Answers to End of Section and Review Exercises for Chapter 4

Exercises 4.1

1. Random access uses the base address of an array and the index of a position in the array to locate the cell in memory corresponding to the position. It does this by adding to the base address the result of multiplying the index value by the size of the data value. Because this is a constant time operation, random access is a constant time operation, no matter which position is accessed.
2. An array allows the user to access or replace items with a subscript operation at given positions in a structure of fixed size, whereas a list allows the user to insert or remove items, automatically adjusting the size of the structure if necessary. A list tracks the number of items currently in it and available to the user.
3. The physical size of an array is the number of cells or positions available for storing items, which is fixed when the array is created. The logical size of an array is the number of items currently in it and available to the user, which ranges from 0 to the physical size minus 1.

Exercises 4.2

1. If an item is removed or inserted at the logical end of an array, no other items have to be shifted. If an item is removed anywhere else, the items to its right must be shifted to the left by one position to close the hole vacated by the removed item. If an item is inserted anywhere else, the items from the target index down to the last logical position must be shifted to the right by one position to open a hole for the new item.
2. During an insertion of an item into an array, the item at the last logical position is moved to the right first. This opens a hole for its predecessor so it can be moved to the right, and so on, until a hole is eventually opened at the target position for the inserted item. This is done so that a moved item does not clobber the item next to it.
3. The complexity is O(1) when space is available. If space is unavailable, the complexity is O(n) because of resizing the array. If the array size is doubled every time resizing occurs, the complexity of an insertion at the logical end of the array is O(1) on the average.
4. The array's capacity is 14/.70, or 20.

Exercises 4.3

1. A two-dimensional array is a data structure that organizes data in rows and columns. You can access or replace each item with random access using two subscripts, which specify the row and column of the item.

2. You might use a two-dimensional array to represent a checkerboard in a game-playing program.
3. Here is the code:

```
row = g.getHeight()
column = g.getWidth()
found = False
for r in range(g.getHeight()):
    for c in range(g.getWidth()):
        if g[r][c] < 0:
            row = r
            column = c
            found = True
            break
    if found:
        break
```

4. Each cell contains the product of its row and column.
5. Here is the code:

```
g = Array(3)
g[0] = Array(3)
g[1]] = Array(6)
g[2] = Array(9)
```

6. The `__init__` method expects the three dimensions and a fill value as arguments. This method sets an instance variable to a new grid with the first two dimensions. It then fills the cells in this grid with new arrays whose sizes are equal to the third dimension. Finally, the `__init__` method fills this structure, using three subscripts, with the fill value. The new class also includes a `getDepth` method that returns the value of the third dimension, as well as modified `__getitem__` and `__setitem__` methods that use two subscripts instead of one.
7. Here is the code:

```
for r in range(g.getHeight()):
    for c in range(g.getWidth()):
        for d in range(g.getDepth()):
            g[r][c][d] = r * c * d
```

8. Here is the code:

```
for r in range(g.getHeight()):
    for c in range(g.getWidth()):
        for d in range(g.getDepth()):
            print(g[r][c][d], end = " ")
        print()
```

```
print()
```

Exercises 4.4

1. If an item is removed or inserted at the logical end of an array, no other items have to be shifted. If an item is removed anywhere else, the items to its right must be shifted to the left by one position to close the hole vacated by the removed item. If an item is inserted anywhere else, the items from the target index down to the last logical position must be shifted to the right by one position to open a hole for the new item.
2. When a programmer attempts to access a node's data fields and the node variable refers to `None`, Python raises an exception.
3. Here is the code:

```
head = None
i = len(array) - 1
while i >= 0:
    head = Node(array[i], head)
    i -= 1
```

Exercises 4.5

1. The run-time complexity for removing an item from a singly linked structure after that item has been located is O(1).
2. It is not practical to perform a binary search for an item on a singly linked structure. The run-time cost of locating the midpoint would be linear, so the running time of such a search would be worse than that of a sequential search.
3. A Python list uses an array rather than a linked structure to hold its items because the running times of the access and replacement operations are O(1), and the memory usage is better when more than half of the array's positions are occupied.

Exercises 4.6

1. A circular linked structure with a dummy header node enables the programmer to avoid special cases in the code for inserting or removing items at the head and tail of the structure.
2. One benefit of a doubly linked structure is that the operation to move to a previous item runs in constant time. One cost is that an extra memory cell is required for the second link in each node.

Answers to Review Questions

1. b

2. a
3. a
4. b
5. b
6. b