o pipefail {0}
2026-01-11T07:35:56.2728027Z env:
2026-01-11T07:35:56.2728189Z  PYTHONIOENCODING: utf-8
2026-01-11T07:35:56.2728379Z  PYTHONUTF8: 1
2026-01-11T07:35:56.2728544Z  PYTHONUNBUFFERED: 1
2026-01-11T07:35:56.2728801Z  pythonLocation: C:
\hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:56.2729387Z  PKG_CONFIG_PATH: C:
\hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig
2026-01-11T07:35:56.2729804Z  Python_ROOT_DIR: C:
\hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:56.2730170Z  Python2_ROOT_DIR: C:
\hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:56.2730582Z  Python3_ROOT_DIR: C:
\hostedtoolcache\windows\Python\3.10.11\x64
2026-01-11T07:35:56.2730870Z ##[endgroup]
2026-01-11T07:35:56.3992190Z 1
2026-01-11T07:35:56.3992579Z 2
2026-01-11T07:35:56.3992853Z Fizz
2026-01-11T07:35:56.3993135Z 4
2026-01-11T07:35:56.3993633Z Buzz
2026-01-11T07:35:56.3994225Z Fizz
2026-01-11T07:35:56.3994481Z 7
2026-01-11T07:35:56.3994691Z 8
2026-01-11T07:35:56.3994908Z Fizz
2026-01-11T07:35:56.3995071Z Buzz
2026-01-11T07:35:56.3995282Z 11
2026-01-11T07:35:56.3995492Z Fizz
2026-01-11T07:35:56.3995748Z 13
2026-01-11T07:35:56.3995958Z 14
2026-01-11T07:35:56.3996198Z FizzBuzz
2026-01-11T07:35:56.3996460Z 16
2026-01-11T07:35:56.3996651Z 17
2026-01-11T07:35:56.3996783Z Fizz
2026-01-11T07:35:56.3997296Z 19
2026-01-11T07:35:56.3997480Z Buzz
2026-01-11T07:35:56.3997705Z Fizz
2026-01-11T07:35:56.3997914Z 22
2026-01-11T07:35:56.3998046Z 23
2026-01-11T07:35:56.3998285Z Fizz
2026-01-11T07:35:56.3998850Z Buzz
```

2026-01-11T07:35:56.3999101Z 26  
2026-01-11T07:35:56.3999271Z Fizz  
2026-01-11T07:35:56.3999436Z 28  
2026-01-11T07:35:56.3999583Z 29  
2026-01-11T07:35:56.3999728Z FizzBuzz  
2026-01-11T07:35:56.3999877Z 31  
2026-01-11T07:35:56.4000276Z 32  
2026-01-11T07:35:56.4000421Z Fizz  
2026-01-11T07:35:56.4000550Z 34  
2026-01-11T07:35:56.4000680Z Buzz  
2026-01-11T07:35:56.4000807Z Fizz  
2026-01-11T07:35:56.4000938Z 37  
2026-01-11T07:35:56.4001062Z 38  
2026-01-11T07:35:56.4001191Z Fizz  
2026-01-11T07:35:56.4001315Z Buzz  
2026-01-11T07:35:56.4001445Z 41  
2026-01-11T07:35:56.4001568Z Fizz  
2026-01-11T07:35:56.4001701Z 43  
2026-01-11T07:35:56.4001825Z 44  
2026-01-11T07:35:56.4001957Z FizzBuzz  
2026-01-11T07:35:56.4002101Z 46  
2026-01-11T07:35:56.4002237Z 47  
2026-01-11T07:35:56.4002362Z Fizz  
2026-01-11T07:35:56.4002495Z 49  
2026-01-11T07:35:56.4002626Z Buzz  
2026-01-11T07:35:56.4002751Z Fizz  
2026-01-11T07:35:56.4002884Z 52  
2026-01-11T07:35:56.4003008Z 53  
2026-01-11T07:35:56.4003139Z Fizz  
2026-01-11T07:35:56.4003264Z Buzz  
2026-01-11T07:35:56.4003397Z 56  
2026-01-11T07:35:56.4003523Z Fizz  
2026-01-11T07:35:56.4003676Z 58  
2026-01-11T07:35:56.4003832Z 59  
2026-01-11T07:35:56.4003964Z FizzBuzz  
2026-01-11T07:35:56.4004099Z 61  
2026-01-11T07:35:56.4004231Z 62  
2026-01-11T07:35:56.4004355Z Fizz  
2026-01-11T07:35:56.4004487Z 64  
2026-01-11T07:35:56.4004613Z Buzz  
2026-01-11T07:35:56.4004759Z Fizz

2026-01-11T07:35:56.4004889Z 67  
2026-01-11T07:35:56.4005016Z 68  
2026-01-11T07:35:56.4005151Z Fizz  
2026-01-11T07:35:56.4005279Z Buzz  
2026-01-11T07:35:56.4005417Z 71  
2026-01-11T07:35:56.4005547Z Fizz  
2026-01-11T07:35:56.4005680Z 73  
2026-01-11T07:35:56.4005812Z 74  
2026-01-11T07:35:56.4005938Z FizzBuzz  
2026-01-11T07:35:56.4006087Z 76  
2026-01-11T07:35:56.4006212Z 77  
2026-01-11T07:35:56.4006346Z Fizz  
2026-01-11T07:35:56.4006473Z 79  
2026-01-11T07:35:56.4006604Z Buzz  
2026-01-11T07:35:56.4006744Z Fizz  
2026-01-11T07:35:56.4006880Z 82  
2026-01-11T07:35:56.4007001Z 83  
2026-01-11T07:35:56.4007132Z Fizz  
2026-01-11T07:35:56.4007256Z Buzz  
2026-01-11T07:35:56.4007386Z 86  
2026-01-11T07:35:56.4007514Z Fizz  
2026-01-11T07:35:56.4007639Z 88  
2026-01-11T07:35:56.4010029Z 89  
2026-01-11T07:35:56.4010232Z FizzBuzz  
2026-01-11T07:35:56.4010377Z 91  
2026-01-11T07:35:56.4010498Z 92  
2026-01-11T07:35:56.4010621Z Fizz  
2026-01-11T07:35:56.4010746Z 94  
2026-01-11T07:35:56.4010868Z Buzz  
2026-01-11T07:35:56.4010985Z Fizz  
2026-01-11T07:35:56.4011110Z 97  
2026-01-11T07:35:56.4011225Z 98  
2026-01-11T07:35:56.4011349Z Fizz  
2026-01-11T07:35:56.4011467Z Buzz  
2026-01-11T07:35:56.4208573Z ##[group]Run python compiler\_gen3.py unicode\_test.py1 > output\_uni.py  
2026-01-11T07:35:56.4209108Z [36;1mpython compiler\_gen3.py unicode\_test.py1 > output\_uni.py[0m  
2026-01-11T07:35:56.4209638Z [36;1mpython output\_uni.py[0m  
2026-01-11T07:35:56.4226178Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}

2026-01-11T07:35:56.4226540Z env:  
2026-01-11T07:35:56.4226704Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:56.4226896Z PYTHONUTF8: 1  
2026-01-11T07:35:56.4227065Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:56.4227324Z pythonLocation: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.4227743Z PKG\_CONFIG\_PATH: C:  
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig  
2026-01-11T07:35:56.4228164Z Python\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.4228529Z Python2\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.4228922Z Python3\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.4229205Z ##[endgroup]  
2026-01-11T07:35:56.5466450Z 1.7724538509055159  
2026-01-11T07:35:56.5466735Z 45  
2026-01-11T07:35:56.5467139Z 日本語もOK  
2026-01-11T07:35:56.5678833Z ##[group]Run python compiler\_gen3.py compiler\_ir.py1 > compiler\_ir.py  
2026-01-11T07:35:56.5679435Z [36;1mpython compiler\_gen3.py compiler\_ir.py1 > compiler\_ir.py[0m  
2026-01-11T07:35:56.5680020Z [36;1mpython compiler\_ir.py fizzbuzz\_while.py1 > [fizzbuzz.ir](#)[0m  
2026-01-11T07:35:56.5680352Z [36;1mcat [fizzbuzz.ir](#)[0m  
2026-01-11T07:35:56.5695940Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}  
2026-01-11T07:35:56.5696371Z env:  
2026-01-11T07:35:56.5696533Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:56.5696741Z PYTHONUTF8: 1  
2026-01-11T07:35:56.5696903Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:56.5697162Z pythonLocation: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.5697594Z PKG\_CONFIG\_PATH: C:  
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig  
2026-01-11T07:35:56.5698321Z Python\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.5698739Z Python2\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.5699116Z Python3\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64

2026-01-11T07:35:56.5699408Z ##[endgroup]  
2026-01-11T07:35:56.7449339Z PUSH 1  
2026-01-11T07:35:56.7449620Z STORE i  
2026-01-11T07:35:56.7450188Z LABEL L1  
2026-01-11T07:35:56.7450443Z LOAD i  
2026-01-11T07:35:56.7450691Z PUSH 101  
2026-01-11T07:35:56.7450948Z LT  
2026-01-11T07:35:56.7451171Z JZERO L2  
2026-01-11T07:35:56.7451402Z LOAD i  
2026-01-11T07:35:56.7451625Z PUSH 15  
2026-01-11T07:35:56.7451855Z MOD  
2026-01-11T07:35:56.7452068Z PUSH 0  
2026-01-11T07:35:56.7452294Z EQ  
2026-01-11T07:35:56.7452506Z JZERO L3  
2026-01-11T07:35:56.7452730Z PUSH Fizz  
2026-01-11T07:35:56.7452956Z PUSH Buzz  
2026-01-11T07:35:56.7453175Z ADD  
2026-01-11T07:35:56.7453384Z PRINT  
2026-01-11T07:35:56.7453612Z JUMP L4  
2026-01-11T07:35:56.7453839Z LABEL L3  
2026-01-11T07:35:56.7454058Z LOAD i  
2026-01-11T07:35:56.7454270Z PUSH 3  
2026-01-11T07:35:56.7454477Z MOD  
2026-01-11T07:35:56.7454686Z PUSH 0  
2026-01-11T07:35:56.7454941Z EQ  
2026-01-11T07:35:56.7455158Z JZERO L5  
2026-01-11T07:35:56.7455377Z PUSH Fizz  
2026-01-11T07:35:56.7455613Z PRINT  
2026-01-11T07:35:56.7456422Z JUMP L6  
2026-01-11T07:35:56.7456660Z LABEL L5  
2026-01-11T07:35:56.7456928Z LOAD i  
2026-01-11T07:35:56.7457191Z PUSH 5  
2026-01-11T07:35:56.7457408Z MOD  
2026-01-11T07:35:56.7457619Z PUSH 0  
2026-01-11T07:35:56.7457822Z EQ  
2026-01-11T07:35:56.7458038Z JZERO L7  
2026-01-11T07:35:56.7458276Z PUSH Buzz  
2026-01-11T07:35:56.7458509Z PRINT  
2026-01-11T07:35:56.7458736Z JUMP L8  
2026-01-11T07:35:56.7458958Z LABEL L7  
2026-01-11T07:35:56.7459183Z LOAD i

2026-01-11T07:35:56.7459397Z PRINT  
2026-01-11T07:35:56.7459609Z LABEL L8  
2026-01-11T07:35:56.7459820Z LABEL L6  
2026-01-11T07:35:56.7460037Z LABEL L4  
2026-01-11T07:35:56.7460254Z LOAD i  
2026-01-11T07:35:56.7460820Z PUSH 1  
2026-01-11T07:35:56.7461061Z ADD  
2026-01-11T07:35:56.7461489Z STORE i  
2026-01-11T07:35:56.7461751Z JUMP L1  
2026-01-11T07:35:56.7462172Z LABEL L2  
2026-01-11T07:35:56.7627820Z ##[group]Run gcc -o vm.exe vm.c  
2026-01-11T07:35:56.7628344Z [36;1mgcc -o vm.exe vm.c[0m  
2026-01-11T07:35:56.7628572Z [36;1m./vm.exe fizzbuzz.ir[0m  
2026-01-11T07:35:56.7645813Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}  
2026-01-11T07:35:56.7646189Z env:  
2026-01-11T07:35:56.7646349Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:56.7646554Z PYTHONUTF8: 1  
2026-01-11T07:35:56.7646715Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:56.7646978Z pythonLocation: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.7647414Z PKG\_CONFIG\_PATH: C:  
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig  
2026-01-11T07:35:56.7647849Z Python\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.7648228Z Python2\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.7648594Z Python3\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:56.7648890Z ##[endgroup]  
2026-01-11T07:35:59.6244055Z 1  
2026-01-11T07:35:59.6244387Z 2  
2026-01-11T07:35:59.6244916Z Fizz  
2026-01-11T07:35:59.6245131Z 4  
2026-01-11T07:35:59.6245319Z Buzz  
2026-01-11T07:35:59.6245531Z Fizz  
