h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Darmstadt University of Applied Sciences

– Faculty of Computer Science –

Writing a scientific paper in AsciiDoc

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science (B.Sc.)

by
Zebreus
Matriculation number: XXXXXX

First Examiner : Prof. Dr. Some Person

Second Examiner : Prof. Dr. Another Person

DECLARATION

Suppose you are writing a thesis; you probably need this bit to confirm that you wrote it all by yourself. This template adds the `signature-required` CSS class, which adds a nice line where you can write your name.

If you are not writing a thesis, just delete this whole section.

Darmstadt, 5.7.2023

Zebreus

ABSTRACT

This document presents the Asciidoctor.js thesis template, which offers a versatile and easily understandable alternative to traditional typesetting systems for scientific writing. The template leverages the flexibility of web technologies, allowing seamless design modifications and rendering of PDF and website versions of the thesis. The source document structure resembles markdown, enhancing its readability. This abstract provides an overview of the template's benefits, getting started instructions, toolchain details, customization options using JavaScript, integration of source code listings and syntax highlighting, philosophical considerations behind the design decisions, and a comprehensive guide to using the template's features for scientific writing. Overall, the Asciidoctor.js thesis template provides a user-friendly and efficient approach for creating scientific theses, offering enhanced readability and ease of customization compared to traditional typesetting systems.

TABLE OF CONTENTS

LIST OF LISTINGS

LIST OF FIGURESS

LIST OF TABLES

Part I

THESIS

1 INTRODUCTION

Scientific writing plays a vital role in conveying research findings and academic knowledge. As traditional typesetting systems for scientific documents can be complex, there is a growing interest in alternative approaches that prioritize flexibility, readability, and ease of use. The Asciidoctor.js thesis template offers a comprehensive framework for creating well-structured and visually appealing theses. By leveraging web technologies, this template allows for seamless design modifications and facilitates the generation of PDF and website versions. In this introduction, we will explore the key features, benefits, and practical considerations of the Asciidoctor.js thesis template, highlighting its potential to revolutionize scientific writing.

2 FIGURES, TABLES, AND LISTINGS

Figures, tables, and listings are automatically numbered and added to their respective list after the table of contents.

2.1 USING FIGURES

All chart types except for vega-lite should just be used as asciidoctor-kroki charts. You can reference your chart in the text like Figure 1, by giving it an id, in this case, `sample-nomnoml-chart`.

