Foundations of Computer Science
Past Papers by topic

Lazy/infinite lists:

- **[2017P1Q2]** Infinite lists and enumeration
- **[2015P1Q2 (a)]** Lazy lists
- **[2010P1Q1 (a),(b),(c)]** Lazy lists, interleave, `map` for infinite lists, + `iterates` and `iterates2`
- **[2008P1Q5 (a)-(d)]** Lazy lists, enumerating a lazy list of lazy lists.
- **[2006P1Q6 (a)]** datatype + filter function
- **[2003P1Q5 (a)]** Lazy lists
- **[2003P1Q5 (b),(c),(d),(e)]** concatenate infinite lists, interleave for infinite lists, lazy list with all zeros and ones, lazy list with all palindromes
- **[2001P1Q6]** Taylor series for infinite list operations
- **[2000P1Q6 (a)]** Memory representation of lists b) cyclic lists in ML

List operations:

- **[2019P1Q2 (a)-(g)]** Nested lists, `flatten`, `nested_map`, `pack_as`, nested lazy lists.
- **[2018P1Q1 (a)]** Representing sets using `lists1`
- **[2016P1Q1 (b)]** `zarg` (essentially `foldr`)
- **[2014P1Q2 (c)]** compute cost difference to convert between two lists.
- **[2014P1Q2 (b)]** Run-length encoding on a list
- **[2012P1Q2 (a)]** replicate item function
- **[2010P1Q2 (a)]** `foldl`
- **[2009P1Q1 (a),(b)]** implement `delFirst`
- **[2008P1Q1 (b)]** determine if a list is a sublist of another list
- **[2007P1Q6 (d)]** replace the k-th instance of an item on a list
- **[2004P1Q1]** map and `foldr`/`foldl`
- **[2003P1Q1 (b),(c)]** `foldr` function and zipping, write a function that returns all elements except those at an index that is a multiple of three
- **[2000P1Q1]** `exf` operation (test which elements have $f(x)$ in the list). Remove duplicates
- **[2000P1Q6 c)]** check if a list is cyclic

Binary trees:

- **[2016P1Q1 (c)]**
- **[2013P1Q1 (b),(c)]** Generate all labelled trees
- **[2008P1Q5 (b),(c),(d)]**
- **[2007P1Q5 (a),(c)]**
- **[2006P1Q6 (b)]** intersection of two binary trees

- **[2005P1Q6]** Finding all paths in a binary tree + lazy listing of paths
- **[2004P1Q5]** Getting all values, enumerating an infinite binary tree
- **[2002P1Q1]** Find a path to a given value in the binary tree. Find all paths to a given value in the binary tree.
- **[1998P1Q6 (b)]** Pre-order traversal

BST:

- **[2012P1Q1]** `union`, `dropSlice`, `takeSlice`
- **[2009P1Q2 (c)]** Does BST A include all entries of BST B? Do it in linear time.
- **[2003P1Q6]** mutable BSTs
- **[1999P1Q5 (b)]** Choosing between different BSTs

Queues:

- **[2014P1Q2 (a)]**
- **[2006P1Q5 (a)]** interface, implementation, amortization

Permutations:

- **[2013P1Q2]** Lazy permutations
- **[2009P1Q1 (e)]** generalised permutation (one element can occur multiple times in the other list)
- **[2009P1Q1 (d)]** determine if L_1 is a permutation of L_2
- **[2006P1Q5 (b)]** compute all permutations of a list
- **[1999P1Q1]** compute all permutations for items of a list.

Reference types:

- **[2012P1Q2 (b),(c)]**
- **[2007P1Q6 (a)]**
- **[2001P1Q1 (a),(b)]**

Datatypes:

- **[2014P1Q1 (b)-(e)]**
- **[2013P1Q1 (a)]**
- **[2011P1Q2 (c)(i)]** find the type of a function with a datatype
- **[2007P1Q6 (b),(c)]**
- **[2000P1Q6 (c)]** ordinary types vs datatypes

Control structures:

- **[2011P1Q2 (a)]**
- **[2007P1Q6 (d)]**

Exceptions:

- **[2011P1Q2 (a),(b),(c)]** (use options instead of Exceptions)
- **[2007P1Q5]**
- **[2001P1Q5 (a)(iii)]**

Sorting:

- **[2010P1Q2 (b)]** implement selection sort
- **[2009P1Q2 (a)]** compare and contrast insertion sort and merge sort
- **[2007P1Q1]** Merge sort
- **[2005P1Q5]** Quicksort
- **[2001P1Q1 (c)]** use filter to implement Quicksort
- **[1998P1Q1]** finding the k smallest items in a list (without using sorting)

Trees:

- **[2009P1Q2]** pre-order, post-order, in-order
- **[2002P1Q5]** flip a tree, map each node of a tree, count the number of nodes in a tree.
- **[2001P1Q5 (a)(ii)]** difference between BFS and DFS

Type inference:

- **[2018P1Q1 (c)]**
- **[2014P1Q2 (b)]** find the type of run-length encoding
- **[2014P1Q1 (d),(e)]** type inference on datatype related functions
- **[2014P1Q1 (a)]** ML polymorphism
- **[2009P1Q1 (c)]** infer the types of `delFirst` (paying attention to currying and equality)

Functions:

- **[2018P1Q1 (b)]**
- **[2016P1Q1 (a)]** brief notes on functions as values and return types
- **[2012P1Q2 (a)]** brief notes on function types and currying
- **[2004P1Q6 (a)]** functions as inputs and outputs
- **[2003P1Q1 (a)]** explain curried functions
- **[2001P1Q5 (a)(i)]** making a function iterative

Pattern matching:

- **[2013P1Q1 (a)]**

Polynomials:

- **[2005P1Q1]** addition and equality testing for multinomials
- **[2001P1Q6]** Taylor series for infinite list operations

Recurrence relations / big-$\mathcal{O}$ notation:

- **[2006P1Q1]** give an example of an OCaml function belonging to each complexity class.
- **[2002P1Q6 (a),(b),(c)]** explain big-$\mathcal{O}$ notation, put complexities in order, binary search to solve the equation
- **[1999P1Q5 (c)]**
- **[1998P1Q6 (a)]** state the definition of big-$\mathcal{O}$ notation.

AdHoc:

- **[2019P1Q1]** Church numerals, Peano arithmetic and binary systems, binary addition
- **[2016P1Q2 (a)]** Write the code for computing the Sieve of Eratosthenes
- **[2010P1Q2 (c),(d)]** Multiplication tables
- **[2001P1Q5 (b)]** Find all possible sums of given integers, c) make sure your output is ordered and with no duplicates

Functional arrays:

- **[2015P1Q1]**
- **[2004P1Q6]**
- **[1993P13Q9]** Arguing that the functional array is a balanced binary tree, counting the number of nodes in each depth of a functional array.

Strings:

- **[2016P1Q2 (b),(c)]** remove duplicate strings (in time faster than quadratic), check if a string can be formed from a set of strings (can use multiple times each string -> exponential search or faster using dynamic programming)

Puzzles/Combinatorial games:

- **[2018P1Q2]**
- **[2017P1Q1]**
- **[2011P1Q1]** Labyrinth puzzle
- **[2008P1Q6]** BFS, DFS for games