2026-01-11T07:35:59.6245722Z 7  
2026-01-11T07:35:59.6245928Z 8  
2026-01-11T07:35:59.6246137Z Fizz  
2026-01-11T07:35:59.6246551Z Buzz  
2026-01-11T07:35:59.6246782Z 11

2026-01-11T07:35:59.6246936Z Fizz  
2026-01-11T07:35:59.6247062Z 13  
2026-01-11T07:35:59.6247189Z 14  
2026-01-11T07:35:59.6247311Z FizzBuzz  
2026-01-11T07:35:59.6247454Z 16  
2026-01-11T07:35:59.6247585Z 17  
2026-01-11T07:35:59.6247701Z Fizz  
2026-01-11T07:35:59.6247829Z 19  
2026-01-11T07:35:59.6247948Z Buzz  
2026-01-11T07:35:59.6248072Z Fizz  
2026-01-11T07:35:59.6248190Z 22  
2026-01-11T07:35:59.6248310Z 23  
2026-01-11T07:35:59.6248427Z Fizz  
2026-01-11T07:35:59.6248548Z Buzz  
2026-01-11T07:35:59.6248663Z 26  
2026-01-11T07:35:59.6248785Z Fizz  
2026-01-11T07:35:59.6248901Z 28  
2026-01-11T07:35:59.6249020Z 29  
2026-01-11T07:35:59.6249137Z FizzBuzz  
2026-01-11T07:35:59.6249285Z 31  
2026-01-11T07:35:59.6249407Z 32  
2026-01-11T07:35:59.6249530Z Fizz  
2026-01-11T07:35:59.6249650Z 34  
2026-01-11T07:35:59.6249768Z Buzz  
2026-01-11T07:35:59.6249891Z Fizz  
2026-01-11T07:35:59.6250008Z 37  
2026-01-11T07:35:59.6250129Z 38  
2026-01-11T07:35:59.6250243Z Fizz  
2026-01-11T07:35:59.6250364Z Buzz  
2026-01-11T07:35:59.6250481Z 41  
2026-01-11T07:35:59.6250602Z Fizz  
2026-01-11T07:35:59.6250717Z 43  
2026-01-11T07:35:59.6250837Z 44  
2026-01-11T07:35:59.6250956Z FizzBuzz  
2026-01-11T07:35:59.6251089Z 46  
2026-01-11T07:35:59.6251205Z 47  
2026-01-11T07:35:59.6251328Z Fizz  
2026-01-11T07:35:59.6251445Z 49  
2026-01-11T07:35:59.6251564Z Buzz  
2026-01-11T07:35:59.6251684Z Fizz  
2026-01-11T07:35:59.6251804Z 52

2026-01-11T07:35:59.6251923Z 53  
2026-01-11T07:35:59.6252037Z Fizz  
2026-01-11T07:35:59.6252157Z Buzz  
2026-01-11T07:35:59.6252273Z 56  
2026-01-11T07:35:59.6252397Z Fizz  
2026-01-11T07:35:59.6252517Z 58  
2026-01-11T07:35:59.6252640Z 59  
2026-01-11T07:35:59.6252762Z FizzBuzz  
2026-01-11T07:35:59.6253035Z 61  
2026-01-11T07:35:59.6253254Z 62  
2026-01-11T07:35:59.6253465Z Fizz  
2026-01-11T07:35:59.6254252Z 64  
2026-01-11T07:35:59.6254765Z Buzz  
2026-01-11T07:35:59.6254907Z Fizz  
2026-01-11T07:35:59.6255039Z 67  
2026-01-11T07:35:59.6255162Z 68  
2026-01-11T07:35:59.6255290Z Fizz  
2026-01-11T07:35:59.6255419Z Buzz  
2026-01-11T07:35:59.6255590Z 71  
2026-01-11T07:35:59.6255806Z Fizz  
2026-01-11T07:35:59.6255943Z 73  
2026-01-11T07:35:59.6256182Z 74  
2026-01-11T07:35:59.6256322Z FizzBuzz  
2026-01-11T07:35:59.6256955Z 76  
2026-01-11T07:35:59.6257214Z 77  
2026-01-11T07:35:59.6257407Z Fizz  
2026-01-11T07:35:59.6257612Z 79  
2026-01-11T07:35:59.6258512Z Buzz  
2026-01-11T07:35:59.6258990Z Fizz  
2026-01-11T07:35:59.6259804Z 82  
2026-01-11T07:35:59.6260020Z 83  
2026-01-11T07:35:59.6260222Z Fizz  
2026-01-11T07:35:59.6260427Z Buzz  
2026-01-11T07:35:59.6260645Z 86  
2026-01-11T07:35:59.6261482Z Fizz  
2026-01-11T07:35:59.6262971Z 88  
2026-01-11T07:35:59.6263211Z 89  
2026-01-11T07:35:59.6263423Z FizzBuzz  
2026-01-11T07:35:59.6266110Z 91  
2026-01-11T07:35:59.6267107Z 92  
2026-01-11T07:35:59.6267325Z Fizz

2026-01-11T07:35:59.6267534Z 94  
2026-01-11T07:35:59.6267838Z Buzz  
2026-01-11T07:35:59.6268155Z Fizz  
2026-01-11T07:35:59.6268427Z 97  
2026-01-11T07:35:59.6268733Z 98  
2026-01-11T07:35:59.6269058Z Fizz  
2026-01-11T07:35:59.6269360Z Buzz  
2026-01-11T07:35:59.6460517Z ##[group]Run python compiler\_gen3.py py1\_compiler.py1 > py1\_compiler.py  
2026-01-11T07:35:59.6461101Z [36;1mpython compiler\_gen3.py py1\_compiler.py1 > py1\_compiler.py[0m  
2026-01-11T07:35:59.6461534Z [36;1mpython py1\_compiler.py fizzbuzz\_while.py1 > fizzbuzz\_new.py[0m  
2026-01-11T07:35:59.6461874Z [36;1mpython fizzbuzz\_new.py[0m  
2026-01-11T07:35:59.6478400Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}  
2026-01-11T07:35:59.6478773Z env:  
2026-01-11T07:35:59.6478947Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:59.6479157Z PYTHONUTF8: 1  
2026-01-11T07:35:59.6479335Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:59.6479604Z pythonLocation: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:59.6480047Z PKG\_CONFIG\_PATH: C:  
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig  
2026-01-11T07:35:59.6480463Z Python\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:59.6480871Z Python2\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:59.6481241Z Python3\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:59.6481539Z ##[endgroup]  
2026-01-11T07:35:59.8154694Z 1  
2026-01-11T07:35:59.8154978Z 2  
2026-01-11T07:35:59.8155204Z Fizz  
2026-01-11T07:35:59.8155422Z 4  
2026-01-11T07:35:59.8155674Z Buzz  
2026-01-11T07:35:59.8155894Z Fizz  
2026-01-11T07:35:59.8156106Z 7  
2026-01-11T07:35:59.8156323Z 8  
2026-01-11T07:35:59.8156529Z Fizz  
2026-01-11T07:35:59.8156751Z Buzz

2026-01-11T07:35:59.8156959Z 11  
2026-01-11T07:35:59.8157178Z Fizz  
2026-01-11T07:35:59.8157385Z 13  
2026-01-11T07:35:59.8157604Z 14  
2026-01-11T07:35:59.8157820Z FizzBuzz  
2026-01-11T07:35:59.8158060Z 16  
2026-01-11T07:35:59.8158296Z 17  
2026-01-11T07:35:59.8158515Z Fizz  
2026-01-11T07:35:59.8158739Z 19  
2026-01-11T07:35:59.8158943Z Buzz  
2026-01-11T07:35:59.8159162Z Fizz  
2026-01-11T07:35:59.8159391Z 22  
2026-01-11T07:35:59.8159599Z 23  
2026-01-11T07:35:59.8159803Z Fizz  
2026-01-11T07:35:59.8160013Z Buzz  
2026-01-11T07:35:59.8160236Z 26  
2026-01-11T07:35:59.8160449Z Fizz  
2026-01-11T07:35:59.8160658Z 28  
2026-01-11T07:35:59.8160869Z 29  
2026-01-11T07:35:59.8161089Z FizzBuzz  
2026-01-11T07:35:59.8161329Z 31  
2026-01-11T07:35:59.8161539Z 32  
2026-01-11T07:35:59.8161749Z Fizz  
2026-01-11T07:35:59.8161957Z 34  
2026-01-11T07:35:59.8162165Z Buzz  
2026-01-11T07:35:59.8162382Z Fizz  
2026-01-11T07:35:59.8162587Z 37  
2026-01-11T07:35:59.8162787Z 38  
2026-01-11T07:35:59.8162989Z Fizz  
2026-01-11T07:35:59.8163184Z Buzz  
2026-01-11T07:35:59.8163359Z 41  
2026-01-11T07:35:59.8163569Z Fizz  
2026-01-11T07:35:59.8163791Z 43  
2026-01-11T07:35:59.8163999Z 44  
2026-01-11T07:35:59.8164210Z FizzBuzz  
2026-01-11T07:35:59.8164462Z 46  
2026-01-11T07:35:59.8164680Z 47  
2026-01-11T07:35:59.8164884Z Fizz  
2026-01-11T07:35:59.8165099Z 49  
2026-01-11T07:35:59.8165307Z Buzz  
2026-01-11T07:35:59.8165520Z Fizz

2026-01-11T07:35:59.8165726Z 52  
2026-01-11T07:35:59.8165937Z 53  
2026-01-11T07:35:59.8166136Z Fizz  
2026-01-11T07:35:59.8166350Z Buzz  
2026-01-11T07:35:59.8166570Z 56  
2026-01-11T07:35:59.8166777Z Fizz  
2026-01-11T07:35:59.8166979Z 58  
2026-01-11T07:35:59.8167185Z 59  
2026-01-11T07:35:59.8167366Z FizzBuzz  
2026-01-11T07:35:59.8167574Z 61  
2026-01-11T07:35:59.8167773Z 62  
2026-01-11T07:35:59.8167985Z Fizz  
2026-01-11T07:35:59.8168193Z 64  
2026-01-11T07:35:59.8168403Z Buzz  
2026-01-11T07:35:59.8168619Z Fizz  
2026-01-11T07:35:59.8168824Z 67  
2026-01-11T07:35:59.8169033Z 68  
2026-01-11T07:35:59.8169239Z Fizz  
2026-01-11T07:35:59.8169454Z Buzz  
2026-01-11T07:35:59.8169660Z 71  
2026-01-11T07:35:59.8169874Z Fizz  
2026-01-11T07:35:59.8170573Z 73  
2026-01-11T07:35:59.8170798Z 74  
2026-01-11T07:35:59.8171013Z FizzBuzz  
2026-01-11T07:35:59.8171249Z 76  
2026-01-11T07:35:59.8171765Z 77  
2026-01-11T07:35:59.8171977Z Fizz  
2026-01-11T07:35:59.8172180Z 79  
2026-01-11T07:35:59.8172389Z Buzz  
2026-01-11T07:35:59.8172588Z Fizz  
2026-01-11T07:35:59.8172807Z 82  
2026-01-11T07:35:59.8173015Z 83  
2026-01-11T07:35:59.8173221Z Fizz  
2026-01-11T07:35:59.8175775Z Buzz  
2026-01-11T07:35:59.8176059Z 86  
2026-01-11T07:35:59.8176289Z Fizz  
2026-01-11T07:35:59.8176511Z 88  
2026-01-11T07:35:59.8176730Z 89  
2026-01-11T07:35:59.8176941Z FizzBuzz  
2026-01-11T07:35:59.8177181Z 91  
2026-01-11T07:35:59.8177377Z 92

2026-01-11T07:35:59.8177591Z Fizz  
2026-01-11T07:35:59.8177800Z 94  
2026-01-11T07:35:59.8178009Z Buzz  
2026-01-11T07:35:59.8178215Z Fizz  
2026-01-11T07:35:59.8178424Z 97  
2026-01-11T07:35:59.8178627Z 98  
2026-01-11T07:35:59.8178836Z Fizz  
2026-01-11T07:35:59.8179043Z Buzz  
2026-01-11T07:35:59.8378340Z ##[group]Run python compiler\_gen3.py py1\_compiler.py1 > stage1\_compiler.py  
2026-01-11T07:35:59.8378955Z [36;1mpython compiler\_gen3.py py1\_compiler.py1 > stage1\_compiler.py[0m  
2026-01-11T07:35:59.8379489Z [36;1mpython stage1\_compiler.py py1\_compiler.py1 > stage2\_compiler.py[0m  
2026-01-11T07:35:59.8379907Z [36;1mpython stage2\_compiler.py py1repl.py1 > py1repl\_final.py[0m  
2026-01-11T07:35:59.8380251Z [36;1mpython py1repl\_final.py [fizzbuzz.ir](#)[0m  
2026-01-11T07:35:59.8396369Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}  
2026-01-11T07:35:59.8396732Z env:  
2026-01-11T07:35:59.8396895Z PYTHONIOENCODING: utf-8  
2026-01-11T07:35:59.8397097Z PYTHONUTF8: 1  
2026-01-11T07:35:59.8397262Z PYTHONUNBUFFERED: 1  
2026-01-11T07:35:59.8397532Z pythonLocation: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:59.8397946Z PKG\_CONFIG\_PATH: C:  
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig  
2026-01-11T07:35:59.8398388Z Python\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:59.8398759Z Python2\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:59.8399148Z Python3\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:35:59.8399439Z ##[endgroup]  
2026-01-11T07:36:00.0736991Z 1  
2026-01-11T07:36:00.0737203Z 2  
2026-01-11T07:36:00.0737712Z Fizz  
2026-01-11T07:36:00.0737961Z 4  
2026-01-11T07:36:00.0738312Z Buzz  
2026-01-11T07:36:00.0739123Z Fizz  
2026-01-11T07:36:00.0739740Z 7

2026-01-11T07:36:00.0740415Z 8  
2026-01-11T07:36:00.0741005Z Fizz  