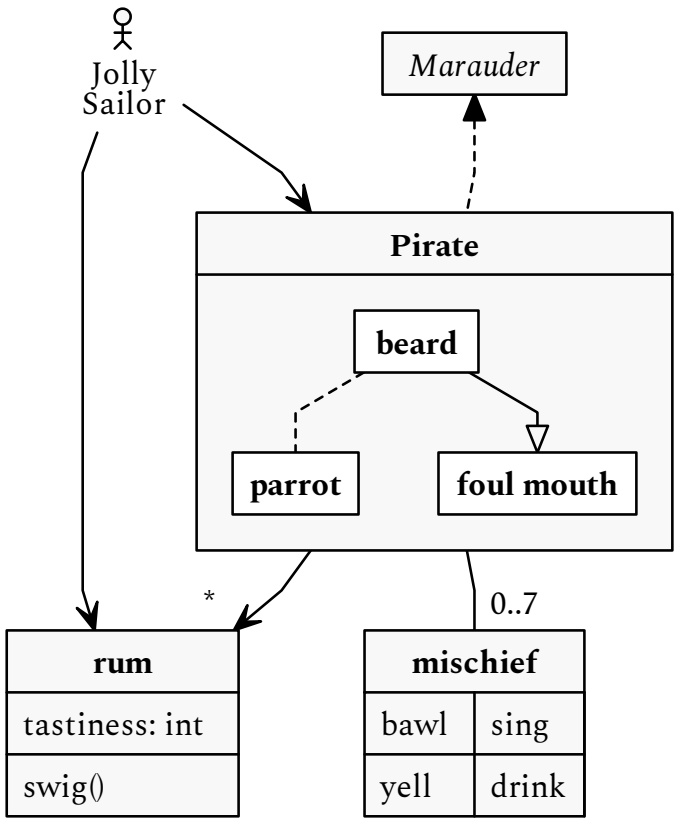

Figure 1. Sample nomnoml chart

Listing 1. Source for the section above

ASCIIDOC

```
All chart types except for vega-lite should just be used as asciidoctor-
kroki charts. You can reference your chart in the text like <<sample-nomnoml-
chart>>, by giving it an id, in this case, `sample-nomnoml-chart`.

.Sample nomnoml chart
[nomnoml,id=sample-nomnoml-chart]
....
```

```
[<actor>Jolly;Sailor]
[Jolly;Sailor]->[Pirate]
[Jolly;Sailor]->[rum]

[Pirate|
  [beard]--[parrot]
  [beard]-:>[foul mouth]
]
[Pirate]-> *[rum|tastiness: int|swig()]
[<abstract>Marauder]<:--[Pirate]

[<table>mischief| bawl | sing || yell | drink ]
[Pirate] - 0..7[mischief]

#gutter: 10
#lineWidth: 1.25
#stroke: #000000
#font: Spectral
#fill: #f7f8f7; #ffffff; #f7f8f7; #ffffff; #f7f8f7; #ffffff
....
```

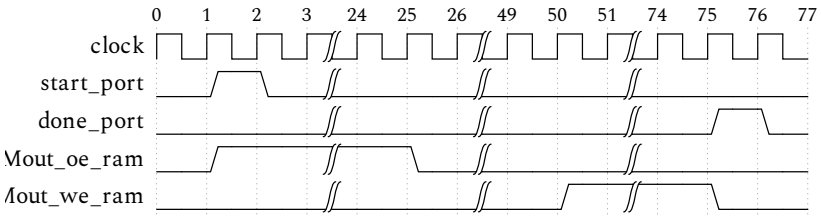

Figure 2. Sample wavedrom chart

Listing 2. Source for the wavedrom chart

ASCIIDOC

```
.Sample wavedrom chart
[wavedrom,id=sample-wavedrom-chart]
....
include::assets/keccak_clang_speed.wavejson.json[]
....
```

To make the chart extend over the margins of the page, add a `slightly-oversized`, `oversized`, or `completely-oversized` to the chart.

