drill-1.cpp – first part of the drill exercise from Chapter 17

drill-2.cpp – second part of the drill exercise from chapter 17

Drill

This drill has two parts. The first exercises/builds your understanding of free-store-allocated arrays and contrasts arrays with **vectors**:

1. Allocate an array of ten **ints** on the free store using **new**.
2. Print the values of the ten **ints** to **cout**.
3. Deallocate the array (using **delete[]**).
4. Write a function **print_array10(ostream& os, int* a)** that prints out the values of **a** (assumed to have ten elements) to **os**.
5. Allocate an array of ten **ints** on the free store; initialize it with the values 100, 101, 102, etc.; and print out its values.
6. Allocate an array of 11 **ints** on the free store; initialize it with the values 100, 101, 102, etc.; and print out its values.
7. Write a function **print_array(ostream& os, int* a, int n)** that prints out the values of **a** (assumed to have **n** elements) to **os**.
8. Allocate an array of 20 ints on the free store; initialize it with the values 100, 101, 102, etc.; and print out its values.
9. Did you remember to delete the arrays? (If not, do it.)
10. Do 5, 6, and 8 using a **vector** instead of an array and a **print_vector()** instead of **print_array()**.

The second part focuses on pointers and their relation to arrays. Using **print_array()** from the last drill:

1. Allocate an **int**, initialize it to 7, and assign its address to a variable **p1**.
2. Print out the value of **p1** and of the **int** it points to.
3. Allocate an array of seven **ints**; initialize it to 1, 2, 4, 8, etc.; and assign its address to a variable **p2**.
4. Print out the value of **p2** and of the array it points to.
5. Declare an **int*** called **p3** and initialize it with **p2**.
6. Assign **p1** to **p2**.
7. Assign **p3** to **p2**.
8. Print out the values of **p1** and **p2** and of what they point to.
9. Deallocate all the memory you allocated from the free store.

10. Allocate an array of ten **ints**; initialize it to 1, 2, 4, 8, etc.; and assign its address to a variable **p1**.
11. Allocate an array of ten **ints**, and assign its address to a variable **p2**.
12. Copy the values from the array pointed to by **p1** into the array pointed to by **p2**.
13. Repeat 10–12 using a **vector** rather than an array.

exercises.cpp – 3,4,5 exercise from Chapter 17

3. Write a function, **void to_lower(char* s)**, that replaces all uppercase characters in the C-style string **s** with their lowercase equivalents. For example, **Hello, World!** becomes **hello, world!**. Do not use any standard library functions. A C-style string is a zero-terminated array of characters, so if you find a **char** with the value **0** you are at the end.
4. Write a function, **char* strdup(const char*)**, that copies a C-style string into memory it allocates on the free store. Do not use any standard library functions.
5. Write a function, **char* findx(const char* s, const char* x)**, that finds the first occurrence of the C-style string **x** in **s**.