2026-01-11T07:36:00.0742038Z Buzz  
2026-01-11T07:36:00.0742333Z 11  
2026-01-11T07:36:00.0742563Z Fizz  
2026-01-11T07:36:00.0743345Z 13  
2026-01-11T07:36:00.0743944Z 14  
2026-01-11T07:36:00.0744465Z FizzBuzz  
2026-01-11T07:36:00.0745109Z 16  
2026-01-11T07:36:00.0745747Z 17  
2026-01-11T07:36:00.0746337Z Fizz  
2026-01-11T07:36:00.0746980Z 19  
2026-01-11T07:36:00.0747615Z Buzz  
2026-01-11T07:36:00.0748204Z Fizz  
2026-01-11T07:36:00.0748868Z 22  
2026-01-11T07:36:00.0749503Z 23  
2026-01-11T07:36:00.0750076Z Fizz  
2026-01-11T07:36:00.0750708Z Buzz  
2026-01-11T07:36:00.0751363Z 26  
2026-01-11T07:36:00.0751944Z Fizz  
2026-01-11T07:36:00.0752594Z 28  
2026-01-11T07:36:00.0753970Z 29  
2026-01-11T07:36:00.0754282Z FizzBuzz  
2026-01-11T07:36:00.0754519Z 31  
2026-01-11T07:36:00.0754784Z 32  
2026-01-11T07:36:00.0755710Z Fizz  
2026-01-11T07:36:00.0755889Z 34  
2026-01-11T07:36:00.0756672Z Buzz  
2026-01-11T07:36:00.0757234Z Fizz  
2026-01-11T07:36:00.0757744Z 37  
2026-01-11T07:36:00.0758555Z 38  
2026-01-11T07:36:00.0758885Z Fizz  
2026-01-11T07:36:00.0760629Z Buzz  
2026-01-11T07:36:00.0761544Z 41  
2026-01-11T07:36:00.0762163Z Fizz  
2026-01-11T07:36:00.0763035Z 43  
2026-01-11T07:36:00.0763263Z 44  
2026-01-11T07:36:00.0763483Z FizzBuzz  
2026-01-11T07:36:00.0763746Z 46  
2026-01-11T07:36:00.0764835Z 47  
2026-01-11T07:36:00.0765073Z Fizz

2026-01-11T07:36:00.0765534Z 49  
2026-01-11T07:36:00.0766453Z Buzz  
2026-01-11T07:36:00.0766752Z Fizz  
2026-01-11T07:36:00.0767759Z 52  
2026-01-11T07:36:00.0768028Z 53  
2026-01-11T07:36:00.0768889Z Fizz  
2026-01-11T07:36:00.0769277Z Buzz  
2026-01-11T07:36:00.0770218Z 56  
2026-01-11T07:36:00.0770487Z Fizz  
2026-01-11T07:36:00.0771231Z 58  
2026-01-11T07:36:00.0772156Z 59  
2026-01-11T07:36:00.0773620ZFizzBuzz  
2026-01-11T07:36:00.0773914Z 61  
2026-01-11T07:36:00.0774140Z 62  
2026-01-11T07:36:00.0774362Z Fizz  
2026-01-11T07:36:00.0775094Z 64  
2026-01-11T07:36:00.0775573Z Buzz  
2026-01-11T07:36:00.0776090Z Fizz  
2026-01-11T07:36:00.0776585Z 67  
2026-01-11T07:36:00.0776916Z 68  
2026-01-11T07:36:00.0777236Z Fizz  
2026-01-11T07:36:00.0777958Z Buzz  
2026-01-11T07:36:00.0778685Z 71  
2026-01-11T07:36:00.0780534Z Fizz  
2026-01-11T07:36:00.0780921Z 73  
2026-01-11T07:36:00.0781267Z 74  
2026-01-11T07:36:00.0781942ZFizzBuzz  
2026-01-11T07:36:00.0782287Z 76  
2026-01-11T07:36:00.0782623Z 77  
2026-01-11T07:36:00.0783308Z Fizz  
2026-01-11T07:36:00.0784606Z 79  
2026-01-11T07:36:00.0785517Z Buzz  
2026-01-11T07:36:00.0786112Z Fizz  
2026-01-11T07:36:00.0789906Z 82  
2026-01-11T07:36:00.0790265Z 83  
2026-01-11T07:36:00.0790941Z Fizz  
2026-01-11T07:36:00.0791314Z Buzz  
2026-01-11T07:36:00.0791453Z 86  
2026-01-11T07:36:00.0791580Z Fizz  
2026-01-11T07:36:00.0791814Z 88  
2026-01-11T07:36:00.0792023Z 89

2026-01-11T07:36:00.0792156Z FizzBuzz  
2026-01-11T07:36:00.0792292Z 91  
2026-01-11T07:36:00.0792498Z 92  
2026-01-11T07:36:00.0792743Z Fizz  
2026-01-11T07:36:00.0793644Z 94  
2026-01-11T07:36:00.0794371Z Buzz  
2026-01-11T07:36:00.0795057Z Fizz  
2026-01-11T07:36:00.0796050Z 97  
2026-01-11T07:36:00.0796442Z 98  
2026-01-11T07:36:00.0797287Z Fizz  
2026-01-11T07:36:00.0797951Z Buzz  
2026-01-11T07:36:00.1023284Z ##[group]Run # --- 1. WinIR Generator Spec ---  
2026-01-11T07:36:00.1024289Z [36;1m# --- 1. WinIR Generator Spec ---[0m  
2026-01-11T07:36:00.1024562Z [36;1mcat <<EOF > win\_ir\_spec.py1[0m  
2026-01-11T07:36:00.1024802Z [36;1m# Windows Native IR Generator[0m  
2026-01-11T07:36:00.1025036Z [36;1m@v 表 'print'[0m  
2026-01-11T07:36:00.1025222Z [36;1m@v 追 'append'[0m  
2026-01-11T07:36:00.1025398Z [36;1m@v 字 'str'[0m  
2026-01-11T07:36:00.1025567Z [36;1m@v 結 "'\n".join'[0m  
2026-01-11T07:36:00.1025737Z [36;1m[0m  
2026-01-11T07:36:00.1025950Z [36;1m# Opcodes (Concatenated to avoid strict checks)[0m  
2026-01-11T07:36:00.1026243Z [36;1m@v 置 "'M"+"O"+"V'"[0m  
2026-01-11T07:36:00.1026436Z [36;1m@v 取 "'L"+"E"+"A'"[0m  
2026-01-11T07:36:00.1026618Z [36;1m@v 呼 "'C"+"A"+"L"+"L'"[0m  
2026-01-11T07:36:00.1026815Z [36;1m@v 連 "'L"+"O"+"A"+"D'"[0m  
2026-01-11T07:36:00.1026997Z [36;1m@v 得 "'G"+"E"+"T'"[0m  
2026-01-11T07:36:00.1027184Z [36;1m@v 書 "'W"+"R"+"I"+"T"+"E'"[0m  
2026-01-11T07:36:00.1027419Z [36;1m@v 札 "'L"+"A"+"B"+"E"+"L'"[0m  
2026-01-11T07:36:00.1027617Z [36;1m@v 比 "'C"+"M"+"P'"[0m  
2026-01-11T07:36:00.1027803Z [36;1m@v 零 "'J"+"Z'"[0m  
2026-01-11T07:36:00.1027977Z [36;1m@v 飛 "'J"+"M"+"P'"[0m  
2026-01-11T07:36:00.1028151Z [36;1m@v 加 "'A"+"D"+"D'"[0m  
2026-01-11T07:36:00.1028334Z [36;1m@v 押 "'P"+"U"+"S"+"H'"[0m  
2026-01-11T07:36:00.1028515Z [36;1m[0m  
2026-01-11T07:36:00.1028662Z [36;1m# Registers[0m  
2026-01-11T07:36:00.1028823Z [36;1m@v 壱 "'R"+"B"+"X'"[0m  
2026-01-11T07:36:00.1029001Z [36;1m@v 弐 "'R"+"1"+"2'"[0m  
2026-01-11T07:36:00.1029181Z [36;1m@v 肆 "'R"+"1"+"3'"[0m  
2026-01-11T07:36:00.1029354Z [36;1m@v 繼 "'R"+"C"+"X'"[0m  
2026-01-11T07:36:00.1029524Z [36;1m@v 夕 "'R"+"D"+"X'"[0m  
2026-01-11T07:36:00.1029701Z [36;1m@v 蜂 "'R"+"8'"[0m

2026-01-11T07:36:00.1029869Z [36;1m@v 苦 "'R"+"9"[0m  
2026-01-11T07:36:00.1030031Z [36;1m@v 蕃 "'R"+"A"+"X"[0m  
2026-01-11T07:36:00.1030209Z [36;1m@v 繼 "'R"+"C"+"X"[0m  
2026-01-11T07:36:00.1030381Z [36;1m[0m  
2026-01-11T07:36:00.1030517Z [36;1m# Data[0m  
2026-01-11T07:36:00.1030664Z [36;1m@v 口 'codes'[0m  
2026-01-11T07:36:00.1030874Z [36;1m@v 藏  
'"k"+"e"+"r"+"n"+"e"+|"l"+"3"+"2"+"."+"d"+|"l"+|"l"[0m  
2026-01-11T07:36:00.1031178Z [36;1m@v 出  
'"G"+"e"+"t"+"S"+"t"+"d"+"H"+"a"+"n"+"d"+|"l"+|"e"[0m  
2026-01-11T07:36:00.1031767Z [36;1m@v 記 "'W"+"r"+|"i"+"t"+"e"+"F"+|"i"+|"l"+|"e"[0m  
2026-01-11T07:36:00.1032133Z [36;1m@v 終  
'"E"+"x"+|"i"+"t"+"P"+"r"+"o"+"c"+"e"+"s"+"s"[0m  
2026-01-11T07:36:00.1032390Z [36;1m@v 隱 "'-+"1"+"1"[0m  
2026-01-11T07:36:00.1032572Z [36;1m@v 空 '' ''[0m  
2026-01-11T07:36:00.1032950Z [36;1m[0m  
2026-01-11T07:36:00.1033091Z [36;1m# Constants[0m  
2026-01-11T07:36:00.1033248Z [36;1m@v — "'1"[0m  
2026-01-11T07:36:00.1033410Z [36;1m@v + "'1"+"0"[0m  
2026-01-11T07:36:00.1033570Z [36;1m@v 限 "'1"+"7"[0m  
2026-01-11T07:36:00.1033733Z [36;1m@v 二 "'2"[0m  
2026-01-11T07:36:00.1033883Z [36;1m@v 三 "'3"[0m  
2026-01-11T07:36:00.1034026Z [36;1m[0m  
2026-01-11T07:36:00.1034174Z [36;1m# Buffers[0m  
2026-01-11T07:36:00.1034337Z [36;1m@v 壺 "'1"+"0"+"0"[0m  
2026-01-11T07:36:00.1034519Z [36;1m@v 泡 "'2"+"0"+"0"[0m  
2026-01-11T07:36:00.1034690Z [36;1m@v 鳴 "'3"+"0"+"0"[0m  
2026-01-11T07:36:00.1034863Z [36;1m@v 混 "'4"+"0"+"0"[0m  
2026-01-11T07:36:00.1035029Z [36;1m@v 衍 "'5"+"0"+"0"[0m  
2026-01-11T07:36:00.1035199Z [36;1m@v 針 "'6"+"0"+"0"[0m  
2026-01-11T07:36:00.1035361Z [36;1m[0m  
2026-01-11T07:36:00.1035503Z [36;1m# Values[0m  
2026-01-11T07:36:00.1035658Z [36;1m@v 穴 ''0''[0m  
2026-01-11T07:36:00.1035810Z [36;1m@v 改 "'1"+"0"[0m  
2026-01-11T07:36:00.1035988Z [36;1m@v 父 "'7"+"0"[0m  
2026-01-11T07:36:00.1036147Z [36;1m@v 愛 "'1"+"0"+"5"[0m  
2026-01-11T07:36:00.1036322Z [36;1m@v 寢 "'1"+"2"+"2"[0m  
2026-01-11T07:36:00.1036491Z [36;1m@v 豚 "'6"+"6"[0m  
2026-01-11T07:36:00.1036653Z [36;1m@v 鶲 "'1"+"1"+"7"[0m  
2026-01-11T07:36:00.1036817Z [36;1m@v 丸 "'4"+"8"[0m  
2026-01-11T07:36:00.1036979Z [36;1m@v 棒 "'4"+"9"[0m

2026-01-11T07:36:00.1038686Z [36;1m@v 損 ""-""+"1"+ "0"] [0m  
2026-01-11T07:36:00.1038936Z [36;1m[0m  
2026-01-11T07:36:00.1039089Z [36;1m# Labels[0m  
2026-01-11T07:36:00.1039255Z [36;1m@v 囇 ""L""+"O""+"O""+"P"] [0m  
2026-01-11T07:36:00.1039464Z [36;1m@v 去 ""E""+"X""+"I""+"T"] [0m  
2026-01-11T07:36:00.1039648Z [36;1m@v 甲 ""L""+"F"] [0m  
2026-01-11T07:36:00.1039826Z [36;1m@v 乙 ""L""+"B"] [0m  
2026-01-11T07:36:00.1039993Z [36;1m@v 丙 ""L""+"F""+"B"] [0m  
2026-01-11T07:36:00.1040178Z [36;1m@v 丁 ""L""+"2""+"D"] [0m  
2026-01-11T07:36:00.1040351Z [36;1m@v 次 ""N""+"X"] [0m  
2026-01-11T07:36:00.1040513Z [36;1m[0m  
2026-01-11T07:36:00.1040654Z [36;1m\$[0m  
2026-01-11T07:36:00.1040793Z [36;1m[0m  
2026-01-11T07:36:00.1040963Z [36;1m 口 = [] [0m  
2026-01-11T07:36:00.1041110Z [36;1m[0m  
2026-01-11T07:36:00.1041254Z [36;1m# --- Init --- [0m  
2026-01-11T07:36:00.1041418Z [36;1m 口.追(連 + 空 + 藏) [0m  
2026-01-11T07:36:00.1041581Z [36;1m 口.追(得 + 空 + 出) [0m  
2026-01-11T07:36:00.1041741Z [36;1m 口.追(置 + 空 + 繼 + 空 + 隱) [0m  
2026-01-11T07:36:00.1041935Z [36;1m 口.追(呼 + 空 + 出) [0m  
2026-01-11T07:36:00.1042095Z [36;1m 口.追(置 + 空 + 肆 + 空 + 蓄) [0m  
2026-01-11T07:36:00.1042277Z [36;1m 口.追(得 + 空 + 記) [0m  
2026-01-11T07:36:00.1042455Z [36;1m 口.追(得 + 空 + 終) [0m  
2026-01-11T07:36:00.1042614Z [36;1m[0m  
2026-01-11T07:36:00.1042770Z [36;1m# --- Prepare Buffers --- [0m  
2026-01-11T07:36:00.1042978Z [36;1m 口.追(書 + 空 + 泡 + 空 + 父) [0m  
2026-01-11T07:36:00.1043176Z [36;1m 口.追(書 + 空 + 字(201) + 空 + 愛) [0m  
2026-01-11T07:36:00.1043374Z [36;1m 口.追(書 + 空 + 字(202) + 空 + 寢) [0m  
2026-01-11T07:36:00.1043572Z [36;1m 口.追(書 + 空 + 字(203) + 空 + 寢) [0m  
2026-01-11T07:36:00.1043764Z [36;1m 口.追(書 + 空 + 字(204) + 空 + 改) [0m  
2026-01-11T07:36:00.1043982Z [36;1m[0m  
2026-01-11T07:36:00.1044120Z [36;1m 口.追(書 + 空 + 鳴 + 空 + 豚) [0m  
2026-01-11T07:36:00.1044311Z [36;1m 口.追(書 + 空 + 字(301) + 空 + 鶴) [0m  
2026-01-11T07:36:00.1044507Z [36;1m 口.追(書 + 空 + 字(302) + 空 + 寢) [0m  
2026-01-11T07:36:00.1044693Z [36;1m 口.追(書 + 空 + 字(303) + 空 + 寢) [0m  
2026-01-11T07:36:00.1044884Z [36;1m 口.追(書 + 空 + 字(304) + 空 + 改) [0m  
2026-01-11T07:36:00.1045080Z [36;1m[0m  
2026-01-11T07:36:00.1045219Z [36;1m 口.追(書 + 空 + 混 + 空 + 父) [0m  
2026-01-11T07:36:00.1045402Z [36;1m 口.追(書 + 空 + 字(401) + 空 + 愛) [0m  
