```
\documentclass{article}
\title{The application of noisy-channel coding techniques to DNA barcoding (Early
↪  structure/ideas)}
\author{Izaak van Dongen}

\usepackage{fullpage}

\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{commath}

\usepackage[nottoc]{tocbibind}

\usepackage[parfill]{parskip}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\usepackage[square]{natbib}

\usepackage{graphicx}

\graphicspath{ {images/} }

\usepackage{listings}
\usepackage{color}

\definecolor{codegreen}{rgb}{ 0,0.6,0}
\definecolor{codegray}{rgb}{0.5,0.5,0.5}
\definecolor{codepurple}{rgb}{0.58,0,0.82}
\definecolor{backcolour}{rgb}{0.95,0.95,0.92}

\lstdefinestyle{mystyle}{
    backgroundcolor=\color{backcolour},
    commentstyle=\color{codegreen},
    keywordstyle=\color{magenta},
    numberstyle=\tiny\color{codegray},
    stringstyle=\color{codepurple},
    basicstyle=\footnotesize,
    breaklines=true,
    postbreak=\mbox{\textcolor{red}{$\hookrightarrow$}\space},
    captionpos=b,
    keepspaces=true,
    numbers=left,
    numbersep=5pt,
    showspaces=false,
    showstringspaces=false,
    showtabs=false,
    tabsize=2
}
```

```
\lstset{style=mystyle}

\lstset{
    literate={~} {$\sim$}{1}
}
```

Listing 1: Original preamble, from commit `1bc2c47`

An extra row and column, including an extra corner piece is appended like so:

0	1	0	0	1
0	0	0	1	1
0	1	0	1	0
0	1	0	0	1
0	1	0	0	1

Each of the extra bits documents the pairty of its row.

4 The Hamming code

The Hamming code is a binary encoding scheme that uses parity to detect, and correct errors. The insertion of "parity bits" is a common practice in basic encoding. Parity refers to the "oddness" or "evenness" of some data. Commonly, this is determined by the sum of the data modulo 2. For example, "00101" results in a parity bit of 0, because the sum of all the bits is 2, which has a remainder of 0 when divided by 2 (is equal to 0 mod 2).

5 Implementing the Hamming code

The script implementing a simple binary Hamming code is as follows:

```python
#!/usr/bin/env python3

"""
Hamming encoding framework for binary objects, using even parity.
"""

from itertools import count, takewhile

def powers_to(n):
    return takewhile(lambda x: x < n, (1 << i for i in count()))

def hamming_encode(bin_stream):
    pwr = 1
    out = []

    for bit in bin_stream:
        while len(out) + 1 == pwr:
            pwr <<= 1
            out.append(0)
        out.append(bit)

    for i in powers_to(len(out)):
        out[i - 1] = 1 & sum(out[pbit] for pstart in range(i - 1, len(out), i << 1)
                                       for pbit in range(pstart, pstart + i))
    return out
```

