Alpha Chen
CS 3304 Comparative Languages
Professor Edwards
HW 06
October 11th, 2005

Problem 1: Chapter 6, Review Question 9
Define static*,* fixed stack-dynamic*,* stack-dynamic*,* fixed heap-dynamic*, and* heap-dynamic arrays. *What are the advantages of each?*

Static arrays are efficient, not requiring any dynamic allocation or deallocation, since subscript ranges and storage allocation are determined before run time.

A fixed stack-dynamic array has statically bound subscript ranges, but the allocation is done at run time, allowing for better space efficiency.

A stack-dynamic array has subscript ranges and storage allocation done dynamically at run time. However, after allocation, these values are not allowed to change. This has the advantage of being flexible; the array size can be determined at run time.

Fixed heap-dynamic arrays are like stack-dynamic arrays, except that the bindings are done at the request of the program and the stroage is allocaated from a heap, rather than a stack.

Heap-dynamic arrays are fully dynamic, allowing users to change both the subscript ranges and the storage allocation. This is the most flexible of the array types, allowing the array to grow and shrink as needed.

Problem 2: Chapter 6, Review Question 19
Define union*,* free union*, and* discriminated union.

A union is a data structure which can store several types of data at a single location. Free unions are unions where type checking is not used, while discriminated unions are unions with type indicators.

Problem 3: Chapter 6, Problem 10
Multidimensional arrays can be stored in row major order, as in C++, or in column major order, as in Fortran. Develop the access functions for both of these arrangements for three-dimensional arrays.

Let `n` be the number of rows, `o` the number of columns, and `p` the size of the third dimensional array.

Row major order:

```
location(a[i,j,k]) = address of a[0,0,0] +
                     (k + p*(n*j + i))*element_size
```

Column major order:

```
location(a[i,j,k]) = address of a[0,0,0] +
                     (k + p*(j + o*i))*element_size
```

Problem 4: Chapter 6, Problem 11

In the Burroughs Extended ALGOL language, matrixes are stored as a single-dimensioned array of pointers to the rows of the matrix, which are treated as single-dimensioned arrays for values. What are the advantages and disadvantages of such a scheme?

The advantage of this scheme is that memory allocation is more efficient. However, there is the tradeoff of less arithmetic calculations but more loading from memory. The access function would be something like `address stored in (a[0,0] + i) + j` instead of just `address of a[0,0] + i + n*j`.