2026-01-11T07:36:00.1045598Z [36;1m 口.追(書 + 空 + 字(402) + 空 + 寢) [0m  
2026-01-11T07:36:00.1045786Z [36;1m 口.追(書 + 空 + 字(403) + 空 + 寢) [0m

2026-01-11T07:36:00.1045977Z [36;1m 口.追(書 + 空 + 字(404) + 空 + 豚)[0m  
2026-01-11T07:36:00.1046169Z [36;1m 口.追(書 + 空 + 字(405) + 空 + 鶏)[0m  
2026-01-11T07:36:00.1046356Z [36;1m 口.追(書 + 空 + 字(406) + 空 + 寢)[0m  
2026-01-11T07:36:00.1046567Z [36;1m 口.追(書 + 空 + 字(407) + 空 + 寢)[0m  
2026-01-11T07:36:00.1046892Z [36;1m 口.追(書 + 空 + 字(408) + 空 + 改)[0m  
2026-01-11T07:36:00.1047087Z [36;1m[0m  
2026-01-11T07:36:00.1047233Z [36;1m# --- Loop Start ---[0m  
2026-01-11T07:36:00.1047427Z [36;1m 口.追(置 + 空 + 壱 + 空 + 一)[0m  
2026-01-11T07:36:00.1047607Z [36;1m 口.追(札 + 空 + 回)[0m  
2026-01-11T07:36:00.1047778Z [36;1m[0m  
2026-01-11T07:36:00.1047914Z [36;1m# Check Limit[0m  
2026-01-11T07:36:00.1048104Z [36;1m 口.追(置 + 空 + 弐 + 空 + 限)[0m  
2026-01-11T07:36:00.1048289Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1048466Z [36;1m 口.追(零 + 空 + 去)[0m  
2026-01-11T07:36:00.1048630Z [36;1m[0m  
2026-01-11T07:36:00.1048767Z [36;1m# Logic[0m  
2026-01-11T07:36:00.1048924Z [36;1m 口.追(置 + 空 + 弐 + 空 + 字(15))[0m  
2026-01-11T07:36:00.1049113Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1049299Z [36;1m 口.追(零 + 空 + 丙)[0m  
2026-01-11T07:36:00.1049451Z [36;1m[0m  
2026-01-11T07:36:00.1049589Z [36;1m 口.追(置 + 空 + 弐 + 空 + 字(5))[0m  
2026-01-11T07:36:00.1049777Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1049973Z [36;1m 口.追(零 + 空 + 乙)[0m  
2026-01-11T07:36:00.1050140Z [36;1m 口.追(置 + 空 + 弐 + 空 + 字(10))[0m  
2026-01-11T07:36:00.1050327Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1050506Z [36;1m 口.追(零 + 空 + 乙)[0m  
2026-01-11T07:36:00.1050657Z [36;1m[0m  
2026-01-11T07:36:00.1050792Z [36;1m 口.追(置 + 空 + 弐 + 空 + 字(3))[0m  
2026-01-11T07:36:00.1050979Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1051611Z [36;1m 口.追(零 + 空 + 甲)[0m  
2026-01-11T07:36:00.1052206Z [36;1m 口.追(置 + 空 + 弐 + 空 + 字(6))[0m  
2026-01-11T07:36:00.1052416Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1052610Z [36;1m 口.追(零 + 空 + 甲)[0m  
2026-01-11T07:36:00.1052781Z [36;1m 口.追(置 + 空 + 弐 + 空 + 字(9))[0m  
2026-01-11T07:36:00.1052979Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1053157Z [36;1m 口.追(零 + 空 + 甲)[0m  
2026-01-11T07:36:00.1053330Z [36;1m 口.追(置 + 空 + 弐 + 空 + 字(12))[0m  
2026-01-11T07:36:00.1053518Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1053702Z [36;1m 口.追(零 + 空 + 甲)[0m  
2026-01-11T07:36:00.1053872Z [36;1m[0m  
2026-01-11T07:36:00.1054017Z [36;1m# Check 2 digits[0m

2026-01-11T07:36:00.1054202Z [36;1m 口.追(置 + 空 + 弐 + 空 + 字(11))[0m  
2026-01-11T07:36:00.1054389Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1054570Z [36;1m 口.追(零 + 空 + 丁)[0m  
2026-01-11T07:36:00.1054732Z [36;1m 口.追(置 + 空 + 弐 + 空 + 字(13))[0m  
2026-01-11T07:36:00.1054928Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1055121Z [36;1m 口.追(零 + 空 + 丁)[0m  
2026-01-11T07:36:00.1055291Z [36;1m 口.追(置 + 空 + 弐 + 空 + 字(14))[0m  
2026-01-11T07:36:00.1055515Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1055699Z [36;1m 口.追(零 + 空 + 丁)[0m  
2026-01-11T07:36:00.1055866Z [36;1m 口.追(置 + 空 + 弐 + 空 + 字(16))[0m  
2026-01-11T07:36:00.1056056Z [36;1m 口.追(比 + 空 + 壱 + 空 + 弐)[0m  
2026-01-11T07:36:00.1056242Z [36;1m 口.追(零 + 空 + 丁)[0m  
2026-01-11T07:36:00.1056398Z [36;1m[0m  
2026-01-11T07:36:00.1056544Z [36;1m# 1 Digit Logic[0m  
2026-01-11T07:36:00.1056715Z [36;1m 口.追(置 + 空 + 蕃 + 空 + 壱)[0m  
2026-01-11T07:36:00.1057042Z [36;1m 口.追(加 + 空 + 蕃 + 空 + 丸)[0m  
2026-01-11T07:36:00.1057221Z [36;1m 口.追(書 + 空 + 壺 + 空 + 蕃)[0m  
2026-01-11T07:36:00.1057415Z [36;1m 口.追(書 + 空 + 字(101) + 空 + 改)[0m  
2026-01-11T07:36:00.1057607Z [36;1m 口.追(置 + 空 + 繰 + 空 + 肆)[0m  
2026-01-11T07:36:00.1057790Z [36;1m 口.追(取 + 空 + 夕 + 空 + 壺)[0m  
2026-01-11T07:36:00.1057977Z [36;1m 口.追(置 + 空 + 蜂 + 空 + 二)[0m  
2026-01-11T07:36:00.1058168Z [36;1m 口.追(取 + 空 + 苦 + 空 + 針)[0m  
2026-01-11T07:36:00.1058349Z [36;1m 口.追(押 + 空 + 穴)[0m  
2026-01-11T07:36:00.1058511Z [36;1m 口.追(呼 + 空 + 記)[0m  
2026-01-11T07:36:00.1058677Z [36;1m 口.追(飛 + 空 + 次)[0m  
2026-01-11T07:36:00.1058828Z [36;1m[0m  
2026-01-11T07:36:00.1058964Z [36;1m# 2 Digit Logic[0m  
2026-01-11T07:36:00.1059134Z [36;1m 口.追(札 + 空 + 丁)[0m  
2026-01-11T07:36:00.1059297Z [36;1m 口.追(書 + 空 + 衎 + 空 + 棒)[0m  
2026-01-11T07:36:00.1059481Z [36;1m 口.追(置 + 空 + 蕃 + 空 + 壱)[0m  
2026-01-11T07:36:00.1059659Z [36;1m 口.追(加 + 空 + 蕃 + 空 + 損)[0m  
2026-01-11T07:36:00.1059840Z [36;1m 口.追(加 + 空 + 蕃 + 空 + 丸)[0m  
2026-01-11T07:36:00.1060029Z [36;1m 口.追(書 + 空 + 字(501) + 空 + 蕃)[0m  
2026-01-11T07:36:00.1060226Z [36;1m 口.追(書 + 空 + 字(502) + 空 + 改)[0m  
2026-01-11T07:36:00.1060414Z [36;1m 口.追(置 + 空 + 繰 + 空 + 肆)[0m  
2026-01-11T07:36:00.1060592Z [36;1m 口.追(取 + 空 + 夕 + 空 + 衎)[0m  
2026-01-11T07:36:00.1060767Z [36;1m 口.追(置 + 空 + 蜂 + 空 + 三)[0m  
2026-01-11T07:36:00.1060944Z [36;1m 口.追(取 + 空 + 苦 + 空 + 針)[0m  
2026-01-11T07:36:00.1061222Z [36;1m 口.追(押 + 空 + 穴)[0m  
2026-01-11T07:36:00.1061386Z [36;1m 口.追(呼 + 空 + 記)[0m  
2026-01-11T07:36:00.1061549Z [36;1m 口.追(飛 + 空 + 次)[0m

2026-01-11T07:36:00.1061703Z [36;1m[0m  
2026-01-11T07:36:00.1061843Z [36;1m# Print Fizz[0m  
2026-01-11T07:36:00.1062006Z [36;1m 口.追(札 + 空 + 甲)[0m  
2026-01-11T07:36:00.1062169Z [36;1m 口.追(置 + 空 + 繰 + 空 + 肆)[0m  
2026-01-11T07:36:00.1062343Z [36;1m 口.追(取 + 空 + 夕 + 空 + 泡)[0m  
2026-01-11T07:36:00.1062525Z [36;1m 口.追(置 + 空 + 蜂 + 空 + 字(5))[0m  
2026-01-11T07:36:00.1062713Z [36;1m 口.追(取 + 空 + 苦 + 空 + 針)[0m  
2026-01-11T07:36:00.1062897Z [36;1m 口.追(押 + 空 + 穴)[0m  
2026-01-11T07:36:00.1063077Z [36;1m 口.追(呼 + 空 + 記)[0m  
2026-01-11T07:36:00.1063233Z [36;1m 口.追(飛 + 空 + 次)[0m  
2026-01-11T07:36:00.1063392Z [36;1m[0m  
2026-01-11T07:36:00.1063530Z [36;1m# Print Buzz[0m  
2026-01-11T07:36:00.1063696Z [36;1m 口.追(札 + 空 + 乙)[0m  
2026-01-11T07:36:00.1063853Z [36;1m 口.追(置 + 空 + 繰 + 空 + 肆)[0m  
2026-01-11T07:36:00.1064036Z [36;1m 口.追(取 + 空 + 夕 + 空 + 鳴)[0m  
2026-01-11T07:36:00.1064231Z [36;1m 口.追(置 + 空 + 蜂 + 空 + 字(5))[0m  
2026-01-11T07:36:00.1064420Z [36;1m 口.追(取 + 空 + 苦 + 空 + 針)[0m  
2026-01-11T07:36:00.1064601Z [36;1m 口.追(押 + 空 + 穴)[0m  
2026-01-11T07:36:00.1064758Z [36;1m 口.追(呼 + 空 + 記)[0m  
2026-01-11T07:36:00.1064923Z [36;1m 口.追(飛 + 空 + 次)[0m  
2026-01-11T07:36:00.1065076Z [36;1m[0m  
2026-01-11T07:36:00.1065233Z [36;1m# Print FizzBuzz[0m  
2026-01-11T07:36:00.1065406Z [36;1m 口.追(札 + 空 + 丙)[0m  
2026-01-11T07:36:00.1065568Z [36;1m 口.追(置 + 空 + 繰 + 空 + 肆)[0m  
2026-01-11T07:36:00.1065748Z [36;1m 口.追(取 + 空 + 夕 + 空 + 混)[0m  
2026-01-11T07:36:00.1065940Z [36;1m 口.追(置 + 空 + 蜂 + 空 + 字(9))[0m  
2026-01-11T07:36:00.1066138Z [36;1m 口.追(取 + 空 + 苦 + 空 + 針)[0m  
2026-01-11T07:36:00.1066323Z [36;1m 口.追(押 + 空 + 穴)[0m  
2026-01-11T07:36:00.1066489Z [36;1m 口.追(呼 + 空 + 記)[0m  
2026-01-11T07:36:00.1066645Z [36;1m 口.追(飛 + 空 + 次)[0m  
2026-01-11T07:36:00.1066807Z [36;1m[0m  
2026-01-11T07:36:00.1066942Z [36;1m# Loop End[0m  
2026-01-11T07:36:00.1067104Z [36;1m 口.追(札 + 空 + 次)[0m  
2026-01-11T07:36:00.1067281Z [36;1m 口.追(加 + 空 + 壱 + 空 + 一)[0m  
2026-01-11T07:36:00.1067460Z [36;1m 口.追(飛 + 空 + 囇)[0m  
2026-01-11T07:36:00.1067611Z [36;1m[0m  
2026-01-11T07:36:00.1067743Z [36;1m# Exit[0m  
2026-01-11T07:36:00.1067887Z [36;1m 口.追(札 + 空 + 去)[0m  
2026-01-11T07:36:00.1068050Z [36;1m 口.追(置 + 空 + 繰 + 空 + 穴)[0m  
2026-01-11T07:36:00.1068231Z [36;1m 口.追(呼 + 空 + 終)[0m  
2026-01-11T07:36:00.1068381Z [36;1m[0m

2026-01-11T07:36:00.1068517Z [36;1m表(結(口))[0m  
2026-01-11T07:36:00.1068661Z [36;1mEOF[0m  
2026-01-11T07:36:00.1068800Z [36;1m[0m  
2026-01-11T07:36:00.1068988Z [36;1m# Generate IR (DEBUG: Print error if failed)[0m  
2026-01-11T07:36:00.1069682Z [36;1mpython stage2\_compiler.py win\_ir\_spec.py1 >  
win\_ir\_gen.py || (echo "--- IR Spec Compilation Failed ---" && cat win\_ir\_gen.py && exit 1)[0m  
2026-01-11T07:36:00.1070256Z [36;1mpython win\_ir\_gen.py > [fizzbuzz\\_win.ir](#)[0m  
2026-01-11T07:36:00.1070488Z [36;1m[0m  
2026-01-11T07:36:00.1070733Z [36;1m # --- 2. Mock VM (Fix: Use '安' helper for WRITE  
instruction) ---[0m  
2026-01-11T07:36:00.1071057Z [36;1mcat <<EOF > vm\_win\_mock.py1[0m  
2026-01-11T07:36:00.1071279Z [36;1m# Mock VM with safety check[0m  
2026-01-11T07:36:00.1071480Z [36;1m@v 表 'print'[0m  
2026-01-11T07:36:00.1071647Z [36;1m@v 整 'int'[0m  
2026-01-11T07:36:00.1071798Z [36;1m@v 寸 'len'[0m  
2026-01-11T07:36:00.1072260Z [36;1m@v 追 'append'[0m  
2026-01-11T07:36:00.1072446Z [36;1m@v 裂 'split'[0m  
2026-01-11T07:36:00.1072605Z [36;1m@v 削 'strip'[0m  
2026-01-11T07:36:00.1072776Z [36;1m@v 行 'splitlines'[0m  
2026-01-11T07:36:00.1072957Z [36;1m@v 開 'open'[0m  
2026-01-11T07:36:00.1073128Z [36;1m@v 讀 'read'[0m  
2026-01-11T07:36:00.1073285Z [36;1m@v 換 'replace'[0m  
2026-01-11T07:36:00.1073464Z [36;1m@v 始 'startswith'[0m  
2026-01-11T07:36:00.1073635Z [36;1m@v 終 'exit'[0m  
2026-01-11T07:36:00.1073794Z [36;1m@v 実 'exec'[0m  
2026-01-11T07:36:00.1073945Z [36;1m@v 字 'chr'[0m  
2026-01-11T07:36:00.1074107Z [36;1m@v 数 'isnumeric'[0m  
2026-01-11T07:36:00.1074283Z [36;1m@v 捨 'get'[0m  
2026-01-11T07:36:00.1074542Z [36;1m[0m  
2026-01-11T07:36:00.1074683Z [36;1m@v 系 'sys'[0m  
2026-01-11T07:36:00.1074832Z [36;1m@v 係 'argv'[0m  
2026-01-11T07:36:00.1074997Z [36;1m@v 込 '"import sys"'[0m  
2026-01-11T07:36:00.1075177Z [36;1m@v 術 'def'[0m  
2026-01-11T07:36:00.1075333Z [36;1m@v も 'if'[0m  
2026-01-11T07:36:00.1075485Z [36;1m@v 他 'else'[0m  
2026-01-11T07:36:00.1075643Z [36;1m@v 循 'while'[0m  
2026-01-11T07:36:00.1075796Z [36;1m@v 入 'in'[0m  
2026-01-11T07:36:00.1075950Z [36;1m@v 或 'elif'[0m  