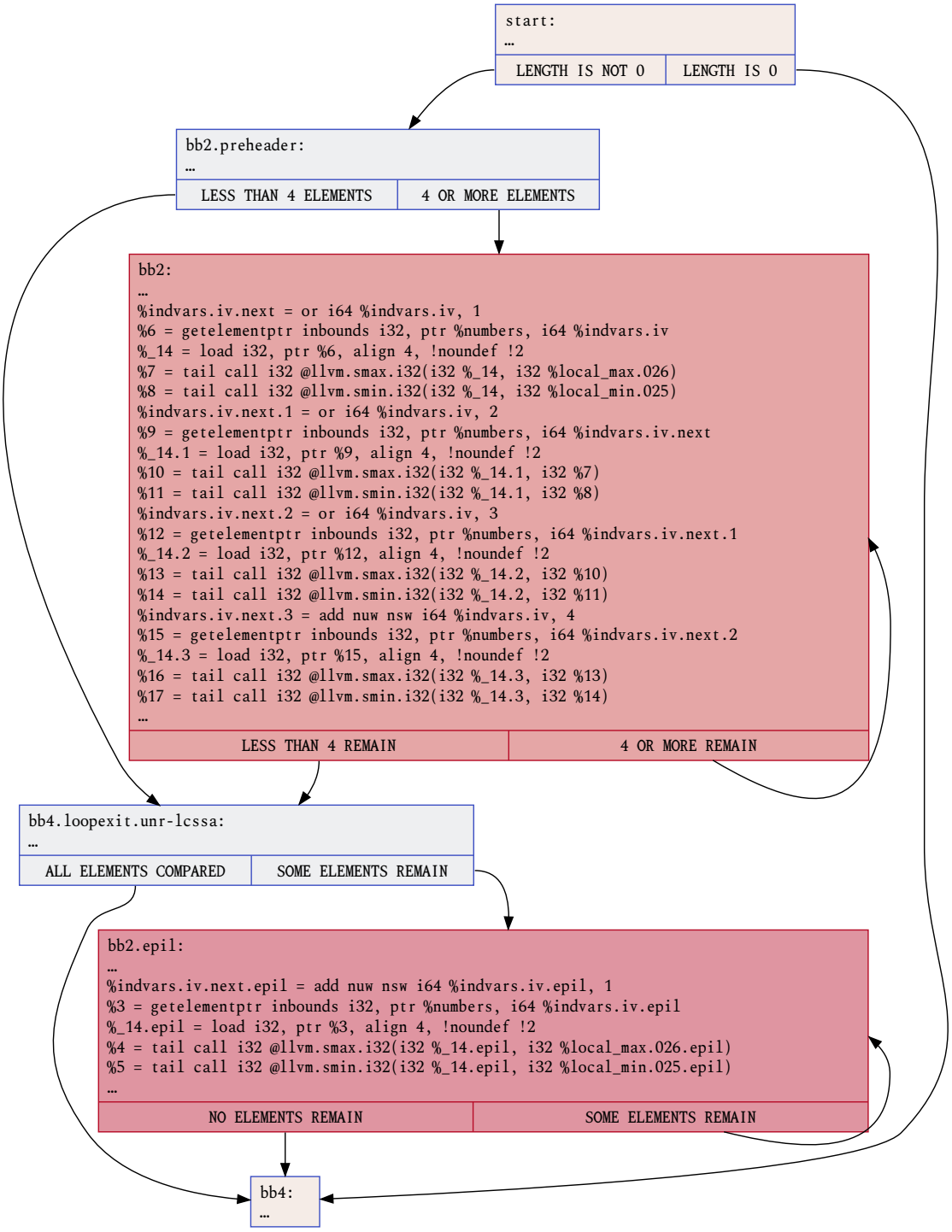

Figure 3. Sample graphviz graph

ASCIIDOC

```
.Sample graphviz graph
[graphviz.slightly-oversized,id=minmax-speed-cfg,width=570px]
....
include::assets/minmax_speed_control_flow.dot[]
....
```

2.1.1 *Data-driven charts*

Vega-lite is the preferred way to display any data-driven charts. You use the included `vega-chart.adoc` script to include vega-lite charts. It detects if the document is currently built for a browser or as a PDF. If the document is built for a browser, it will include the chart directly via the Vega javascript library. That way, the chart supports tooltips and other interactive features.

ASCIIDOC
```
.Sample vega-lite chart
:chart-id: id=minmax-area
:vega-lite-filename: processed-assets/minmax_overview_area.vl.json
include::scripts/vega-chart.adoc[]
```

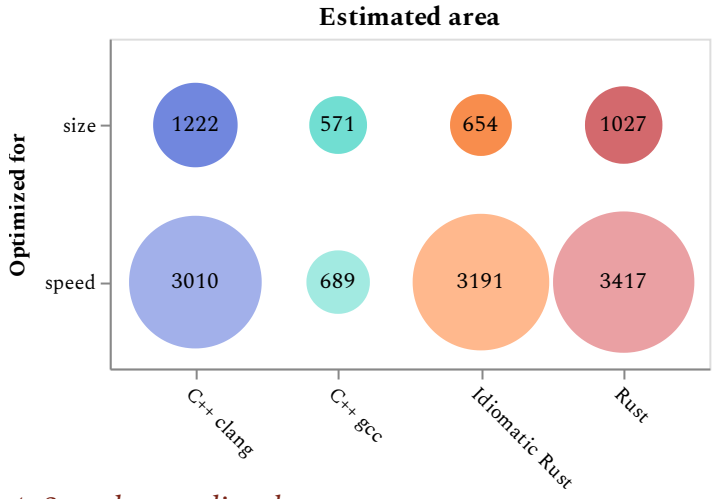

Figure 4. Sample vega-lite chart

2.2 USING SOURCE LISTINGS

Asciidoc also supports source listings. A short verilog listing is shown in Listing 3.

