Contents

Foreword — *page* xiii
Preface — xv

Part I Basic Concepts — 1

1 Introduction — 3
1.1 Functions — 3
1.2 Functional programming — 4
1.3 Features of Haskell — 6
1.4 Historical background — 8
1.5 A taste of Haskell — 9
1.6 Chapter remarks — 13
1.7 Exercises — 13

2 First steps — 14
2.1 Glasgow Haskell Compiler — 14
2.2 Installing and starting — 14
2.3 Standard prelude — 15
2.4 Function application — 16
2.5 Haskell scripts — 17
2.6 Chapter remarks — 21
2.7 Exercises — 21

3 Types and classes — 22
3.1 Basic concepts — 22
3.2 Basic types — 23
3.3 List types — 25
3.4 Tuple types — 26
3.5 Function types — 27
3.6 Curried functions — 27
3.7 Polymorphic types — 29
3.8 Overloaded types — 30
3.9 Basic classes — 31
3.10 Chapter remarks — 36

3.11 Exercises 36

4 Defining functions 38

4.1 New from old 38
4.2 Conditional expressions 38
4.3 Guarded equations 39
4.4 Pattern matching 40
4.5 Lambda expressions 42
4.6 Operator sections 44
4.7 Chapter remarks 45
4.8 Exercises 45

5 List comprehensions 47

5.1 Basic concepts 47
5.2 Guards 48
5.3 The `zip` function 50
5.4 String comprehensions 51
5.5 The Caesar cipher 52
5.6 Chapter remarks 56
5.7 Exercises 57

6 Recursive functions 59

6.1 Basic concepts 59
6.2 Recursion on lists 61
6.3 Multiple arguments 63
6.4 Multiple recursion 64
6.5 Mutual recursion 65
6.6 Advice on recursion 66
6.7 Chapter remarks 71
6.8 Exercises 71

7 Higher-order functions 73

7.1 Basic concepts 73
7.2 Processing lists 74
7.3 The `foldr` function 76
7.4 The `foldl` function 79
7.5 The composition operator 81
7.6 Binary string transmitter 82
7.7 Voting algorithms 86
7.8 Chapter remarks 89
7.9 Exercises 89

8 Declaring types and classes 92

8.1 Type declarations 92

8.2 Data declarations 93
8.3 Newtype declarations 95
8.4 Recursive types 96
8.5 Class and instance declarations 99
8.6 Tautology checker 101
8.7 Abstract machine 106
8.8 Chapter remarks 108
8.9 Exercises 109

9 The countdown problem 111
9.1 Introduction 111
9.2 Arithmetic operators 112
9.3 Numeric expressions 113
9.4 Combinatorial functions 114
9.5 Formalising the problem 115
9.6 Brute force solution 115
9.7 Performance testing 116
9.8 Combining generation and evaluation 117
9.9 Exploiting algebraic properties 118
9.10 Chapter remarks 119
9.11 Exercises 120

Part II Going Further 121

10 Interactive programming 123
10.1 The problem 123
10.2 The solution 124
10.3 Basic actions 125
10.4 Sequencing 126
10.5 Derived primitives 127
10.6 Hangman 128
10.7 Nim 129
10.8 Life 133
10.9 Chapter remarks 137
10.10 Exercises 137

11 Unbeatable tic-tac-toe 139
11.1 Introduction 139
11.2 Basic declarations 140
11.3 Grid utilities 141
11.4 Displaying a grid 142
11.5 Making a move 143
11.6 Reading a number 144
11.7 Human vs human 144

11.8 Game trees 145
11.9 Pruning the tree 147
11.10 Minimax algorithm 148
11.11 Human vs computer 150
11.12 Chapter remarks 151
11.13 Exercises 151

12 Monads and more 153
12.1 Functors 153
12.2 Applicatives 157
12.3 Monads 164
12.4 Chapter remarks 174
12.5 Exercises 175

13 Monadic parsing 177
13.1 What is a parser? 177
13.2 Parsers as functions 177
13.3 Basic definitions 179
13.4 Sequencing parsers 179
13.5 Making choices 181
13.6 Derived primitives 183
13.7 Handling spacing 186
13.8 Arithmetic expressions 187
13.9 Calculator 191
13.10 Chapter remarks 194
13.11 Exercises 194

14 Foldables and friends 196
14.1 Monoids 196
14.2 Foldables 200
14.3 Traversables 206
14.4 Chapter remarks 210
14.5 Exercises 210

15 Lazy evaluation 212
15.1 Introduction 212
15.2 Evaluation strategies 213
15.3 Termination 216
15.4 Number of reductions 217
15.5 Infinite structures 219
15.6 Modular programming 220
15.7 Strict application 223
15.8 Chapter remarks 226
15.9 Exercises 226

16 Reasoning about programs 228

16.1 Equational reasoning 228
16.2 Reasoning about Haskell 229
16.3 Simple examples 230
16.4 Induction on numbers 231
16.5 Induction on lists 234
16.6 Making append vanish 238
16.7 Compiler correctness 241
16.8 Chapter remarks 246
16.9 Exercises 246

17 Calculating compilers 249

17.1 Introduction 249
17.2 Syntax and semantics 249
17.3 Adding a stack 250
17.4 Adding a continuation 252
17.5 Defunctionalising 254
17.6 Combining the steps 257
17.7 Chapter remarks 261
17.8 Exercises 261

Appendix A Selected solutions 263

A.1 Introduction 263
A.2 First steps 264
A.3 Types and classes 265
A.4 Defining functions 266
A.5 List comprehensions 267
A.6 Recursive functions 267
A.7 Higher-order functions 268
A.8 Declaring types and classes 269
A.9 The countdown problem 270
A.10 Interactive programming 270
A.11 Unbeatable tic-tac-toe 271
A.12 Monads and more 272
A.13 Monadic parsing 273
A.14 Foldables and friends 274
A.15 Lazy evaluation 275
A.16 Reasoning about programs 276
A.17 Calculating compilers 279

Appendix B Standard prelude 280

B.1 Basic classes 280
B.2 Booleans 281
B.3 Characters 282

B.4 Strings 283
B.5 Numbers 283
B.6 Tuples 284
B.7 Maybe 284
B.8 Lists 285
B.9 Functions 287
B.10 Input/output 287
B.11 Functors 288
B.12 Applicatives 289
B.13 Monads 290
B.14 Alternatives 290
B.15 MonadPlus 291
B.16 Monoids 292
B.17 Foldables 294
B.18 Traversables 297

Bibliography 298
Index 300