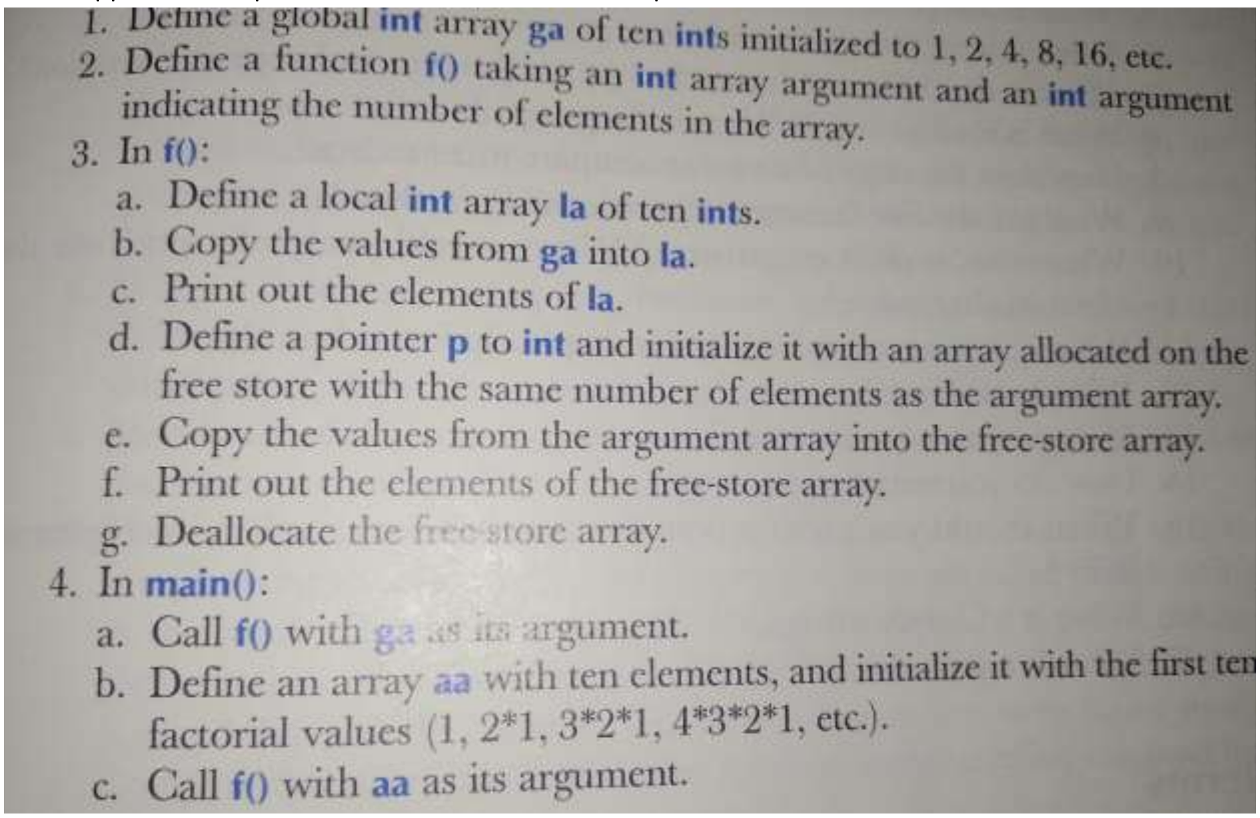
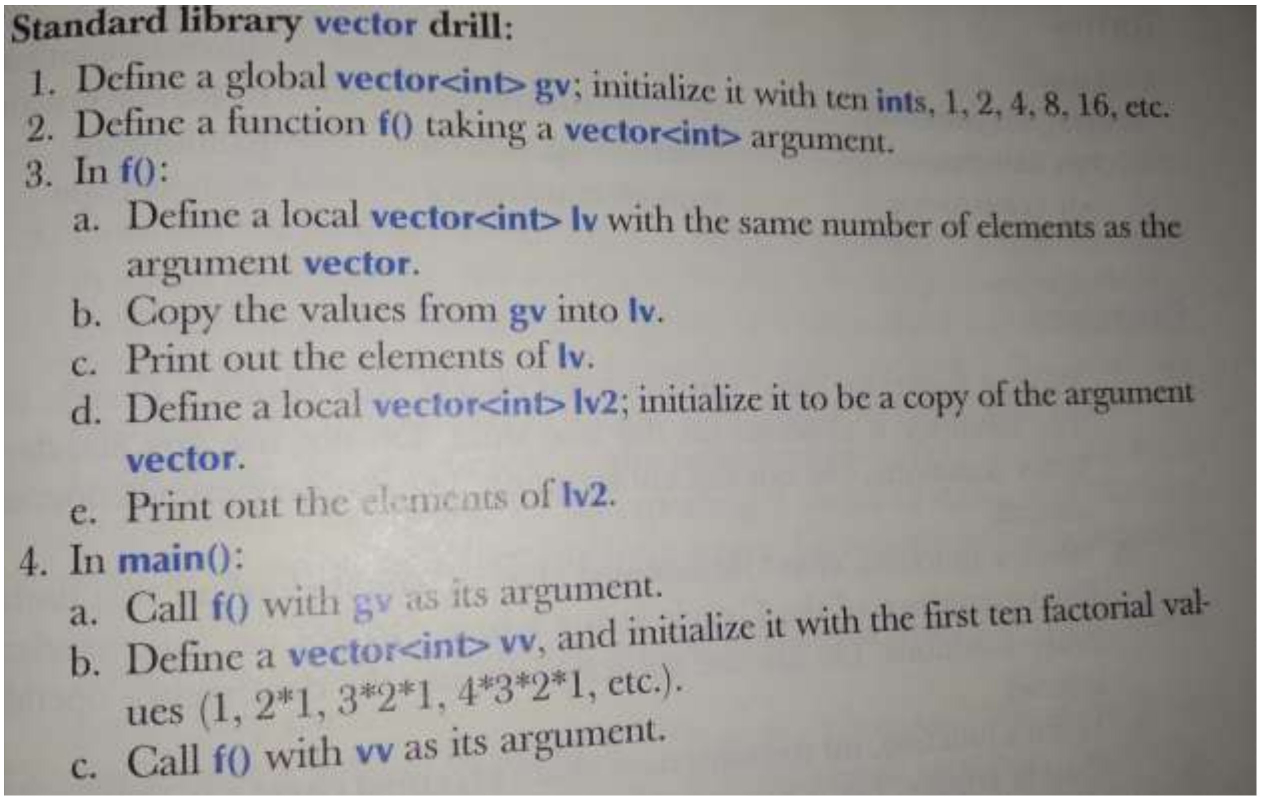
drill-1.cpp – the first part of the drill exercise from Chapter 18