Listing 3. Short Verilog listing

VERILOG
```
module Blinker (input clock, output blinker);
  reg [6:0] counter = 0;
  reg state = 0;
  always @(posedge clock) begin
    counter <= counter + 1;
    if (counter == 9) begin
      state   <= ~state;
      counter <= 0;
    end
  end
  assign blinker = state;
endmodule
```

Listing 4 shows a long Rust listing. This template has a `.linenums` class that can be added to code listings to enable line numbers. The normal AsciiDoc `linenums` attribute is not supported.

Listing 4. Long rust listing with line numbers

RUST

```rust
const KECCAK_ROUND_CONSTANTS: [u64; 24] = [
    0x0000000000000001u64,
    0x0000000000008082u64,
    0x800000000000808au64,
    0x8000000080008000u64,
    0x000000000000808bu64,
    0x0000000080000001u64,
    0x8000000080008081u64,
    0x8000000000008009u64,
    0x000000000000008au64,
    0x0000000000000088u64,
    0x0000000080008009u64,
    0x000000008000000au64,
    0x000000008000808bu64,
    0x800000000000008bu64,
    0x8000000000008089u64,
    0x8000000000008003u64,
    0x8000000000008002u64,
    0x8000000000000080u64,
    0x000000000000800au64,
    0x800000008000000au64,
    0x8000000080008081u64,
    0x8000000000008080u64,
    0x0000000080000001u64,
    0x8000000080008008u64,
];

const KECCAK_RHO_OFFSETS: [u8; 25] = [
    0, 1, 62, 28, 27, 36, 44, 6, 55, 20, 3, 10, 43, 25, 39, 41, 45, 15, 21, 8,
    18, 2, 61, 56, 14,
];

macro_rules! index {
    ($x:expr, $y:expr) => {
        (($x) % 5) + 5 * (($y) % 5)
    };
}

macro_rules! rol64 {
    ($a:expr, $offset:expr) => {
        (if ($offset != 0) {
            (((u64::from($a)) << $offset) ^ ((u64::from($a)) >> (64 - $offset)))
        } else {
            $a
        })
    };
}
```

```
48
49 fn theta(a: &mut [u64; 25]) -> () {
50     let mut c: [u64; 5] = [0; 5];
51     let mut d: [u64; 5] = [0; 5];
52
53     for x in 0..5 {
54         for y in 0..5 {
55             c[x] ^= a[index!(x, y)];
56         }
57     }
58     for x in 0..5 {
59         d[x] = rol64!(c[(x + 1) % 5], 1) ^ c[(x + 4) % 5];
60     }
61     for x in 0..5 {
62         for y in 0..5 {
63             a[index!(x, y)] ^= d[x];
64         }
65     }
66 }
67
68 fn rho(a: &mut [u64; 25]) -> () {
69     for x in 0..5 {
70         for y in 0..5 {
71             a[index!(x, y)] =
72                 rol64!(a[index!(x, y)], KECCAK_RHO_OFFSETS[index!(x, y)]);
73         }
74     }
75 }
76
77 fn pi(a: &mut [u64; 25]) -> () {
78     let mut temp_a: [u64; 25] = [0; 25];
79
80     for x in 0..5 {
81         for y in 0..5 {
82             temp_a[index!(x, y)] = a[index!(x, y)];
83         }
84     }
85
86     for x in 0..5 {
87         for y in 0..5 {
88             a[index!(0 * x + 1 * y, 2 * x + 3 * y)] = temp_a[index!(x, y)];
89         }
90     }
91 }
92
93 fn chi(a: &mut [u64; 25]) -> () {
94     let mut c: [u64; 5] = [0; 5];
95
96     for y in 0..5 {
97         for x in 0..5 {
98             c[x] = a[index!(x, y)]
99                 ^ ((!a[index!(x + 1, y)]) & a[index!(x + 2, y)]);
100        }
101        for x in 0..5 {
```

```
            a[index!(x, y)] = c[x];
        }
    }
}

fn iota(a: &mut [u64; 25], index_round: usize) -> () {
    a[0] ^= KECCAK_ROUND_CONSTANTS[index_round];
}

pub unsafe extern "C" fn keccak(a: *mut u64) -> () {
    let a: &mut [u64; 25] = std::mem::transmute(a);
    for i in 0..24 {
        theta(a);
        rho(a);
        pi(a);
        chi(a);
        iota(a, i);
    }
}
```

Your code should be no wider than 80 characters. If it is, try using the `oversized` classes to avoid unnecessary line breaks.