2026-01-11T07:36:00.1076111Z [36;1m@v 返 'return'[0m  
2026-01-11T07:36:00.1076279Z [36;1m[0m  
2026-01-11T07:36:00.1076423Z [36;1m@v 置 '"M"+"O"+"V"'[0m

2026-01-11T07:36:00.1076600Z [36;1m@v 取 "'L"+"E"+'A'[0m  
2026-01-11T07:36:00.1076783Z [36;1m@v 呼 "'C"+'A"+'L"+'L'[0m  
2026-01-11T07:36:00.1076974Z [36;1m@v 連 "'L"+'O"+'A"+'D'[0m  
2026-01-11T07:36:00.1077166Z [36;1m@v 得 "'G"+'E"+'T'[0m  
2026-01-11T07:36:00.1077342Z [36;1m@v 書 "'W"+'R"+'I"+'T"+'E'[0m  
2026-01-11T07:36:00.1077554Z [36;1m@v 札 "'L"+'A"+'B"+'E"+'L'[0m  
2026-01-11T07:36:00.1077743Z [36;1m@v 比 "'C"+'M"+'P'[0m  
2026-01-11T07:36:00.1077924Z [36;1m@v 零 "'J"+'Z'[0m  
2026-01-11T07:36:00.1078092Z [36;1m@v 飛 "'J"+'M"+'P'[0m  
2026-01-11T07:36:00.1078264Z [36;1m@v 加 "'A"+'D"+'D'[0m  
2026-01-11T07:36:00.1078442Z [36;1m@v 押 "'P"+'U"+'S"+'H'[0m  
2026-01-11T07:36:00.1078618Z [36;1m[0m  
2026-01-11T07:36:00.1078760Z [36;1m@v 外 'args'[0m  
2026-01-11T07:36:00.1078912Z [36;1m@v 径 'path'[0m  
2026-01-11T07:36:00.1079073Z [36;1m@v 本 'body'[0m  
2026-01-11T07:36:00.1079222Z [36;1m@v 生 'lines'[0m  
2026-01-11T07:36:00.1079394Z [36;1m@v 口 'codes'[0m  
2026-01-11T07:36:00.1079547Z [36;1m@v 順 'i'[0m  
2026-01-11T07:36:00.1079698Z [36;1m@v 線 'line'[0m  
2026-01-11T07:36:00.1079859Z [36;1m@v 部 'parts'[0m  
2026-01-11T07:36:00.1080016Z [36;1m@v 技 'op'[0m  
2026-01-11T07:36:00.1080175Z [36;1m@v 偽 'mock\_api'[0m  
2026-01-11T07:36:00.1080345Z [36;1m@v 名 'name'[0m  
2026-01-11T07:36:00.1080503Z [36;1m@v 先 'dst'[0m  
2026-01-11T07:36:00.1080658Z [36;1m@v 元 'src'[0m  
2026-01-11T07:36:00.1080811Z [36;1m@v 値 'val'[0m  
2026-01-11T07:36:00.1080955Z [36;1m@v レ 'regs'[0m  
2026-01-11T07:36:00.1081111Z [36;1m@v メモリ 'mem'[0m  
2026-01-11T07:36:00.1081261Z [36;1m@v 局 'apis'[0m  
2026-01-11T07:36:00.1081420Z [36;1m@v 指 'ip'[0m  
2026-01-11T07:36:00.1081576Z [36;1m@v 辞 'labels'[0m  
2026-01-11T07:36:00.1081735Z [36;1m@v 鍵 'key'[0m  
2026-01-11T07:36:00.1081883Z [36;1m[0m  
2026-01-11T07:36:00.1082013Z [36;1m@v 所 'addr'[0m  
2026-01-11T07:36:00.1082175Z [36;1m@v 基 'buf\_addr'[0m  
2026-01-11T07:36:00.1082344Z [36;1m@v 幅 'length'[0m  
2026-01-11T07:36:00.1082612Z [36;1m@v 器 'buffer'[0m  
2026-01-11T07:36:00.1082769Z [36;1m@v 力 'k'[0m  
2026-01-11T07:36:00.1082926Z [36;1m@v 符 'char\_code'[0m  
2026-01-11T07:36:00.1083097Z [36;1m@v 甲 'val\_a'[0m  
2026-01-11T07:36:00.1083263Z [36;1m@v 乙 'val\_b'[0m

2026-01-11T07:36:00.1083429Z [36;1m[0m  
2026-01-11T07:36:00.1083572Z [36;1m@v 核 "'k"+ "e" + "r" + "n" + "e" + "l"'[0m  
2026-01-11T07:36:00.1083799Z [36;1m@v ハ "'G"+ "e" + "t"'[0m  
2026-01-11T07:36:00.1083978Z [36;1m@v ヲ "'W"+ "r" + "i" + "t" + "e"'[0m  
2026-01-11T07:36:00.1084195Z [36;1m@v 逝 "'E"+ "x" + "i" + "t"'[0m  
2026-01-11T07:36:00.1084383Z [36;1m@v 題 "'M"+ "o" + "c" + "k" + ":"'[0m  
2026-01-11T07:36:00.1084578Z [36;1m@v 間 "' ''[0m  
2026-01-11T07:36:00.1084728Z [36;1m@v モ "'r"'[0m  
2026-01-11T07:36:00.1084891Z [36;1m@v 権 "'u"+ "t" + "f" + "8"'[0m  
2026-01-11T07:36:00.1085100Z [36;1m@v 号 'encoding'[0m  
2026-01-11T07:36:00.1085279Z [36;1m@v 蕎 "'R"+ "A" + "X"'[0m  
2026-01-11T07:36:00.1085461Z [36;1m@v 繰 "'R"+ "C" + "X"'[0m  
2026-01-11T07:36:00.1085643Z [36;1m@v 夕 "'R"+ "D" + "X"'[0m  
2026-01-11T07:36:00.1085815Z [36;1m@v 蜂 "'R"+ "8"'[0m  
2026-01-11T07:36:00.1086115Z [36;1m@v 旗 "'Z"+ "F"'[0m  
2026-01-11T07:36:00.1086373Z [36;1m@v 空 ''''[0m  
2026-01-11T07:36:00.1086651Z [36;1m@v 説 "'Usage"'[0m  
2026-01-11T07:36:00.1097449Z [36;1m[0m  
2026-01-11T07:36:00.1097660Z [36;1m\$[0m  
2026-01-11T07:36:00.1097817Z [36;1m[0m  
2026-01-11T07:36:00.1098640Z [36;1m実(込)[0m  
2026-01-11T07:36:00.1098829Z [36;1m[0m  
2026-01-11T07:36:00.1098989Z [36;1m# Safety Helper[0m  
2026-01-11T07:36:00.1099178Z [36;1m術 安(鍵, レ):[0m  
2026-01-11T07:36:00.1099349Z [36;1m も 鍵 入 レ:[0m  
2026-01-11T07:36:00.1099509Z [36;1m 収(レ[鍵])[0m  
2026-01-11T07:36:00.1099659Z [36;1m 或 鍵.数():[0m  
2026-01-11T07:36:00.1099810Z [36;1m 収(整(鍵))[0m  
2026-01-11T07:36:00.1099955Z [36;1m 他:[0m  
2026-01-11T07:36:00.1100101Z [36;1m 収(0)[0m  
2026-01-11T07:36:00.1100242Z [36;1m[0m  
2026-01-11T07:36:00.1100383Z [36;1m術 偽(名, レ, ×):[0m  
2026-01-11T07:36:00.1100552Z [36;1m も 名.始(ハ):[0m  
2026-01-11T07:36:00.1100727Z [36;1m レ[蓄] = 1[0m  
2026-01-11T07:36:00.1100875Z [36;1m 或 名.始(ヲ):[0m  
2026-01-11T07:36:00.1101026Z [36;1m 基 = 安(夕, レ)[0m  
2026-01-11T07:36:00.1101189Z [36;1m 幅 = 安(蜂, レ)[0m  
2026-01-11T07:36:00.1101343Z [36;1m 器 = 空[0m  
2026-01-11T07:36:00.1101489Z [36;1m 力 = 0[0m  
2026-01-11T07:36:00.1101631Z [36;1m 循 力 < 幅:[0m  
2026-01-11T07:36:00.1101805Z [36;1m 符 = ×.拾(基 + 力, 0)[0m

2026-01-11T07:36:00.1101985Z [36;1m 器 = 器 + 字(符)][0m  
2026-01-11T07:36:00.1102157Z [36;1m 力 = 力 + 1][0m  
2026-01-11T07:36:00.1102321Z [36;1m 表(題 + 器.削())][0m  
2026-01-11T07:36:00.1102488Z [36;1m レ[蓄] = 1][0m  
2026-01-11T07:36:00.1102638Z [36;1m 或名始(逝):][0m  
2026-01-11T07:36:00.1102789Z [36;1m 系終(0)][0m  
2026-01-11T07:36:00.1102934Z [36;1m[0m  
2026-01-11T07:36:00.1103072Z [36;1m 術動(口):][0m  
2026-01-11T07:36:00.1103220Z [36;1m レ = {}][0m  
2026-01-11T07:36:00.1103365Z [36;1m × = {}][0m  
2026-01-11T07:36:00.1103512Z [36;1m 局 = {}][0m  
2026-01-11T07:36:00.1104153Z [36;1m 指 = 0][0m  
2026-01-11T07:36:00.1104304Z [36;1m [0m  
2026-01-11T07:36:00.1104439Z [36;1m 辞 = {}][0m  
2026-01-11T07:36:00.1104584Z [36;1m 順 = 0][0m  
2026-01-11T07:36:00.1104726Z [36;1m 循順 < 寸(口):][0m  
2026-01-11T07:36:00.1104892Z [36;1m 線 = 口[順][0m  
2026-01-11T07:36:00.1105050Z [36;1m 部 = 線.裂(間)][0m  
2026-01-11T07:36:00.1105216Z [36;1m も部[0] == 札:][0m  
2026-01-11T07:36:00.1105399Z [36;1m 辞[部[1]] = 順[0m  
2026-01-11T07:36:00.1105564Z [36;1m 順 = 順 + 1][0m  
2026-01-11T07:36:00.1105724Z [36;1m [0m  
2026-01-11T07:36:00.1105860Z [36;1m 循指 < 寸(口):][0m  
2026-01-11T07:36:00.1106017Z [36;1m 線 = 口[指][0m  
2026-01-11T07:36:00.1106164Z [36;1m 部 = 線.裂(間)][0m  
2026-01-11T07:36:00.1106325Z [36;1m 技 = 部[0][0m  
2026-01-11T07:36:00.1106470Z [36;1m [0m  
2026-01-11T07:36:00.1106618Z [36;1m も技 == 連:][0m  
2026-01-11T07:36:00.1106770Z [36;1m 0[0m  
2026-01-11T07:36:00.1106927Z [36;1m 或技 == 得:][0m  
2026-01-11T07:36:00.1107092Z [36;1m 局[部[1]] = 部[1][0m  
2026-01-11T07:36:00.1107277Z [36;1m 或技 == 呼:][0m  
2026-01-11T07:36:00.1107442Z [36;1m 偽(部[1], レ, ×)][0m  
2026-01-11T07:36:00.1107608Z [36;1m [0m  
2026-01-11T07:36:00.1107749Z [36;1m 或技 == 置:][0m  
2026-01-11T07:36:00.1107899Z [36;1m 先 = 部[1][0m  
2026-01-11T07:36:00.1108054Z [36;1m 元 = 部[2][0m  
2026-01-11T07:36:00.1108202Z [36;1m レ[先] = 安(元, レ)][0m  
2026-01-11T07:36:00.1108377Z [36;1m[0m  
2026-01-11T07:36:00.1108615Z [36;1m 或技 == 取:][0m  
2026-01-11T07:36:00.1108781Z [36;1m 先 = 部[1][0m

2026-01-11T07:36:00.1108940Z [36;1m 元 = 部[2][0m  
2026-01-11T07:36:00.1109103Z [36;1m レ[先] = 安(元, レ)[0m  
2026-01-11T07:36:00.1109275Z [36;1m[0m  
2026-01-11T07:36:00.1109440Z [36;1m # FIXED: Use safety helper here![0m  
2026-01-11T07:36:00.1109670Z [36;1m 或 技 == 書:[0m  
2026-01-11T07:36:00.1109824Z [36;1m 所 = 整(部[1])[0m  
2026-01-11T07:36:00.1109993Z [36;1m 値 = 安(部[2], レ)[0m  
2026-01-11T07:36:00.1110158Z [36;1m メ[所] = 値[0m  
2026-01-11T07:36:00.1110312Z [36;1m [0m  
2026-01-11T07:36:00.1110445Z [36;1m 或 技 == 比:[0m  
2026-01-11T07:36:00.1110609Z [36;1m 先 = 部[1][0m  
2026-01-11T07:36:00.1110757Z [36;1m 元 = 部[2][0m  
2026-01-11T07:36:00.1110914Z [36;1m 甲 = 安(先, レ)[0m  
2026-01-11T07:36:00.1111080Z [36;1m 乙 = 安(元, レ)[0m  
2026-01-11T07:36:00.1111237Z [36;1m も 甲 == 乙:[0m  
2026-01-11T07:36:00.1111402Z [36;1m レ[旗] = 1[0m  
2026-01-11T07:36:00.1111555Z [36;1m 他:[0m  
2026-01-11T07:36:00.1111717Z [36;1m レ[旗] = 0[0m  
2026-01-11T07:36:00.1111863Z [36;1m [0m  
2026-01-11T07:36:00.1111999Z [36;1m 或 技 == 零:[0m  
2026-01-11T07:36:00.1112151Z [36;1m も レ.拾(旗, 0) == 1:[0m  
2026-01-11T07:36:00.1112731Z [36;1m 指 = 辞[部[1]][0m  
2026-01-11T07:36:00.1112908Z [36;1m[0m  
2026-01-11T07:36:00.1113055Z [36;1m 或 技 == 飛:[0m  
2026-01-11T07:36:00.1113216Z [36;1m 指 = 辞[部[1]][0m  
2026-01-11T07:36:00.1113646Z [36;1m [0m  
2026-01-11T07:36:00.1113791Z [36;1m 或 技 == 加:[0m  
2026-01-11T07:36:00.1113946Z [36;1m 先 = 部[1][0m  
2026-01-11T07:36:00.1114091Z [36;1m 元 = 部[2][0m  
2026-01-11T07:36:00.1114258Z [36;1m 甲 = 安(先, レ)[0m  
2026-01-11T07:36:00.1114416Z [36;1m 乙 = 安(元, レ)[0m  
2026-01-11T07:36:00.1114577Z [36;1m レ[先] = 甲 + 乙[0m  
2026-01-11T07:36:00.1114739Z [36;1m [0m  
2026-01-11T07:36:00.1114872Z [36;1m 指 = 指 + 1[0m  
2026-01-11T07:36:00.1115025Z [36;1m[0m  
2026-01-11T07:36:00.1115162Z [36;1m 外 = 系.係[0m  
2026-01-11T07:36:00.1115313Z [36;1m も 寸(外) < 2:[0m  
2026-01-11T07:36:00.1115472Z [36;1m 表(説)[0m  
2026-01-11T07:36:00.1115619Z [36;1m 系.終(1)[0m  
2026-01-11T07:36:00.1115766Z [36;1m[0m  
2026-01-11T07:36:00.1115902Z [36;1m 径 = 外[1][0m

2026-01-11T07:36:00.1116051Z [36;1m本 = 開(径, 毛, 号=権).読() [0m  
2026-01-11T07:36:00.1116234Z [36;1m生 = 本.行() [0m  
2026-01-11T07:36:00.1116385Z [36;1m口 = [] [0m  
2026-01-11T07:36:00.1116526Z [36;1m[0m  
2026-01-11T07:36:00.1116659Z [36;1m順 = 0 [0m  
2026-01-11T07:36:00.1116799Z [36;1m循順 < 寸(生): [0m  
2026-01-11T07:36:00.1116957Z [36;1m線 = 生[順] [0m  
2026-01-11T07:36:00.1117102Z [36;1m線 = 線.削() [0m  
2026-01-11T07:36:00.1117397Z [36;1mも寸(線) > 0: [0m  
2026-01-11T07:36:00.1117551Z [36;1m口.追(線) [0m  
2026-01-11T07:36:00.1117705Z [36;1m順 = 順 + 1 [0m  
2026-01-11T07:36:00.1117850Z [36;1m[0m  
2026-01-11T07:36:00.1117985Z [36;1m動(口) [0m  
2026-01-11T07:36:00.1118125Z [36;1mEOF [0m  
2026-01-11T07:36:00.1118268Z [36;1m[0m  
2026-01-11T07:36:00.1118398Z [36;1m[0m  
2026-01-11T07:36:00.1118521Z [36;1m[0m  