1. Define a global **int** array **ga** of ten **int**s initialized to 1, 2, 4, 8, 16, etc.
2. Define a function **f()** taking an **int** array argument and an **int** argument indicating the number of elements in the array.
3. In **f()**:
   a. Define a local **int** array **la** of ten **int**s.
   b. Copy the values from **ga** into **la**.
   c. Print out the elements of **la**.
   d. Define a pointer **p** to **int** and initialize it with an array allocated on the free store with the same number of elements as the argument array.
   e. Copy the values from the argument array into the free-store array.
   f. Print out the elements of the free-store array.
   g. Deallocate the free-store array.
4. In **main()**:
   a. Call **f()** with **ga** as its argument.
   b. Define an array **aa** with ten elements, and initialize it with the first ten factorial values (1, 2*1, 3*2*1, 4*3*2*1, etc.).
   c. Call **f()** with **aa** as its argument.

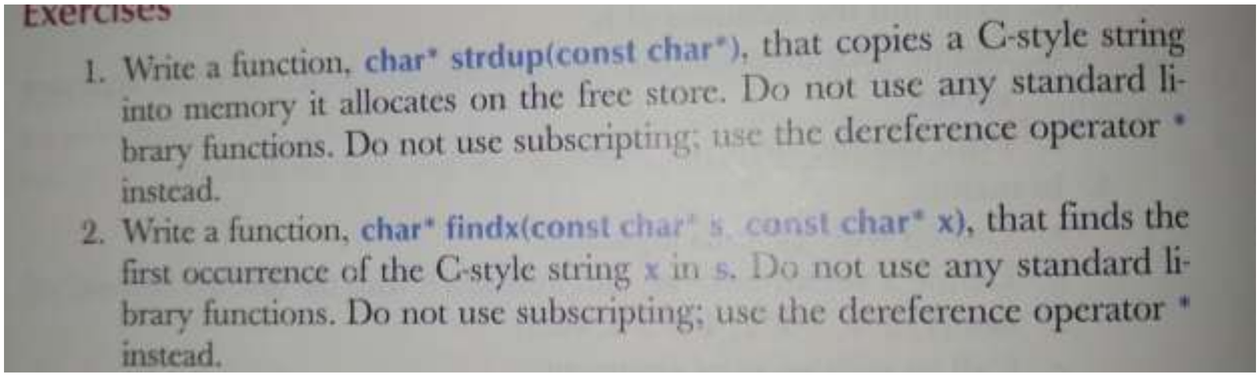
drill-2.cpp – the second part of the drill exercise from Chapter 18



Standard library vector drill:

1. Define a global **vector<int> gv**; initialize it with ten **int**s, 1, 2, 4, 8, 16, etc.
2. Define a function **f()** taking a **vector<int>** argument.
3. In **f()**:
   a. Define a local **vector<int> lv** with the same number of elements as the argument **vector**.
   b. Copy the values from **gv** into **lv**.
   c. Print out the elements of **lv**.
   d. Define a local **vector<int> lv2**; initialize it to be a copy of the argument **vector**.
   e. Print out the elements of **lv2**.
4. In **main()**:
   a. Call **f()** with **gv** as its argument.
   b. Define a **vector<int> vv**, and initialize it with the first ten factorial values (1, 2*1, 3*2*1, 4*3*2*1, etc.).
   c. Call **f()** with **vv** as its argument.

exercises.cpp – exercises 1,2 from Chapter 18



Exercises

1. 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. Do not use subscripting; use the dereference operator * instead.
2. Write a function, **char* findx(const char* s, const char* x)**, that finds the first occurrence of the C-style string x in s. Do not use any standard library functions. Do not use subscripting; use the dereference operator * instead.