2.3 USING TABLES

Asciidoc supports tables. A simple table is shown in Table 1.

Table 1. Somewhat complex table

Port	Size per channel in bits	Description
`Mout_oe_ram`	1	Set to 1 to read from the channel.
`Mout_we_ram`	1	Set to 1 to write to the channel.
`Mout_data_ram_size`	$\log_2(\text{dataWidth}) + 1$	Set the width of bits that should be written to the memory. It can be a value between 0 and the width of your data.
`Mout_addr_ram`	addressWidth	Select the address this channel should operate on.
`M_Wdata_ram`	dataWidth	Contains the data that will be written to memory if `Mout_we_ram` is set.
`M_Rdata_ram`	dataWidth	Contains the data that was read from memory if `Mout_oe_ram` was set in the last cycle.
`M_DataRdy`	1	Nonzero if the memory is not ready.

3 REFERENCING OTHER PARTS OF THE DOCUMENT

Asciidoc supports referencing other parts of the document. To reference other parts of the document, you can use the `[id]` syntax. The template styles them like this: Table 1 and Listing 3 and Figure 2 and Section 2.1 and [invalid-reference]

3.1 USING ABBREVIATIONS

When you first use abbreviations, you should introduce them like field-programmable gate array (FPGA). You should also add them to the List of abbreviations. Every abbreviation added to that list will automatically be linked when it is used in the document.

For example, we can just use FPGA, HLS, RAII, and LLVM IR in any sentence, and they will be linked without any special markup.

I also recommend adding a link to a more detailed web source like Wikipedia to every entry in the list of abbreviations. I usually have the list of abbreviations as the first section after the main content.

To see an example of a list of abbreviations, look at the source for List of abbreviations.

3.2 CITATIONS AND BIBLIOGRAPHY

This template uses the built-in bibliography support of Asciidoctor. It is relatively basic, but I found it to work really well in practice. The Readme goes into more detail on how you can structure your bibliography; alternatively, you can directly look into the source of this document.

To cite a source, just reference it like a figure, table, or section. References to sources should grammatically not be a part of your sentence.

Example

The Rust programming language [Kla23] is a modern systems programming language. It is the most loved programming language of the last years [Sta16] [Sta20] [Sta23]. On an unrelated note, Microsoft uses FPGAs to accelerate their Bing search engine [Nan16], and Bambu [Fer21] is a framework for HLS. I don't think Microsoft uses Bambu for their FPGAs; they probably use SystemVerilog, the most common language for hardware design [Soz22].

3.3 REFERENCING THINGS FROM THE APPENDIX

Just add an extra chapter named Appendix after the references and dump your figures and whatnot there. You can reference them like any other figure. Listing 5 was banished into the appendix because I needed to put something there as an example.

4 CONCLUSION

In conclusion, the Asciidoctor.js thesis template offers a flexible, readable, and user-friendly approach to scientific writing. By leveraging web technologies, the template enables easy customization and modification of the document's design. The inclusion of interactive figures using asciidoctor-kroki and vega-lite enhances the visual presentation of data and improves the overall reading experience. The template's support for source listings and syntax highlighting ensures clear and legible code representation. It is important to note that this section was generated with an AI language model trained by OpenAI because a conclusion does not make sense in a dummy document like this. With its comprehensive documentation and practical examples, the Asciidoctor.js thesis template empowers users to create well-structured and visually appealing scientific theses. By combining the simplicity of markdown-like syntax with the power of web technologies, this template sets a new standard for scientific writing, making the process more accessible and enjoyable for researchers and students alike.