2026-01-11T07:36:00.1118650Z [36;1m[0m  
2026-01-11T07:36:00.1118879Z [36;1m# Compile and Run Mock VM (DEBUG: Print error if failed) [0m  
2026-01-11T07:36:00.1119520Z [36;1mpython stage2\_compiler.py vm\_win\_mock.py 1 >  
vm\_win\_mock.py || (echo "--- Mock Spec Compilation Failed ---" && cat vm\_win\_mock.py &&  
exit 1) [0m  
2026-01-11T07:36:00.1120109Z [36;1mpython vm\_win\_mock.py fizzbuzz\_win.ir [0m  
2026-01-11T07:36:00.1137316Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o  
pipefail {0}  
2026-01-11T07:36:00.1137682Z env:  
2026-01-11T07:36:00.1137846Z PYTHONIOENCODING: utf-8  
2026-01-11T07:36:00.1138046Z PYTHONUTF8: 1  
2026-01-11T07:36:00.1138226Z PYTHONUNBUFFERED: 1  
2026-01-11T07:36:00.1138487Z pythonLocation: C:  
\\hostedtoolcache\\windows\\Python\\3.10.11\\x64  
2026-01-11T07:36:00.1138897Z PKG\_CONFIG\_PATH: C:  
\\hostedtoolcache\\windows\\Python\\3.10.11\\x64\\lib\\pkgconfig  
2026-01-11T07:36:00.1139335Z Python\_ROOT\_DIR: C:  
\\hostedtoolcache\\windows\\Python\\3.10.11\\x64  
2026-01-11T07:36:00.1139818Z Python2\_ROOT\_DIR: C:  
\\hostedtoolcache\\windows\\Python\\3.10.11\\x64  
2026-01-11T07:36:00.1140188Z Python3\_ROOT\_DIR: C:  
\\hostedtoolcache\\windows\\Python\\3.10.11\\x64  
2026-01-11T07:36:00.1140474Z ##[endgroup]

2026-01-11T07:36:00.3541754Z Mock:1  
2026-01-11T07:36:00.3542014Z Mock:2  
2026-01-11T07:36:00.3542476Z Mock:Fizz  
2026-01-11T07:36:00.3542710Z Mock:4  
2026-01-11T07:36:00.3542908Z Mock:Buzz  
2026-01-11T07:36:00.3543166Z Mock:Fizz  
2026-01-11T07:36:00.3543567Z Mock:7  
2026-01-11T07:36:00.3543974Z Mock:8  
2026-01-11T07:36:00.3544281Z Mock:Fizz  
2026-01-11T07:36:00.3544528Z Mock:Buzz  
2026-01-11T07:36:00.3545418Z Mock:1;  
2026-01-11T07:36:00.3545707Z Mock:Fizz  
2026-01-11T07:36:00.3545960Z Mock:1=

2026-01-11T07:36:00.3546185Z Mock:1>  
2026-01-11T07:36:00.3546412Z Mock:FizzBuzz  
2026-01-11T07:36:00.3547235Z Mock:1@  
2026-01-11T07:36:00.3767854Z ##[group]Run choco install nasm -y  
2026-01-11T07:36:00.3768207Z [36;1mchoco install nasm -y[0m  
2026-01-11T07:36:00.3768484Z [36;1mecho "C:\Program Files\NASM" >> \$GITHUB\_PATH[0m  
2026-01-11T07:36:00.3784614Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e  
-o pipefail {0}  
2026-01-11T07:36:00.3784984Z env:  
2026-01-11T07:36:00.3785146Z PYTHONIOENCODING: utf-8  
2026-01-11T07:36:00.3785348Z PYTHONUTF8: 1  
2026-01-11T07:36:00.3785515Z PYTHONUNBUFFERED: 1  
2026-01-11T07:36:00.3785789Z pythonLocation: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:00.3786217Z PKG\_CONFIG\_PATH: C:  
|hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig  
2026-01-11T07:36:00.3787082Z Python\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:00.3787478Z Python2\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:00.3787853Z Python3\_ROOT\_DIR: C:  
|hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:00.3788175Z ##[endgroup]  
2026-01-11T07:36:01.9441582Z Chocolatey v2.6.0  
2026-01-11T07:36:02.1800496Z Installing the following packages:  
2026-01-11T07:36:02.1807966Z nasm  
2026-01-11T07:36:02.1814135Z By installing, you accept licenses for the packages.  
2026-01-11T07:36:04.8712623Z Downloading package from source '<https://>

[community.chocolatey.org/api/v2/](https://community.chocolatey.org/api/v2/)

2026-01-11T07:36:05.0736265Z

2026-01-11T07:36:05.0737214Z Progress: Downloading nasm 3.1.0... %

2026-01-11T07:36:05.0737735Z Progress: Downloading nasm 3.1.0... %

2026-01-11T07:36:05.0738544Z Progress: Downloading nasm 3.1.0... %

2026-01-11T07:36:05.0741202Z Progress: Downloading nasm 3.1.0... %

2026-01-11T07:36:05.0741665Z Progress: Downloading nasm 3.1.0... %

2026-01-11T07:36:05.0742086Z Progress: Downloading nasm 3.1.0... %

2026-01-11T07:36:05.0742832Z Progress: Downloading nasm 3.1.0... %

2026-01-11T07:36:05.0743589Z Progress: Downloading nasm 3.1.0... %

2026-01-11T07:36:05.0743990Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0745180Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0745989Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0746854Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0747664Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0748506Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0749478Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0750763Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0751473Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0752127Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0752680Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0753789Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0754438Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0755414Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0758190Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0758701Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0759129Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0759716Z Progress: Downloading nasm 3.1.0... 1%

2026-01-11T07:36:05.0760074Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0761295Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0761942Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0762401Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0763078Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0763575Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0764017Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0764889Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0765886Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0766359Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0767346Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0768071Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0768690Z Progress: Downloading nasm 3.1.0... 2%

2026-01-11T07:36:05.0772118Z Progress: Downloading nasm 3.1.0... 2%  
2026-01-11T07:36:05.0772608Z Progress: Downloading nasm 3.1.0... 2%  
2026-01-11T07:36:05.0773031Z Progress: Downloading nasm 3.1.0... 3%  
2026-01-11T07:36:05.0773413Z Progress: Downloading nasm 3.1.0... 3%  
2026-01-11T07:36:05.0773793Z Progress: Downloading nasm 3.1.0... 3%  
2026-01-11T07:36:05.0774567Z Progress: Downloading nasm 3.1.0... 3%  
2026-01-11T07:36:05.0775443Z Progress: Downloading nasm 3.1.0... 3%  
2026-01-11T07:36:05.0776578Z Progress: Downloading nasm 3.1.0... 3%  
2026-01-11T07:36:05.0777589Z Progress: Downloading nasm 3.1.0... 4%  
2026-01-11T07:36:05.0778615Z Progress: Downloading nasm 3.1.0... 4%  
2026-01-11T07:36:05.0779589Z Progress: Downloading nasm 3.1.0... 4%  
2026-01-11T07:36:05.0780348Z Progress: Downloading nasm 3.1.0... 4%  
2026-01-11T07:36:05.0781452Z Progress: Downloading nasm 3.1.0... 4%  
2026-01-11T07:36:05.0782493Z Progress: Downloading nasm 3.1.0... 5%  
2026-01-11T07:36:05.0783338Z Progress: Downloading nasm 3.1.0... 5%  
2026-01-11T07:36:05.0784364Z Progress: Downloading nasm 3.1.0... 5%  
2026-01-11T07:36:05.0785418Z Progress: Downloading nasm 3.1.0... 5%  
2026-01-11T07:36:05.0786540Z Progress: Downloading nasm 3.1.0... 5%  
2026-01-11T07:36:05.0787729Z Progress: Downloading nasm 3.1.0... 5%  
2026-01-11T07:36:05.0788403Z Progress: Downloading nasm 3.1.0... 6%  
2026-01-11T07:36:05.0789032Z Progress: Downloading nasm 3.1.0... 6%  
2026-01-11T07:36:05.0790590Z Progress: Downloading nasm 3.1.0... 6%  
2026-01-11T07:36:05.0791284Z Progress: Downloading nasm 3.1.0... 6%  
2026-01-11T07:36:05.0791943Z Progress: Downloading nasm 3.1.0... 6%  
2026-01-11T07:36:05.0792487Z Progress: Downloading nasm 3.1.0... 7%  
2026-01-11T07:36:05.0793141Z Progress: Downloading nasm 3.1.0... 7%  
2026-01-11T07:36:05.0793752Z Progress: Downloading nasm 3.1.0... 7%  
2026-01-11T07:36:05.0794422Z Progress: Downloading nasm 3.1.0... 7%  
2026-01-11T07:36:05.0795004Z Progress: Downloading nasm 3.1.0... 7%  
2026-01-11T07:36:05.0795647Z Progress: Downloading nasm 3.1.0... 7%  
2026-01-11T07:36:05.0796116Z Progress: Downloading nasm 3.1.0... 8%  
2026-01-11T07:36:05.0796701Z Progress: Downloading nasm 3.1.0... 8%  
2026-01-11T07:36:05.0797316Z Progress: Downloading nasm 3.1.0... 8%  
2026-01-11T07:36:05.0797899Z Progress: Downloading nasm 3.1.0... 8%  
2026-01-11T07:36:05.0798435Z Progress: Downloading nasm 3.1.0... 8%  
2026-01-11T07:36:05.0799647Z Progress: Downloading nasm 3.1.0... 9%  
2026-01-11T07:36:05.0800307Z Progress: Downloading nasm 3.1.0... 9%  
2026-01-11T07:36:05.0800952Z Progress: Downloading nasm 3.1.0... 9%  
2026-01-11T07:36:05.0801405Z Progress: Downloading nasm 3.1.0... 9%  
2026-01-11T07:36:05.0801844Z Progress: Downloading nasm 3.1.0... 9%  
2026-01-11T07:36:05.0802452Z Progress: Downloading nasm 3.1.0... 9%

2026-01-11T07:36:05.0803082Z Progress: Downloading nasm 3.1.0... 10%  
2026-01-11T07:36:05.0803575Z Progress: Downloading nasm 3.1.0... 10%  
2026-01-11T07:36:05.0804295Z Progress: Downloading nasm 3.1.0... 10%  
2026-01-11T07:36:05.0804765Z Progress: Downloading nasm 3.1.0... 11%  
2026-01-11T07:36:05.0809189Z Progress: Downloading nasm 3.1.0... 11%  
2026-01-11T07:36:05.0809902Z Progress: Downloading nasm 3.1.0... 11%  
2026-01-11T07:36:05.0810330Z Progress: Downloading nasm 3.1.0... 12%  
2026-01-11T07:36:05.0810753Z Progress: Downloading nasm 3.1.0... 12%  
2026-01-11T07:36:05.0811511Z Progress: Downloading nasm 3.1.0... 12%  
2026-01-11T07:36:05.0811944Z Progress: Downloading nasm 3.1.0... 13%  
2026-01-11T07:36:05.0812371Z Progress: Downloading nasm 3.1.0... 13%  
2026-01-11T07:36:05.0812785Z Progress: Downloading nasm 3.1.0... 13%  
