COMPUTER SCIENCE TRIPOS Part IA – 2023 – Paper 1

7 Algorithms 1 (fms27)

A DLL (doubly-linked list) may be trivially implemented as a chain of records, each record `r` having a pointer `r.p` to the previous record and a pointer `r.n` to the next. At the machine level, objects are identified by their memory address and pointers are memory addresses, so `r`, `r.p` and `r.n` are all pointers *and* memory addresses, as well as the handles by which we refer to those three records. At the logical level, there are two main classes: the record `r` and the DLL `d` itself—the latter being an object that contains, possibly among other things, a pointer `d.h` to the head record of the list.

An experienced machine code programmer suggests an improvement, the DLLx (Doubly-Linked List with XOR). By replacing the two `r.p` and `r.n` machine words with a single machine word holding their bitwise exclusive-or (`r.pn = r.p` $\oplus$ `r.n`), we save one word per record. The null pointer is represented by the word `0`. [*Hint:* For any three machine words `X, Y, Z`, whenever `X` $\oplus$ `Y == Z`, knowledge of any two of `X, Y, Z` lets you derive the third.]

(*a*) In a DLLx, how exactly do we move from one record to the next? Do we need any extra information to access the successor of record `r`, besides the address of `r` itself? If so, what? Do we need to store any extra information in the DLLx object besides the address of the head record `h`? If so, what? [4 marks]

(*b*) Let `a0`, `a1`, `a2`, ..., `a9` be the addresses of the records of a 10-element DLLx `d`, with `d.h == a0`. Express `a3` as a function of `a0`. [*Hint:* The final expression for `a3` will only use: parentheses, the two operators $\oplus$ and `.pn`, and the constant `a0`.] [4 marks]

(*c*) Give clear and well-commented pseudocode for an iterative function that returns the length of a DLLx `d`, where `d.h` is the address of its head record. [4 marks]

(*d*) Give clear and well-commented pseudocode for a function that inserts a new record `r` as the new head into a DLLx `d`, given `d` and `r`. [4 marks]

(*e*) What changes would you suggest to turn a DLLx into a CDLLx, meaning a *circular* doubly linked list with XOR, in which the successor of the last record is the head? Explain what fields the CDLLx object should contain. Assuming the CDLLx has at least three records, give a short sequence of algebraic expressions yielding the address of the third record from the end, going backwards from the head of the CDLLx. [*Note:* To avoid ambiguity with 0-based indexing: the first record from the end is the last record of the CDLLx. The 0-th record from the end would be the head.] [4 marks]