5 FUTURE WORK

This template is by no means perfect; there are still some things that could be improved. For example, the template does not support footnotes because I don't need them. Images are also untested but should work exactly like figures. Admonitions look acceptable but could definitely be fancier. But who uses admonitions in scientific writing anyway? There are probably more AsciiDoc features I don't use that may not work properly. As the default stylesheets are included in the template, most should still look fine. I also do not like blue links. Maybe I should style them differently.

The template could also be adjusted to work well with Asciidoctor slides, but I have not tried that yet. Thanks for reading to the end, have a nice day and a cookie 🍪.

LIST OF ABBREVIATIONS

FPGA
: Field-Programmable Gate Array 🔗

HLS
: High-Level Synthesis 🔗

LLVM IR
: LLVM Intermediate Representation 🔗

RAII
: Resource Acquisition Is Initialization / Scope-Bound Resource Management 🔗

REFERENCES

[Kla23] Steve Klabnik, Carol Nichols
The Rust programming language
[Online; accessed 5.7.23]
doc.rust-lang.org/stable/book

[Sta16] Stack Overflow
Stack Overflow Developer Survey 2016
[Online; accessed 5.7.23]
insights.stackoverflow.com/survey/2016

[Sta20] Stack Overflow
Stack Overflow Developer Survey 2020
[Online; accessed 5.7.23]
insights.stackoverflow.com/survey/2020

[Sta23] Stack Overflow
Stack Overflow Developer Survey 2023
[Online; accessed 5.7.23]
survey.stackoverflow.co/2023

[Nan16] Razvan Nane, Vlad-Mihai Sima, Christian Pilato, Jongsok Choi, Blair Fort, Andrew Canis, Yu Ting Chen, Hsuan Hsiao, Stephen Brown, Fabrizio Ferrandi, Jason Anderson, Koen Bertels
A Survey and Evaluation of FPGA High-Level Synthesis Tools
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems
10.1109/tcad.2015.2513673

[Fer21] Fabrizio Ferrandi, Vito Giovanni Castellana, Serena Curzel, Pietro Fezzardi, Michele Fiorito, Marco Lattuada, Marco Minutoli, Christian Pilato, Antonino Tumeo
Invited: Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications
ACM/IEEE Design Automation Conference
10.1109/DAC18074.2021.9586110

[Soz22] Emanuele Del Sozzo, Davide Conficconi, Alberto Zeni, Mirko Salaris, Donatella Sciuto, Marco D. Santambrogio
Pushing the level of abstraction of digital system design: A survey on how to program FPGAs
ACM Computing Surveys
10.1145/3532989