2026-01-11T07:36:05.0813187Z Progress: Downloading nasm 3.1.0... 14%  
2026-01-11T07:36:05.0813585Z Progress: Downloading nasm 3.1.0... 14%  
2026-01-11T07:36:05.0814229Z Progress: Downloading nasm 3.1.0... 15%  
2026-01-11T07:36:05.0814655Z Progress: Downloading nasm 3.1.0... 15%  
2026-01-11T07:36:05.0815061Z Progress: Downloading nasm 3.1.0... 15%  
2026-01-11T07:36:05.0815478Z Progress: Downloading nasm 3.1.0... 16%  
2026-01-11T07:36:05.0816946Z Progress: Downloading nasm 3.1.0... 16%  
2026-01-11T07:36:05.0817823Z Progress: Downloading nasm 3.1.0... 16%  
2026-01-11T07:36:05.0818298Z Progress: Downloading nasm 3.1.0... 17%  
2026-01-11T07:36:05.0818700Z Progress: Downloading nasm 3.1.0... 17%  
2026-01-11T07:36:05.0819087Z Progress: Downloading nasm 3.1.0... 17%  
2026-01-11T07:36:05.0819461Z Progress: Downloading nasm 3.1.0... 18%  
2026-01-11T07:36:05.0819845Z Progress: Downloading nasm 3.1.0... 18%  
2026-01-11T07:36:05.0820437Z Progress: Downloading nasm 3.1.0... 18%  
2026-01-11T07:36:05.0820810Z Progress: Downloading nasm 3.1.0... 19%  
2026-01-11T07:36:05.0821215Z Progress: Downloading nasm 3.1.0... 19%  
2026-01-11T07:36:05.0821592Z Progress: Downloading nasm 3.1.0... 19%  
2026-01-11T07:36:05.0821972Z Progress: Downloading nasm 3.1.0... 20%  
2026-01-11T07:36:05.0822365Z Progress: Downloading nasm 3.1.0... 20%  
2026-01-11T07:36:05.0822800Z Progress: Downloading nasm 3.1.0... 21%  
2026-01-11T07:36:05.0823203Z Progress: Downloading nasm 3.1.0... 21%  
2026-01-11T07:36:05.0823980Z Progress: Downloading nasm 3.1.0... 21%  
2026-01-11T07:36:05.0825156Z Progress: Downloading nasm 3.1.0... 22%  
2026-01-11T07:36:05.0825571Z Progress: Downloading nasm 3.1.0... 22%  
2026-01-11T07:36:05.0826177Z Progress: Downloading nasm 3.1.0... 22%  
2026-01-11T07:36:05.0827471Z Progress: Downloading nasm 3.1.0... 23%  
2026-01-11T07:36:05.0828192Z Progress: Downloading nasm 3.1.0... 23%  
2026-01-11T07:36:05.0828935Z Progress: Downloading nasm 3.1.0... 23%  
2026-01-11T07:36:05.0829600Z Progress: Downloading nasm 3.1.0... 24%

2026-01-11T07:36:05.0829999Z Progress: Downloading nasm 3.1.0... 24%  
2026-01-11T07:36:05.0830390Z Progress: Downloading nasm 3.1.0... 24%  
2026-01-11T07:36:05.0830785Z Progress: Downloading nasm 3.1.0... 25%  
2026-01-11T07:36:05.0831175Z Progress: Downloading nasm 3.1.0... 25%  
2026-01-11T07:36:05.0832100Z Progress: Downloading nasm 3.1.0... 25%  
2026-01-11T07:36:05.0832517Z Progress: Downloading nasm 3.1.0... 26%  
2026-01-11T07:36:05.0832870Z Progress: Downloading nasm 3.1.0... 26%  
2026-01-11T07:36:05.0833456Z Progress: Downloading nasm 3.1.0... 27%  
2026-01-11T07:36:05.0833918Z Progress: Downloading nasm 3.1.0... 27%  
2026-01-11T07:36:05.0834326Z Progress: Downloading nasm 3.1.0... 27%  
2026-01-11T07:36:05.0834721Z Progress: Downloading nasm 3.1.0... 28%  
2026-01-11T07:36:05.0835100Z Progress: Downloading nasm 3.1.0... 28%  
2026-01-11T07:36:05.0835506Z Progress: Downloading nasm 3.1.0... 28%  
2026-01-11T07:36:05.0835902Z Progress: Downloading nasm 3.1.0... 29%  
2026-01-11T07:36:05.0836276Z Progress: Downloading nasm 3.1.0... 29%  
2026-01-11T07:36:05.0836663Z Progress: Downloading nasm 3.1.0... 29%  
2026-01-11T07:36:05.0837062Z Progress: Downloading nasm 3.1.0... 30%  
2026-01-11T07:36:05.0837448Z Progress: Downloading nasm 3.1.0... 30%  
2026-01-11T07:36:05.0837867Z Progress: Downloading nasm 3.1.0... 30%  
2026-01-11T07:36:05.0838253Z Progress: Downloading nasm 3.1.0... 31%  
2026-01-11T07:36:05.0838637Z Progress: Downloading nasm 3.1.0... 31%  
2026-01-11T07:36:05.0839021Z Progress: Downloading nasm 3.1.0... 31%  
2026-01-11T07:36:05.0839407Z Progress: Downloading nasm 3.1.0... 32%  
2026-01-11T07:36:05.0839796Z Progress: Downloading nasm 3.1.0... 32%  
2026-01-11T07:36:05.0840174Z Progress: Downloading nasm 3.1.0... 33%  
2026-01-11T07:36:05.0840570Z Progress: Downloading nasm 3.1.0... 33%  
2026-01-11T07:36:05.0840948Z Progress: Downloading nasm 3.1.0... 33%  
2026-01-11T07:36:05.0841353Z Progress: Downloading nasm 3.1.0... 33%  
2026-01-11T07:36:05.0841778Z Progress: Downloading nasm 3.1.0... 34%  
2026-01-11T07:36:05.0842179Z Progress: Downloading nasm 3.1.0... 34%  
2026-01-11T07:36:05.0843101Z Progress: Downloading nasm 3.1.0... 34%  
2026-01-11T07:36:05.0843782Z Progress: Downloading nasm 3.1.0... 35%  
2026-01-11T07:36:05.0844225Z Progress: Downloading nasm 3.1.0... 35%  
2026-01-11T07:36:05.0844654Z Progress: Downloading nasm 3.1.0... 35%  
2026-01-11T07:36:05.0845039Z Progress: Downloading nasm 3.1.0... 36%  
2026-01-11T07:36:05.0845634Z Progress: Downloading nasm 3.1.0... 36%  
2026-01-11T07:36:05.0846076Z Progress: Downloading nasm 3.1.0... 36%  
2026-01-11T07:36:05.0846778Z Progress: Downloading nasm 3.1.0... 37%  
2026-01-11T07:36:05.0847193Z Progress: Downloading nasm 3.1.0... 37%  
2026-01-11T07:36:05.0847615Z Progress: Downloading nasm 3.1.0... 38%  
2026-01-11T07:36:05.0848083Z Progress: Downloading nasm 3.1.0... 38%

2026-01-11T07:36:05.0849515Z Progress: Downloading nasm 3.1.0... 38%  
2026-01-11T07:36:05.0850089Z Progress: Downloading nasm 3.1.0... 39%  
2026-01-11T07:36:05.0852857Z Progress: Downloading nasm 3.1.0... 39%  
2026-01-11T07:36:05.0853275Z Progress: Downloading nasm 3.1.0... 39%  
2026-01-11T07:36:05.0853673Z Progress: Downloading nasm 3.1.0... 40%  
2026-01-11T07:36:05.0854073Z Progress: Downloading nasm 3.1.0... 40%  
2026-01-11T07:36:05.0855074Z Progress: Downloading nasm 3.1.0... 40%  
2026-01-11T07:36:05.0855486Z Progress: Downloading nasm 3.1.0... 41%  
2026-01-11T07:36:05.0855882Z Progress: Downloading nasm 3.1.0... 41%  
2026-01-11T07:36:05.0856271Z Progress: Downloading nasm 3.1.0... 41%  
2026-01-11T07:36:05.0856677Z Progress: Downloading nasm 3.1.0... 42%  
2026-01-11T07:36:05.0857055Z Progress: Downloading nasm 3.1.0... 42%  
2026-01-11T07:36:05.0857489Z Progress: Downloading nasm 3.1.0... 43%  
2026-01-11T07:36:05.0857907Z Progress: Downloading nasm 3.1.0... 43%  
2026-01-11T07:36:05.0858313Z Progress: Downloading nasm 3.1.0... 43%  
2026-01-11T07:36:05.0858740Z Progress: Downloading nasm 3.1.0... 44%  
2026-01-11T07:36:05.0859137Z Progress: Downloading nasm 3.1.0... 44%  
2026-01-11T07:36:05.0859526Z Progress: Downloading nasm 3.1.0... 44%  
2026-01-11T07:36:05.0859906Z Progress: Downloading nasm 3.1.0... 45%  
2026-01-11T07:36:05.0860298Z Progress: Downloading nasm 3.1.0... 45%  
2026-01-11T07:36:05.0860686Z Progress: Downloading nasm 3.1.0... 45%  
2026-01-11T07:36:05.0861335Z Progress: Downloading nasm 3.1.0... 46%  
2026-01-11T07:36:05.0861752Z Progress: Downloading nasm 3.1.0... 46%  
2026-01-11T07:36:05.0862200Z Progress: Downloading nasm 3.1.0... 46%  
2026-01-11T07:36:05.0862615Z Progress: Downloading nasm 3.1.0... 47%  
2026-01-11T07:36:05.0863048Z Progress: Downloading nasm 3.1.0... 47%  
2026-01-11T07:36:05.0863457Z Progress: Downloading nasm 3.1.0... 47%  
2026-01-11T07:36:05.0863869Z Progress: Downloading nasm 3.1.0... 48%  
2026-01-11T07:36:05.0864250Z Progress: Downloading nasm 3.1.0... 48%  
2026-01-11T07:36:05.0864648Z Progress: Downloading nasm 3.1.0... 48%  
2026-01-11T07:36:05.0865041Z Progress: Downloading nasm 3.1.0... 49%  
2026-01-11T07:36:05.0865426Z Progress: Downloading nasm 3.1.0... 49%  
2026-01-11T07:36:05.0865812Z Progress: Downloading nasm 3.1.0... 49%  
2026-01-11T07:36:05.0866192Z Progress: Downloading nasm 3.1.0... 50%  
2026-01-11T07:36:05.0866844Z Progress: Downloading nasm 3.1.0... 50%  
2026-01-11T07:36:05.0867238Z Progress: Downloading nasm 3.1.0... 51%  
2026-01-11T07:36:05.0867634Z Progress: Downloading nasm 3.1.0... 51%  
2026-01-11T07:36:05.0868026Z Progress: Downloading nasm 3.1.0... 51%  
2026-01-11T07:36:05.0868409Z Progress: Downloading nasm 3.1.0... 52%  
2026-01-11T07:36:05.0868801Z Progress: Downloading nasm 3.1.0... 52%  
2026-01-11T07:36:05.0869177Z Progress: Downloading nasm 3.1.0... 52%

2026-01-11T07:36:05.0869555Z Progress: Downloading nasm 3.1.0... 53%  
2026-01-11T07:36:05.0869948Z Progress: Downloading nasm 3.1.0... 53%  
2026-01-11T07:36:05.0870324Z Progress: Downloading nasm 3.1.0... 53%  
2026-01-11T07:36:05.0870713Z Progress: Downloading nasm 3.1.0... 54%  
2026-01-11T07:36:05.0871103Z Progress: Downloading nasm 3.1.0... 54%  
2026-01-11T07:36:05.0871476Z Progress: Downloading nasm 3.1.0... 54%  
2026-01-11T07:36:05.0871866Z Progress: Downloading nasm 3.1.0... 55%  
2026-01-11T07:36:05.0872245Z Progress: Downloading nasm 3.1.0... 55%  
2026-01-11T07:36:05.0872663Z Progress: Downloading nasm 3.1.0... 56%  
2026-01-11T07:36:05.0873822Z Progress: Downloading nasm 3.1.0... 56%  
2026-01-11T07:36:05.0874448Z Progress: Downloading nasm 3.1.0... 56%  
2026-01-11T07:36:05.0875909Z Progress: Downloading nasm 3.1.0... 57%  
2026-01-11T07:36:05.0876547Z Progress: Downloading nasm 3.1.0... 57%  
2026-01-11T07:36:05.0877746Z Progress: Downloading nasm 3.1.0... 57%  
2026-01-11T07:36:05.0878281Z Progress: Downloading nasm 3.1.0... 58%  
2026-01-11T07:36:05.0878689Z Progress: Downloading nasm 3.1.0... 58%  
2026-01-11T07:36:05.0879099Z Progress: Downloading nasm 3.1.0... 58%  
2026-01-11T07:36:05.0879749Z Progress: Downloading nasm 3.1.0... 59%  
2026-01-11T07:36:05.0880231Z Progress: Downloading nasm 3.1.0... 59%  
2026-01-11T07:36:05.0880875Z Progress: Downloading nasm 3.1.0... 59%  
2026-01-11T07:36:05.0881372Z Progress: Downloading nasm 3.1.0... 60%  
2026-01-11T07:36:05.0882043Z Progress: Downloading nasm 3.1.0... 60%  
2026-01-11T07:36:05.0882516Z Progress: Downloading nasm 3.1.0... 60%  