Listing 1: Binary Hamming code in Python

This code is accompanied by the following testing scheme:

```python
"""
Unit tests for binary_hamming.py
"""

import unittest
```

```
\documentclass[11pt]{article}
\title{The application of noisy-channel coding techniques to DNA barcoding}
\author{\begin{tabular}{rl}
        Name:& Izaak van Dongen\\
        EP Mentor:& Nicolle Mcnaughton\\
        Tutor:& Paul Ingham\\
        Candidate No:&  6659\\
        \end{tabular}
        }

% so the title can be accessed by fancyhdr (and is automatically correctly
% spelled etc)
\makeatletter
\let\thetitle\@title
\makeatother

% fonts
\usepackage[p,osf]{cochineal}
\usepackage[scale=.95,type1]{cabin}
\usepackage[cochineal,bigdelims,cmintegrals,vvarbb]{newtxmath}
% fixed width font with 80 chars per listing line
\usepackage[scaled=.94]{newtxtt}
\usepackage[cal=boondoxo]{mathalfa}

% make the document take up more of the page
\usepackage[margin=1in,headheight=13.6pt]{geometry}

% no paragraph indent
\usepackage[parfill]{parskip}

% custom document header/footer
\usepackage{fancyhdr}
\usepackage{lastpage}

\pagestyle{fancy}
\fancyhf{}
\lhead{\thetitle}
\rhead{Izaak van Dongen}
\rfoot{Page \thepage\ of \pageref{LastPage}}

% pretty table rules and multirow entries
\usepackage{booktabs}
\usepackage{multirow}

% plotting mathematical functions (needs version request)
\usepackage{pgfplots}
\pgfplotsset{compat=1.15}

% \url function and clickable table of contents. no ugly red boxes though
\usepackage[hidelinks]{hyperref}
```

```
51
52  % maths symbols and other stuff (supersedes the ams* packages)
53  \usepackage{mathtools}
54
55  % for better table of contents stuff, providing the \listof* commands and not
56  % listing the tables in the table of contents
57  \usepackage[nottoc,notlof,notlot]{tocbibind}
58
59  % more advanced handling of utf8 and fonts or something. apparently good to have
60  \usepackage[utf8]{inputenc}
61  \usepackage[T1]{fontenc}
62
63  % bibliography management with square braces for citations
64  \usepackage[square]{natbib}
65
66  % graphics, like eps files and stuff (supersedes graphics)
67  \usepackage{graphicx}
68
69  % used to horizontally align floats
70  \usepackage{subfig}
71
72  % used for figures
73  \usepackage{float}
74
75  % needed for colouring and stuff (xcolor supersedes color)
76  \usepackage{xcolor}
77
78  \definecolor{codegreen}{rgb}{ 0,0.6,0}
79
80  % listings of code
81  \usepackage{minted}
82  \setminted{breaklines,
83             breakbytokenanywhere,
84             linenos
85  }
86  \usemintedstyle{friendly}
87  % bigger line numbers
88  \renewcommand\theFancyVerbLine{\footnotesize\arabic{FancyVerbLine}}
89
90  % that can break across pages while being captioned figures
91  \usepackage{caption}
92  \newenvironment{longlisting}
93  {\addvspace{\baselineskip}\captionsetup{type=listing}}
94  {\addvspace{\baselineskip}}
95
96  % allow maths to break across pages
97  \allowdisplaybreaks
```

Listing 2: Updated preamble, from commit 6c4c392

Parity index	00001	00010	00100	01000	10000
Coverage	00011	00011	00101	01001	10001
	00101	00110	00110	01010	10010
	00111	00111	00111	01011	10011
	01001	01010	01100	01100	10100
	01011	01011	01101	01101	10101
	01101	01110	01110	01110	10110
	01111	01111	01111	01111	10111
	10001	10010	10100	11000	11000
	10011	10011	10101	11001	11001
	10101	10110	10110	11010	11010
	10111	10111	10111	11011	11011
	11001	11010	11100	11100	11100
	11011	11011	11101	11101	11101
	11101	11110	11110	11110	11110
	11111	11111	11111	11111	11111

Table 2: Indices covered by each parity bit shown in binary

The second is shown in figure 2. It represents each covered bit as a filled in square, and each non-covered bit as an empty square, so the whole codeword is shown in every row.

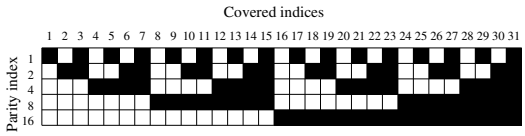

Figure 2: Index coverage of Hamming parity bits

The script implementing a simple binary Hamming code is as follows:

```
#!/usr/bin/env python3


"""
Hamming encoding framework for binary objects, using even parity.
"""

# imports the "count" and "takewhile" functions
from itertools import count, takewhile

# function to get all of the powers of 2 up to a given upper limit.
# Uses count() to produce the set of natural numbers (0, 1, 2, 3..)
# and takewhile() to keep taking powers of 2 until they exceed the limit.
def powers_to(n):
    return takewhile(lambda x: x < n, (1 << i for i in count()))

# function that generates the particular indices covered by a parity bit
```