

## CS2104 Tutorial 4: Data Representation I

- (a) This exercise gets you acquainted with using pointers in C. Using the `cell` structure in the lecture notes (slide 40-42), write C functions `insertFront`, `insertRear`, `deleteFront`, `deleteRear`, so as to insert and delete nodes, both from the front and rear of the list.
  - (b) Write a C program that keeps causing memory leaks until the program runs out of memory to use.
2. The designer of Pascal, Niklaus Wirth, has introduced a keyword ‘upper’ as an optional modifier for array declarations.

```
var x : upper array [l..h] of sometype
```

This will define a 2 dimensional *upper triangular* array. This means that memory will be allocated for

$$x[l, l], x[l, l + 1] \dots x[l, h],$$

and then for

$$x[l + 1, l + 1], x[l + 1, l + 2] \dots x[l + 1, h],$$

and then for

$$x[l + 2, l + 2], x[l + 2, l + 3] \dots x[l + 2, h]$$

and so on. (memory is allocated for the ‘upper triangle’ of the array). Note that memory is *not* allocated for  $x[l + 1, l]$ ,  $x[l + 2, l]$ ,  $x[l + 2, l + 1]$ , etc. (the ‘lower triangle’ of the 2D array)

- (a) Niklaus Wirth claims that there are real situations when you would only need to use the ‘upper triangle’ of a 2D array. Can you give a few examples?
- (b) You are told that the array layout is in a *row-major* format (meaning that after allocating the memory for  $x[l, l]$  to  $x[l, h]$ , the space for  $x[l + 1, l + 1]$  immediately follows the space for  $x[l, h]$ ).
  - (i) Give the accessing formula for  $x[i, j]$ , assuming that the array stores  $n$  bytes of data for each cell;
  - (ii) State what are the pre-computed values that can be stored in the dope vector.
- (c) You are told that before accessing the array  $x[i, j]$ , you need to check that the indices  $i$  and  $j$  are within the bounds of the array.

```
if (...check that i and j are within bounds...) then
    ... access the array x[i,j]...
```

This is known as *array bounds check*. What will be the check conditions?

3. Compare the similarities and differences between the use of pointers and memory between C++ and Java so as to highlight the pros and cons of the different design decisions taken by the two languages.

Sebesta has good discussion questions that are quite thought provoking (pages 287-288). You may like to try them out.