2026-01-11T07:36:05.0883136Z Progress: Downloading nasm 3.1.0... 61%  
2026-01-11T07:36:05.0883723Z Progress: Downloading nasm 3.1.0... 61%  
2026-01-11T07:36:05.0884361Z Progress: Downloading nasm 3.1.0... 62%  
2026-01-11T07:36:05.0884985Z Progress: Downloading nasm 3.1.0... 62%  
2026-01-11T07:36:05.0885804Z Progress: Downloading nasm 3.1.0... 62%  
2026-01-11T07:36:05.0887425Z Progress: Downloading nasm 3.1.0... 63%  
2026-01-11T07:36:05.0888047Z Progress: Downloading nasm 3.1.0... 63%  
2026-01-11T07:36:05.0888657Z Progress: Downloading nasm 3.1.0... 63%  
2026-01-11T07:36:05.0889096Z Progress: Downloading nasm 3.1.0... 64%  
2026-01-11T07:36:05.0889585Z Progress: Downloading nasm 3.1.0... 64%  
2026-01-11T07:36:05.0890220Z Progress: Downloading nasm 3.1.0... 64%  
2026-01-11T07:36:05.0890670Z Progress: Downloading nasm 3.1.0... 65%  
2026-01-11T07:36:05.0891236Z Progress: Downloading nasm 3.1.0... 65%  
2026-01-11T07:36:05.0892163Z Progress: Downloading nasm 3.1.0... 65%  
2026-01-11T07:36:05.0893921Z Progress: Downloading nasm 3.1.0... 66%  
2026-01-11T07:36:05.0895356Z Progress: Downloading nasm 3.1.0... 66%  
2026-01-11T07:36:05.0895994Z Progress: Downloading nasm 3.1.0... 66%  
2026-01-11T07:36:05.0896428Z Progress: Downloading nasm 3.1.0... 67%

2026-01-11T07:36:05.0897384Z Progress: Downloading nasm 3.1.0... 67%  
2026-01-11T07:36:05.0897874Z Progress: Downloading nasm 3.1.0... 67%  
2026-01-11T07:36:05.0898329Z Progress: Downloading nasm 3.1.0... 68%  
2026-01-11T07:36:05.0898767Z Progress: Downloading nasm 3.1.0... 68%  
2026-01-11T07:36:05.0899204Z Progress: Downloading nasm 3.1.0... 68%  
2026-01-11T07:36:05.0899656Z Progress: Downloading nasm 3.1.0... 69%  
2026-01-11T07:36:05.0900066Z Progress: Downloading nasm 3.1.0... 69%  
2026-01-11T07:36:05.0900461Z Progress: Downloading nasm 3.1.0... 70%  
2026-01-11T07:36:05.0900843Z Progress: Downloading nasm 3.1.0... 70%  
2026-01-11T07:36:05.0901238Z Progress: Downloading nasm 3.1.0... 70%  
2026-01-11T07:36:05.0901613Z Progress: Downloading nasm 3.1.0... 71%  
2026-01-11T07:36:05.0901994Z Progress: Downloading nasm 3.1.0... 71%  
2026-01-11T07:36:05.0902432Z Progress: Downloading nasm 3.1.0... 71%  
2026-01-11T07:36:05.0902872Z Progress: Downloading nasm 3.1.0... 72%  
2026-01-11T07:36:05.0903302Z Progress: Downloading nasm 3.1.0... 72%  
2026-01-11T07:36:05.0903724Z Progress: Downloading nasm 3.1.0... 72%  
2026-01-11T07:36:05.0904148Z Progress: Downloading nasm 3.1.0... 73%  
2026-01-11T07:36:05.0904564Z Progress: Downloading nasm 3.1.0... 73%  
2026-01-11T07:36:05.0904984Z Progress: Downloading nasm 3.1.0... 73%  
2026-01-11T07:36:05.0905441Z Progress: Downloading nasm 3.1.0... 74%  
2026-01-11T07:36:05.0905913Z Progress: Downloading nasm 3.1.0... 74%  
2026-01-11T07:36:05.0906979Z Progress: Downloading nasm 3.1.0... 74%  
2026-01-11T07:36:05.0908130Z Progress: Downloading nasm 3.1.0... 75%  
2026-01-11T07:36:05.0909285Z Progress: Downloading nasm 3.1.0... 75%  
2026-01-11T07:36:05.0910939Z Progress: Downloading nasm 3.1.0... 76%  
2026-01-11T07:36:05.0911424Z Progress: Downloading nasm 3.1.0... 76%  
2026-01-11T07:36:05.0911868Z Progress: Downloading nasm 3.1.0... 76%  
2026-01-11T07:36:05.0912316Z Progress: Downloading nasm 3.1.0... 77%  
2026-01-11T07:36:05.0913123Z Progress: Downloading nasm 3.1.0... 77%  
2026-01-11T07:36:05.0914527Z Progress: Downloading nasm 3.1.0... 77%  
2026-01-11T07:36:05.0915869Z Progress: Downloading nasm 3.1.0... 78%  
2026-01-11T07:36:05.0916342Z Progress: Downloading nasm 3.1.0... 78%  
2026-01-11T07:36:05.0916772Z Progress: Downloading nasm 3.1.0... 78%  
2026-01-11T07:36:05.0917504Z Progress: Downloading nasm 3.1.0... 79%  
2026-01-11T07:36:05.0917921Z Progress: Downloading nasm 3.1.0... 79%  
2026-01-11T07:36:05.0918339Z Progress: Downloading nasm 3.1.0... 79%  
2026-01-11T07:36:05.0918751Z Progress: Downloading nasm 3.1.0... 80%  
2026-01-11T07:36:05.0919166Z Progress: Downloading nasm 3.1.0... 80%  
2026-01-11T07:36:05.0919581Z Progress: Downloading nasm 3.1.0... 81%  
2026-01-11T07:36:05.0920034Z Progress: Downloading nasm 3.1.0... 81%  
2026-01-11T07:36:05.0920442Z Progress: Downloading nasm 3.1.0... 81%

2026-01-11T07:36:05.0920854Z Progress: Downloading nasm 3.1.0... 82%  
2026-01-11T07:36:05.0921286Z Progress: Downloading nasm 3.1.0... 82%  
2026-01-11T07:36:05.0921701Z Progress: Downloading nasm 3.1.0... 82%  
2026-01-11T07:36:05.0922120Z Progress: Downloading nasm 3.1.0... 83%  
2026-01-11T07:36:05.0922528Z Progress: Downloading nasm 3.1.0... 83%  
2026-01-11T07:36:05.0922923Z Progress: Downloading nasm 3.1.0... 83%  
2026-01-11T07:36:05.0923368Z Progress: Downloading nasm 3.1.0... 84%  
2026-01-11T07:36:05.0923803Z Progress: Downloading nasm 3.1.0... 84%  
2026-01-11T07:36:05.0924259Z Progress: Downloading nasm 3.1.0... 84%  
2026-01-11T07:36:05.0924699Z Progress: Downloading nasm 3.1.0... 85%  
2026-01-11T07:36:05.0925140Z Progress: Downloading nasm 3.1.0... 85%  
2026-01-11T07:36:05.0925862Z Progress: Downloading nasm 3.1.0... 85%  
2026-01-11T07:36:05.0926663Z Progress: Downloading nasm 3.1.0... 86%  
2026-01-11T07:36:05.0928049Z Progress: Downloading nasm 3.1.0... 86%  
2026-01-11T07:36:05.0928754Z Progress: Downloading nasm 3.1.0... 87%  
2026-01-11T07:36:05.0929763Z Progress: Downloading nasm 3.1.0... 87%  
2026-01-11T07:36:05.0930168Z Progress: Downloading nasm 3.1.0... 87%  
2026-01-11T07:36:05.0930563Z Progress: Downloading nasm 3.1.0... 88%  
2026-01-11T07:36:05.0930956Z Progress: Downloading nasm 3.1.0... 88%  
2026-01-11T07:36:05.0931338Z Progress: Downloading nasm 3.1.0... 88%  
2026-01-11T07:36:05.0931727Z Progress: Downloading nasm 3.1.0... 89%  
2026-01-11T07:36:05.0932118Z Progress: Downloading nasm 3.1.0... 89%  
2026-01-11T07:36:05.0932501Z Progress: Downloading nasm 3.1.0... 89%  
2026-01-11T07:36:05.0932893Z Progress: Downloading nasm 3.1.0... 90%  
2026-01-11T07:36:05.0933276Z Progress: Downloading nasm 3.1.0... 90%  
2026-01-11T07:36:05.0933669Z Progress: Downloading nasm 3.1.0... 90%  
2026-01-11T07:36:05.0934047Z Progress: Downloading nasm 3.1.0... 91%  
2026-01-11T07:36:05.0934436Z Progress: Downloading nasm 3.1.0... 91%  
2026-01-11T07:36:05.0934843Z Progress: Downloading nasm 3.1.0... 91%  
2026-01-11T07:36:05.0935232Z Progress: Downloading nasm 3.1.0... 92%  
2026-01-11T07:36:05.0935630Z Progress: Downloading nasm 3.1.0... 92%  
2026-01-11T07:36:05.0936010Z Progress: Downloading nasm 3.1.0... 93%  
2026-01-11T07:36:05.0936403Z Progress: Downloading nasm 3.1.0... 93%  
2026-01-11T07:36:05.0936794Z Progress: Downloading nasm 3.1.0... 93%  
2026-01-11T07:36:05.0937470Z Progress: Downloading nasm 3.1.0... 94%  
2026-01-11T07:36:05.0937883Z Progress: Downloading nasm 3.1.0... 94%  
2026-01-11T07:36:05.0938262Z Progress: Downloading nasm 3.1.0... 94%  
2026-01-11T07:36:05.0938642Z Progress: Downloading nasm 3.1.0... 95%  
2026-01-11T07:36:05.0939032Z Progress: Downloading nasm 3.1.0... 95%  
2026-01-11T07:36:05.0939425Z Progress: Downloading nasm 3.1.0... 95%  
2026-01-11T07:36:05.0939812Z Progress: Downloading nasm 3.1.0... 96%

2026-01-11T07:36:05.0940193Z Progress: Downloading nasm 3.1.0... 96%  
2026-01-11T07:36:05.0940605Z Progress: Downloading nasm 3.1.0... 96%  
2026-01-11T07:36:05.0940988Z Progress: Downloading nasm 3.1.0... 97%  
2026-01-11T07:36:05.0941383Z Progress: Downloading nasm 3.1.0... 97%  
2026-01-11T07:36:05.0942143Z Progress: Downloading nasm 3.1.0... 97%  
2026-01-11T07:36:05.0942627Z Progress: Downloading nasm 3.1.0... 98%  
2026-01-11T07:36:05.0943266Z Progress: Downloading nasm 3.1.0... 98%  
2026-01-11T07:36:05.0943728Z Progress: Downloading nasm 3.1.0... 98%  
2026-01-11T07:36:05.0944142Z Progress: Downloading nasm 3.1.0... 99%  
2026-01-11T07:36:05.0944567Z Progress: Downloading nasm 3.1.0... 99%  
2026-01-11T07:36:05.0944965Z Progress: Downloading nasm 3.1.0... 99%  
2026-01-11T07:36:05.0945399Z Progress: Downloading nasm 3.1.0... 100%  
2026-01-11T07:36:05.0945840Z Progress: Downloading nasm 3.1.0... 100%  
2026-01-11T07:36:05.3416871Z  
2026-01-11T07:36:05.3417339Z nasm v3.1.0 [Approved]  
2026-01-11T07:36:05.3624456Z nasm package files install completed. Performing other installation steps.  
2026-01-11T07:36:06.4236654Z Installing 64-bit nasm...  
2026-01-11T07:36:07.1586372Z nasm has been installed.  
2026-01-11T07:36:07.4043602Z The install of nasm was successful.  
2026-01-11T07:36:07.4048273Z Software installed as 'EXE', install location is likely default.  
2026-01-11T07:36:07.4181736Z  
2026-01-11T07:36:07.4181975Z Chocolatey installed 1/1 packages.  
2026-01-11T07:36:07.4182421Z See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).  
2026-01-11T07:36:07.4954717Z ##[group]Run nasm -v  
2026-01-11T07:36:07.4954981Z [36;1mnasm -v[0m  
2026-01-11T07:36:07.4972146Z shell: C:\Program Files\Git\bin\bash.EXE --noprofile --norc -e -o pipefail {0}  
2026-01-11T07:36:07.4972521Z env:  
2026-01-11T07:36:07.4972678Z PYTHONIOENCODING: utf-8  
2026-01-11T07:36:07.4972879Z PYTHONUTF8: 1  
2026-01-11T07:36:07.4973043Z PYTHONUNBUFFERED: 1  
2026-01-11T07:36:07.4973329Z pythonLocation: C:\hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:07.4973767Z PKG\_CONFIG\_PATH: C:\hostedtoolcache\windows\Python\3.10.11\x64\lib/pkgconfig  
2026-01-11T07:36:07.4974199Z Python\_ROOT\_DIR: C:\hostedtoolcache\windows\Python\3.10.11\x64  
2026-01-11T07:36:07.4974748Z Python2\_ROOT\_DIR: C:\hostedtoolcache\windows\Python\3.10.11\x64

