Practice Exercises

Exercise 1: Pointers and linked structure

a. How to declare a pointer variable (e.g., a pointer that points to a certain type variable)?

b. How to make a pointer variable "points" to some existing variable?

c. How to allocate dynamic variable or array and store its address in a pointer?

d. What's the difference between "*a" and "&a"?

e. If p and q are two variables, what does p=q do? What does *p=*p do?

f. Dynamic array vs static/fixed size array: what's the difference?

g. If p points to a struct/class type variable, two ways to access that variable's certain member variable (or member function):

```
p->value, or
(*p)->value
```

h. A special pointer named this in member functions of a class is initialized to point to the calling (or revoking) object:

```
//constructor for unsorted type built using dynamic array
UnsortedType::UnsortedType (int size)
{
        this->size = size;
        this->info = new ItemType[size]; //same as info=new ItemType[size];
        //...
}
```

Exercise 2: Unsorted Lists using arrays

Implement functions for the following operations with an array-based unsorted list:

a. Search function:
   int findElement(int arr[], int n, int key)

b. Insert function:

   int insertUnsorted(int arr[], int n, int key, int capacity)

c. Delete function:

int deleteElement(int arr[], int n, int key)

Exercise 3: Array-based unsorted lists with pointers

Define a dynamic array as a class to store integers along with the following operations:

```
class Dynarray {
private:
        int *pa;                    // points to the array
        int length;                 // the # elements
        int nextIndex;              // the next highest index value
public:
        Dynarray();                 // the constructor
        ~Dynarray();                //  the  destructor
        int& operator[](int index); // the indexing operation
        void add(int val);          // add a new value to the end
        int size();                 // return length
};
```

The class declares an integer pointer, pa, that will point to the array itself. length is the number of elements in the array, and *nextIndex* is the next available (empty) element. The class will have a default constructor which will initialize the variables, a destructor, which will do clean-up, and three member functions. We will overload the index *operator []* so that we can index our array just like normal arrays and provide a function for adding a new value at the end of the array. Last, size() will return the length of the array.

The class declaration goes in a file *Dynarray.h*, and functions definitions in Dynarray.cpp.