6 APPENDIX

Listing 5. Extra long rust listing with line numbers

```rust
/*
 * The Keccak sponge function, designed by Guido Bertoni, Joan Daemen,
 * Michaël Peeters and Gilles Van Assche. For more information, feedback or
 * questions, please refer to our website: http://keccak.noekeon.org/
 * Implementation by the designers,
 * hereby denoted as "the implementer".
 * To the extent possible under law, the implementer has waived all copyright
 * and related or neighboring rights to the source code in this file.
 * http://creativecommons.org/publicdomain/zero/1.0/
 *
 * This implementation is ported to more idiomatic rust compared to the other one.
 */
// tag::function[]
const KECCAK_ROUND_CONSTANTS: [u64; 24] = [
    0x0000000000000001u64,
    0x0000000000008082u64,
    0x800000000000808au64,
    0x8000000080008000u64,
    0x000000000000808bu64,
    0x0000000080000001u64,
    0x8000000080008081u64,
    0x8000000000008009u64,
    0x000000000000008au64,
    0x0000000000000088u64,
    0x0000000080008009u64,
    0x000000008000000au64,
    0x000000008000808bu64,
    0x800000000000008bu64,
    0x8000000000008089u64,
    0x8000000000008003u64,
    0x8000000000008002u64,
    0x8000000000000080u64,
    0x000000000000800au64,
    0x800000008000000au64,
    0x8000000080008081u64,
    0x8000000000008080u64,
    0x0000000080000001u64,
    0x8000000080008008u64,
];

const KECCAK_RHO_OFFSETS: [u8; 25] = [
    0, 1, 62, 28, 27, 36, 44, 6, 55, 20, 3, 10, 43, 25, 39, 41, 45, 15, 21, 8,
    18, 2, 61, 56, 14,
];

macro_rules! index {
    ($x:expr, $y:expr) => {
```

```
        (($x) % 5) + 5 * (($y) % 5)
    };
}

macro_rules! rol64 {
    ($a:expr, $offset:expr) => {
        (if ($offset != 0) {
            (((u64::from($a)) << $offset) ^ ((u64::from($a)) >> (64 - $offset)))
        } else {
            $a
        })
    };
}

fn theta(a: &mut [u64; 25]) -> () {
    let mut c: [u64; 5] = [0; 5];
    let mut d: [u64; 5] = [0; 5];

    for x in 0..5 {
        for y in 0..5 {
            c[x] ^= a[index!(x, y)];
        }
    }
    for x in 0..5 {
        d[x] = rol64!(c[(x + 1) % 5], 1) ^ c[(x + 4) % 5];
    }
    for x in 0..5 {
        for y in 0..5 {
            a[index!(x, y)] ^= d[x];
        }
    }
}

fn rho(a: &mut [u64; 25]) -> () {
    for x in 0..5 {
        for y in 0..5 {
            a[index!(x, y)] =
                rol64!(a[index!(x, y)], KECCAK_RHO_OFFSETS[index!(x, y)]);
        }
    }
}

fn pi(a: &mut [u64; 25]) -> () {
    let mut temp_a: [u64; 25] = [0; 25];

    for x in 0..5 {
        for y in 0..5 {
            temp_a[index!(x, y)] = a[index!(x, y)];
        }
    }

    for x in 0..5 {
        for y in 0..5 {
            a[index!(0 * x + 1 * y, 2 * x + 3 * y)] = temp_a[index!(x, y)];
```

```
        }
    }
}

fn chi(a: &mut [u64; 25]) -> () {
    let mut c: [u64; 5] = [0; 5];

    for y in 0..5 {
        for x in 0..5 {
            c[x] = a[index!(x, y)]
                ^ ((!a[index!(x + 1, y)]) & a[index!(x + 2, y)]);
        }
        for x in 0..5 {
            a[index!(x, y)] = c[x];
        }
    }
}

fn iota(a: &mut [u64; 25], index_round: usize) -> () {
    a[0] ^= KECCAK_ROUND_CONSTANTS[index_round];
}

pub unsafe extern "C" fn keccak(a: *mut u64) -> () {
    let a: &mut [u64; 25] = std::mem::transmute(a);
    for i in 0..24 {
        theta(a);
        rho(a);
        pi(a);
        chi(a);
        iota(a, i);
    }
}
// end::function[]

#[cfg(test)]
mod tests {

    use super::keccak;

    #[test]
    fn hashing_zeroes_creates_expected_result() {
        let mut input = [0u64; 25];
        let expected_result: [u64; 25] = [
            0xF1258F7940E1DDE7,
            0x84D5CCF933C0478A,
            0xD598261EA65AA9EE,
            0xBD1547306F80494D,
            0x8B284E056253D057,
            0xFF97A42D7F8E6FD4,
            0x90FEE5A0A44647C4,
            0x8C5BDA0CD6192E76,
            0xAD30A6F71B19059C,
```

```
154             0x30935AB7D08FFC64,
155             0xEB5AA93F2317D635,
156             0xA9A6E6260D712103,
157             0x81A57C16DBCF555F,
158             0x43B831CD0347C826,
159             0x01F22F1A11A5569F,
160             0x05E5635A21D9AE61,
161             0x64BEFEF28CC970F2,
162             0x613670957BC46611,
163             0xB87C5A554FD00ECB,
164             0x8C3EE88A1CCF32C8,
165             0x940C7922AE3A2614,
166             0x1841F924A2C509E4,
167             0x16F53526E70465C2,
168             0x75F644E97F30A13B,
169             0xEAF1FF7B5CECA249,
170         ];
171         unsafe {
172             let input_pointer: *mut u64 = std::mem::transmute(&mut input);
173             keccak(input_pointer);
174             assert_eq!(input, expected_result)
175         }
176     }
177 }
```